

DESIGN OF SEARCH SYSTEM FOR ONLINE DIGITAL LIBRARIES

THESIS

submitted in fulfillment of the requirement of the degree of

DOCTOR OF PHILOSOPHY

to

J.C. BOSE UNIVERSITY OF SCIENCE & TECHNOLOGY YMCA

by

SUMITA GUPTA

Registration No: YMCAUST/PH59/2011

Under the Supervision of

Dr. NEELAM DUHAN

Dr. POONAM BANSAL



Department of Computer Engineering

Faculty of Engineering and Technology

**J.C. Bose University of Science & Technology, YMCA
Sector-6, Mathura Road, Faridabad, Haryana, INDIA**

December 2018

DEDICATION

This Thesis is dedicated to my parents

For their endless love, support and encouragement

DECLARATION

I hereby declare that this thesis entitled “**DESIGN OF A SEARCH SYSTEM FOR ONLINE DIGITAL LIBRARIES**” by **SUMITA GUPTA**, being submitted in fulfilment of requirement for the award of Degree of Doctor of Philosophy in the Department of Computer Engineering under Faculty of Engineering and Technology of J.C. Bose University of Science & Technology, YMCA, Faridabad, during the academic year March 2012 to December 2018, is a bonafide record of my original work carried out under the guidance and supervision of **Dr. NEELAM DUHAN** and co-supervision **Dr. POONAM BANSAL** has not been presented elsewhere.

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

(SUMITA GUPTA)

Registration No. YMCAUST/PH59/2011

CERTIFICATE

This is to certify that this thesis entitled “**DESIGN OF A SEARCH SYSTEM FOR ONLINE DIGITAL LIBRARIES**” by **SUMITA GUPTA** submitted in fulfilment of the requirement for the award of Degree of Doctor of Philosophy in **DEPARTMENT OF COMPUTER ENGINEERING**, under Faculty of Engineering and Technology of J.C. Bose University of Science & Technology YMCA, Faridabad, during the academic year March 2012 to December 2018, is a bonafide record of work carried out under our guidance and supervision.

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

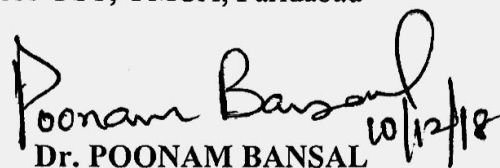
Dr. NEELAM DUHAN

Assistant Professor

Department of Computer Engineering

Faculty of Informatics & Computing

JC Bose UST, YMCA, Faridabad


Dr. POONAM BANSAL

Professor

Department of Computer Science & Engineering

MSIT, Delhi

ACKNOWLEDGEMENT

It gives me immense pleasure to acknowledge the people whose priceless contributions helped me reach here. Without their support, this piece of academic work, in the form of my doctoral thesis, would have just been reduced to mere ink on paper. First and foremost, my deepest and heartfelt thanks to my supervisor **Dr. Neelam Duhani**, not only for her valuable guidance throughout my Ph.D tenure, but also for the care showed me during the hard times. In each and every meeting, we had, with her immense research experience, she brought me encouragement and taught me how to find solution for the various research issues in different ways, which induced me to instigate new ideas into my research. Without her help in terms of technicalities, resources and opportunity, I would not have reached this stage.

I express my deep sense of gratitude to my co-supervisor **Dr. Poonam Bansal**, for her valuable guidance and help without which it was not possible for me to complete my work. Mam, you believed in my abilities and gave unconditional support to help me achieve excellence throughout my research work culminating in this doctoral thesis.

I wish to express my gratitude to **Dr. Komal Kumar Bhatia, Dr. Atul Mishra, Dr. Sapna Gambhir** and **Dr. Parul Tomar** for always motivating me and showing me the right way to pursue my Ph.D. I thank them for all their help and suggestions they showered me with a humble soul. I am indebted them for their constant encouragement and being readily available if I am in a need.

I am whole heartedly thankful to all my *Family Members, Friends* and my *Husband Mr. Geet Gupta* for their support provided in the last few years of my tenure when I was going through a rough phase, both in my professional as well as in my personal life. I wish to thank my *Son Panav* for his unconditional love, affection and the tolerance he has showed all these years.

I owe God for everything!

(SUMITA GUPTA)

Registration No. YMCAUST/PH59/2011

TABLE OF CONTENTS

Declaration	i
Certificate	ii
Acknowledgement	iii
Abstract	iv
Table of Contents	vi
List of Tables	xi
List of Figures	xiii
List of Abbreviations	xvi
Chapter I: INTRODUCTION	1-9
1.1 GENERAL	1
1.2 SEARCH ENGINE	2
1.3 DIGITAL LIBRARIES	2
1.4 MOTIVATION	4
1.5 PROBLEM DEFINITION	5
1.6 OBJECTIVES OF RESEARCH WORK	5
1.7 ORGANIZATION OF THESIS	7
Chapter II: INFORMATION RETRIEVAL & DIGITAL LIBRARY SYSTEMS: A REVIEW	11-48
2.1 INFORMATION RETRIEVAL	11
2.2 SEARCH ENGINES	12
2.2.1 GENERAL ARCHITECTURE OF A SEARCH ENGINE	13
2.2.2 TYPE OF ISSUES WITH THE CURRENT WEB	14
2.3 DIGITAL LIBRARIES: AN INTRODUCTION	16
2.3.1 Benefits of Digital Libraries	16
2.3.2 Principles for Digital Library Design	18
2.4 GENERAL ARCHITECTURE OF DIGITAL LIBRARY SEARCH SYSTEM	18

2.5	CRAWLER	19
2.5.1	Types of Web Crawler	20
2.5.2	Study of Existing Web Crawlers for Digital Libraries	22
2.5.3	Comparison Study of Various Web Crawlers	25
2.6	INDEXER	28
2.6.1	Architecture of Indexing Process	28
2.6.2	Types of Indexing Techniques	29
2.6.3	Study of Recent Indexing Techniques	43
2.6.4	Comparison of Different Indexing Techniques	44
2.7	QUERY PROCESSING	46
2.8	DESIGN ISSUES IN DIGITAL LIBRARY SEARCH ENGINES	47
CHAPTER III: STATE-OF-THE-ART TECHNIQUES IN DIGITAL LIBRARIES		49-85
3.1	INTRODUCTION	49
3.2	PAGE RANKING	49
3.2.1	Citation Count Algorithm	49
3.2.2	Time Dependent Citation Count Algorithm	51
3.2.3	PageRank Algorithm	53
3.2.4	Popularity Weighted Ranking Algorithm	55
3.2.5	HITS Algorithm	56
3.2.6	PaperRank Algorithm	59
3.2.7	Popularity and Similarity based PageRank Algorithm (PSPR)	61
3.2.8	SIMRANK: PageRank Approach Based on Similarity Measure	65
3.2.9	Page Ranking using Social Annotation based on Language Model	67
3.2.10	Comparison Study	69
3.3	WEB DOCUMENT CLUSTERING	69
3.3.1	Major Categories of Clustering	72
3.3.2	Similarity Measures	74
3.4	DOCUMENT CATEGORIZATION	77
3.4.1	Keyword Extraction	78

3.4.2	Different Categorization Techniques	79
3.4.3	Study of Recent Document Categorization Techniques	80
3.4.4	Comparison Study	82
3.5	POSSIBLE APPLICATION AREAS	83
3.6	REVIEW SUMMARY	84

CHAPTER IV: FOCUSED CRAWLER TO HARVEST DIGITAL 87-109

ACADEMIC DOCUMENTS

4.1	GENERAL	87
4.2	PROPOSED CRAWLING PROCESS OF DIGITAL LIBRARIES	87
4.3	PAGE DOWNLOADER	89
4.4	CATEGORIZATION PROCESS	90
4.4.1	Text Extractor	93
4.4.2	Bookmark Creator	93
4.4.3	Keywords Extractor	93
4.4.4	Document Categorizer	94
4.4.5	Topic Taxonomy	94
4.4.6	Incremental Keyword Filter	95
4.4.7	Advantages of Proposed Categorization Process	96
4.5	LINK FORECASTING MODULE	96
4.5.1	Link Extractor	97
4.5.2	Link Filter	97
4.5.3	Link Priority Analyzer	97
4.6	MISSING DOCUMENT FINDER MODULE	98
4.6.1	Query Formation Module	99
4.6.2	Author Homepages Filter	100
4.6.3	Paper Filter	101
4.6.4	Example Illustration	102
4.7	AGING PROCESS	103
4.8	IILLUSTRATION OF PROPOSED CRAWLING SYSTEM	104
4.9	SUMMARY	109

CHAPTER V: MULTI-LEVEL INDEXING TO INDEX DIGITAL DOCUMENTS	111-126
5.1 GENERAL	111
5.2 PROPOSED ARCHITECTURE OF INDEXING	111
5.3 WEB CRAWLER	112
5.4 PRE-PROCESSING MODULE	113
5.4.1 Similarity Analyzer	113
5.4.2 Cluster Generator	114
5.4.3 Illustrative Example	114
5.5 INDEX GENERATOR	117
5.5.1 Index Structure	117
5.5.2 Illustrative Example	118
5.5.3 Data Structures	119
5.6 QUERY PROCESSING ENGINE	121
5.6.1 Query Keyword Extractor	122
5.6.2 Query Analyzer	122
5.6.3 Dynamic Ranking	122
5.6.4 Illustrative Example	123
5.7 SUMMARY	125
CHAPTER VI: SEARCH RESULT ORGANISATION USING CLUSTERING AND RANKING	127-140-
6.1 GENERAL	127
6.2 PROPOSED APPROACH FOR RANKING DOCUMENTS	127
6.3 SIMILARITY MATRIX AND CLUSTERS GENERATION	129
6.4 STATIC RANK CALCULATION	129
6.4.1 Download Score	129
6.4.2 PageRank	130
6.4.3 Bookmark Based Citation Count Rank	132
• Illustration of Proposed Algorithm	132

• Comparison Study	135
• Retrieval of Relevant Papers by BCC	136
6.5 DYNAMIC RANK CALCULATION	136
6.6 ILLUSTRATION OF PROPOSED CLUSTERING AND RANKING MECHANISM	137
6.7 COMPARISON STUDY	139
6.8 SUMMARY	140
CHAPTER VII: IMPLEMENTATION RESULTS AND ANALYSIS	141-164
7.1 GENERAL	141
7.2 PERFORMANCE METRICS	141
7.3 EXPERIMENTAL EVALUATION OF PROPOSED CRAWLER	142
7.4 DIGITAL LIBRARY SYSTEM FOR RETRIEVING RELEVANT RESULTS	150
7.5 DOCUMENT CATEGORIZATION	157
7.6 BOOKMARK BASED CITATION COUNT	162
CHAPTER VIII: CONCLUSION AND FUTURE SCOPE	165-166
8.1 CONCLUSION	165
8.2 FUTURE SCOPE	166
REFERENCES	167
APPENDIX A	182
APPENDIX B	187

LIST OF TABLES

Table		Page No.
Table 2.1	List of some Popular Search Engines	13
Table 2.2	List of some existing Digital Libraries	17
Table 2.3	Comparison of various Web Crawler for Digital Libraries	26
Table 2.4	Comparison of various Web Crawler for Digital Libraries	27
Table 2.5	Term Frequencies in respective Documents	30
Table 2.6	Signature Generation and Comparison	32
Table 2.7	Summary of Indexing Techniques	41
Table 2.8	Different Indexing Methods used by Digital Library	42
Table 2.9	Comparison of Indexing Techniques	45
Table 3.1	Data of Citation Graph	51
Table 3.2	Iteration Method for PageRank	54
Table 3.3	Rank Results of Example Graph	55
Table 3.4	Iteration Method for HITS	59
Table 3.5	Web Session for a Website	63
Table 3.6	1 st Order Transition Probability Matrix	63
Table 3.7	2 nd Order Transition Probability Matrix	64
Table 3.8	Example of Similarity Calculation	64
Table 3.9	Comparison of various Page Ranking Algorithms	70
Table 3.10	Comparison of various Page Ranking Algorithms	71
Table 3.11	Comparison between various Existing Document	82
	Categorization Techniques	
Table 4.1	Description of Topic Taxonomy	95
Table 4.2	Sample Ignore List	97
Table 4.3	Example of Priority Scheduling	103
Table 4.4	Parsed Documents	104
Table 4.5	Keywords	105
Table 4.6	Frequency of Keyword	106
Table 4.7	Extracted Link with their Similarity Score	107
Table 4.8	Summary of Proposed System	109
Table 5.1	Term Frequencies and their Weights	116
Table 5.2	Similarity Matrix	117
Table 5.3	Description of Paper Repository	120
Table 5.4	Description of Clustering _Database	120
Table 5.5	Description of Index	121
Table 5.6	Keywords in Main Categories	123
Table 5.7	Cosine Similarity Values	123
Table 5.8	Keywords of Sub-Category	124

Table 5.9	Cosine Similarity Values	124
Table 5.10	Similarity Value between Clusters and Query Terms	125
Table 5.11	Comparison of Indexing Techniques	126
Table 6.1	Iteration Method for PageRank	132
Table 6.2	Bookmarks of the Research Paper B	133
Table 6.3	Frequency of Keywords	134
Table 6.4	Cosine Similarity Values of Paper with Citation	134
Table 6.5	Comparison Study of CC, PR and BCC	135
Table 6.6	Final Rank Values	137
Table 6.7	Keywords Attached to each Cluster	138
Table 6.8	Final Result Set against the User's Query	139
Table 6.9	Comparison between CC, PR, C3 and Proposed Approach	139
Table 6.10	Summary of Proposed Ranking Technique	140
Table 7.1	P, R and F values of Proposed Crawler	149
Table 7.2	Query Set fired by Different Users	156
Table 7.3	Resultant Papers for both the Approaches	161

LIST OF FIGURES

Figure		Page No.
Figure 1.1	Major System Components of Digital library	3
Figure 1.2	Chapter-wise Organization of the Dissertation	8
Figure 2.1	Basic Process of Information Retrieval	11
Figure 2.2	Architecture of Web Search Engine	14
Figure 2.3	The Architecture of a Digital Library	19
Figure 2.4	Structure of Crawler	20
Figure 2.5	Structure of Focused Web Crawler	21
Figure 2.6	Architecture of Indexing Process	28
Figure 2.7	An Example of Citation Indexing	34
Figure 2.8	An Example of Keyphrase indexing	36
Figure 3.1	Example of Citation Graph	50
Figure 3.2	Example of a Graph	54
Figure 3.3	Hubs and Authorities	56
Figure 3.4	Algorithm to Determine Base Set	57
Figure 3.5	The K-means Algorithm	72
Figure 3.6	The Hierarchical Agglomerative Clustering (HAC) Algorithm	73
Figure 4.1	Architecture of Proposed Crawler System	88
Figure 4.2	Example to Download pdf of Document	89
Figure 4.3	Example of when Button or Link is given on the Web Page to Download pdf	90
Figure 4.4	Architectural Flow of Categorization System	91
Figure 4.5	Algorithm for Categorization Process	92
Figure 4.6	Schema for Topic taxonomy	95
Figure 4.7	Algorithm for Incremental Keyword Filter Module	96
Figure 4.8	Missing Document Finder Module	98
Figure 4.9	Example of URL Feature	100
Figure 4.10	Example of Name-Match Feature	101
Figure 4.11	Search Result list against Query on Google Search Engine	102
Figure 4.12	Example of Priority Scheduling	103

Figure 4.13	Topic Taxonomy	105
Figure 4.14 (a)	When pdf downloader is not able to download pdf	108
Figure 4.14 (b)	list of search result while hitting the Q2 type query i.e. Quoted Title of Document	108
Figure 4.14 (c)	When pdf downloader downloads the document by getting the link through missing Document Finder Module	108
Figure 5.1	Architecture of Proposed Indexing System	112
Figure 5.2	Algorithm for Clustering the Documents	115
Figure 5.3	Multi-Level Index Structure	118
Figure 5.4	Data Structures for Crawling and Indexing Process	119
Figure 6.1	Workflow of the System	128
Figure 6.2	Format of Search Log	130
Figure 6.3	Citation Graph	131
Figure 6.4	Citation Graph of Papers	133
Figure 7.1	Home Page of Crawling System	142
Figure 7.2	List of Seed Document Title	143
Figure 7.3	Paper Repository with Category Information	143
Figure 7.4	Similarity Values Computed by Link Priority Analyzer	144
Figure 7.5	Extracting Meta-data Information (i.e. Title) of References for finding Missing Document	145
Figure 7.6	Extracting Author's Information of References for finding Missing Document	145
Figure 7.7	Results after Browsing Paper Title Query (i.e. without quotes) from Different Search Engines	146
Figure 7.8	Result Screen of Proposed Crawler for query "Survey of Recent Web Prefetching Techniques"	147
Figure 7.9	Result Screen of CiteSeerx for query "Survey of Recent Web Prefetching Techniques"	147
Figure 7.10	Result Screen of Google Scholar for query "Survey of Recent Web Prefetching Techniques"	148
Figure 7.11	P, R and F Values for each Runs of Proposed Crawler	149
Figure 7.12	Home Screen of Proposed DL Search System	150

Figure 7.13	Search Interface of Proposed DL Search System	150
Figure 7.14	Cosine Similarity Values calculated by Similarity Analyzer	151
Figure 7.15	Fragment of Number of Downloads saved in Log	152
Figure 7.16	Computation of Query Category by Query Analyzer	152
Figure 7.17	Result Screen of proposed System for Query “Survey of Web Page Ranking Algorithm”	153
Figure 7.18	Google Scholar’s Results for Query “Survey of Web Page Ranking Algorithm”	153
Figure 7.19	CiteSeer ^x ’s Results for Query “Survey of Web Page Ranking Algorithm”	154
Figure 7.20	Comparison of Precision Values between CiteSeerx, Google Scholar and Proposed DL System	155
Figure 7.21	Comparison of Precision Values between the Existing Approach and Proposed Approach as per User’s Perceptive	155
Figure 7.22	A Graph showing P, R and F values for Query Set 1 and Query Set 2	156
Figure 7.23	Home Screen of the Proposed Document Categorization System	157
Figure 7.24	Keywords of the Networking Category	158
Figure 7.25	Paper Information in the Categorized Document Database	158
Figure 7.26	Successful Uploading of the Paper	159
Figure 7.27	Interface for Searching the Database	159
Figure 7.28	Resultant List of Papers	160
Figure 7.29	Comparison between Single level and Multi level Approach	162
Figure 7.30	Variation of CC, PR, TDCC and BCC Values	163
Figure 7.31	Comparison of CC, TDCC, PR and BCC Values	164

LIST OF ABBREVIATIONS

Abbreviation		Details or Expanded Form
WWW		World Wide Web
URL		Universal Resource Locators
IR		Information Retrieval
DL		Digital Library
ACI		Autonomous Citation Indexing
SSCI		Social Science Citation Index
SCI		Science Citation Index
AHCI		Art & Humanities Citation Index
ISI		Institute for Scientific Information
SVD		Singular Value Decomposition
LSI		Latent Semantic Indexing
VSM		Vector Space Model
SVM		Support Vector Machines
K-NN		K-Nearest Neighbor
P		Precision
R		Recall
F		F-Measure
HAC		Hierarchical Agglomerative Clustering
RAP		Repository Access Protocol
MCQs		Multiple Choice Questions
HTML		Hypertext Markup Language
XML		Extensible Markup Language,
PR		PageRank
BCC		Bookmark based Ciattion Count
CC		Citation Count
TDCC		Time Dependant Citation Count
C3		Content based Citation Count

PSPR		Popularity and Similarity based PageRank
HITS		Hyperlink Induced Topic Search
PSPR		Popularity and Similarity Based Page Rank
TPM		Transition Probability Matrix
WSNs		Web Social Networks
TF		Term Frequency
IDF		Inverse Document Frequency

Chapter I

INTRODUCTION

1.1 GENERAL

The information age is characterized by a rapid growth and explosion in the amount and heterogeneity of the information available [1]. This had led to an information explosion problem and hence better methods to filter, retrieve and manage this potentially unlimited inflow of information, has become a necessity [2]. Thus, the information age leads to the need to understand and manage an increasing amount of information from distributed information repositories. From the user's viewpoint, there is a demand to effectively and efficiently retrieve this information. This has given birth to the area of Information Retrieval (IR).

Information retrieval (IR) [3, 4] is a field of study dealing with the representation, storage, organization of, and access to documents. The documents may be books, reports, pictures, videos, web pages or multimedia files. The whole purpose of an IR system is to provide a user easy access to documents (usually in unstructured form) containing the desired information. A number of sophisticated tools have been developed to aid the retrieval of information from the internet. The best known example of a web IR system is Google search engine [5].

In spite of recent advances in search engine technologies, these are not completely capturing the vast amount of information available in the digital form i.e. digital documents [6]. For retrieving the more relevant results for users and researchers, digital libraries have been introduced. A Digital Library [7, 8] is an integrated collection of various services including catching, indexing, saving, finding, guarding and extracting digital content or information. It enables the user to easily access huge quantity of available digital information on web. Today, digital libraries are being utilized for various communities and in variety of different fields like academic, science, culture, health, and many more [9]. Thus, the introduction of digital libraries has made the creation, storing, sharing and retrieving of information attractive and easy for the web users.

The amount of digital content in digital libraries is rapidly growing which somewhere degrading the performance of digital library search systems. Therefore, in order to provide the fast and efficient retrieval of digital library search results as per user's query, a lot of approaches have been proposed in this thesis for crawling, indexing, ranking and retrieving relevant results.

1.2 SEARCH ENGINE

A Search Engine [10, 11] is an automated information retrieval system designed to help minimize the time required to search for desired information on the World Wide Web (WWW) [12]. A generic Web search engine [13, 14] comprises of three major components: Crawler, Indexer and Query Processor.

Crawler: It works in the background. The main function of Web crawler is to download the Web documents from WWW to be indexed by indexer.

Indexer: It extracts all words from the downloaded documents and retains them along with the associated document in a local repository called Index.

Query Processor: This component works at the front end. When a user hits a query, then query processor retrieves the relevant information from the index as per user interest. An additional Ranking component may work in association with the Query Processor to order the documents before returning them to the user.

1.3 DIGITAL LIBRARY

A Digital Library (DL) [7, 8] is an integrated set of services that allows capturing, cataloging, storing, searching, protecting, and retrieving the information. It provides coherent organization and convenient access to typically large amounts of digital information. Nowadays, Digital libraries are experiencing rapid growth with respect to both the amount and richness of becoming available. Modern search engine technologies [15] are now being introduced in digital libraries to retrieve the relevant content. Digital Libraries are being created day by day for diverse communities and in different fields e.g. education, science, culture, development, health, governance and so on. Digital libraries differ significantly from the traditional libraries because they allow users to gain an on-

line access to and work with the electronic versions of full text documents, research papers and their associated images. Many digital libraries also provide an access to other multi-media content like audio and video.

Components of Digital Library: Digital library framework permits many different computer systems to coexist. The key components are shown in the Fig. 1.1. They run on a variety of computer systems connected by a computer network, such as the Internet. Various components [16, 17] are described as given below:

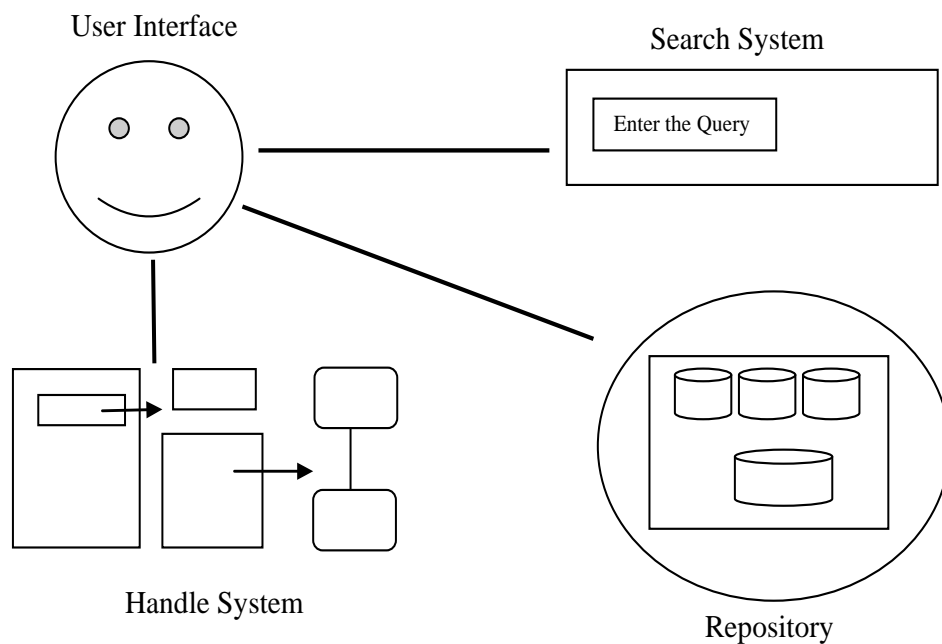


Fig. 1.1 Major System Components of Digital library

User Interface: The goal of a good user interface is to establish a connection between user/patron and the machine which provide valuable information. A digital library must provide a single point of access like portal to a huge quantity of digitized information that is available to a diversity of kind patrons with a different psychological, academic, social backgrounds and information needs over Internet. Digital libraries have two types of user interfaces: one for the end-users of the digital library, the other for digital librarians and system administrators who manage the collections.

Repository: Repository refers to a storage location and often for preservation. In digital library, repository stores digital contents, its metadata and other information. The interface to this repository is called the Repository Access Protocol (RAP) [18].

Handle System: Handles are general-purpose identifiers that can be used to identify Internet resources, such as digital objects [19], over long periods of time and to manage materials stored in any repository or database.

Search System: It is a software system that is designed to search for information on the WWW. The search system results are generally presented in the line of results, which may be a mix of contents. The design of the digital library system assumes that there will be many indexes and catalogs that can be searched to discover information before retrieving it from a repository. These indexes may be independently managed and support a wide range of protocols.

1.4 MOTIVATION

The following issues in the existing literature directed the research towards designing a framework for a separate Digital Library Search System:-

- **Huge size of Web:** The first challenge is related to the vast amount of available digital documents on WWW. But, no single digital library search engine has crawled and indexed the entire Web. The factor that decreases digital library search engine efficiency is the missing information or paid access of some documents. If the crawler can crawl or harvest the Web by using metadata information of such documents, then user can get all the relevant and desired documents through digital library search engines.
- **Lack of efficient data structures and Indexing Process:** The existing data structures employed in the indexing process of digital library search engines lack valuable information related to relevant document retrieval. The existing index structures maintained by majority of digital library systems are keyword based. Therefore, a large number of irrelevant documents are returned posing the problem of *Information Overkill*. This problem can be resolved by indexing the

documents using multi-level index structure which provides better efficiency and effectiveness of digital library search engines.

- ***Irrelevant Search results:*** Digital library search engines generally return a list of results in response to user queries. Typically, those documents are returned whose contents match to some extent with the submitted query. But in this scheme, it becomes hard for them to find the documents they are looking for. If the documents are retrieved based on the category and content of the query, then this problem can be solved.
- ***Inefficient Ranking of Documents:*** The relevancy of a document can be determined based on ranking algorithms and majority of these algorithms are based on content or link analysis. Nevertheless, in many situations, traditional methods are not the perfect solution to determine the relevancy of a document. Therefore, the problem of ranking digital library search results becomes inherently complex. Ranking the documents based on their content and link structure can result in retrieving more relevant results at the top of result list.

1.5 PROBLEM DEFINITION

A critical look at the mentioned issues indicates the need to design a unified framework for online digital library search system which resolves the above mentioned problems. There is a need of focused crawler which gathers all the documents present on WWW and harvests the documents from author's homepages as well. In order to organize these documents, an index structure is needed which provides the fast retrieval of document based on their domain/ topic or category instead of keyword based search. Further, to represent these returned result lists, a novel method to rank the results in form of cluster needs to be proposed which provides the relevant results at the top of the result list.

1.6 OBJECTIVES OF RESEARCH WORK

Today, the main challenge in front of digital library search engines is the retrieval of relevant and quality documents in correspondence to user information needs, but a number of limitations as identified in the previous sections render the relevant

information retrieval a complicated task. To resolve these issues toward building cost effective and efficient digital library search systems, the following objectives were set:

- **Design of a Unified Approach:** Several researches are available in the literature that resolve only one or few related issues, but not any unified technique has been reported that simultaneously can resolve most of these issues.

Proposal: *In this thesis, a unified framework of digital library search system has been proposed, which can optimize multiple processes such as crawling, indexing, ranking and query processing of digital library search engines.*

- **Efficient Crawling of Missing Information:** The existing data structures employed in the crawling process of search engines lack valuable information related to relevant document retrieval. Moreover, the current digital library crawlers crawl up to a specific depth on the web owing to which many important publications (e.g. paid publications) may remain unvisited.

Proposal: *Novel data structures have been designed that provide better efficiency and effectiveness of digital library search engines. An efficient technique to utilize the meta-data in the crawling for finding the missing or paid publications has been proposed.*

- **Categorization of Documents for better Organisation:** When a researcher or user submits a query, then the existing systems compare the query terms with the documents. If some match occurs, then the search results are displayed to user. But if user has not defined the topic or domain of the search, then the list of search results are irrelevant to the user. No work has been performed to retrieve and display the publications based on category of the query.

Proposal: *In this work, in order to retrieve more relevant results as per user query, Document categorization is considered. A categorized multi-level database structure is taken in the form of hierarchy. The category of publication is extracted by matching the keywords of publications with the keywords of category. When a user hits the query, first the system checks the category of the query and then displays the results under that category.*

- **Efficient Retrieval of Relevant Results:** Most of the search results returned by the search systems are ranked generally based on content –oriented or link

oriented approaches. Thus, sometimes the ranking based on these concepts does not reflect relevancy and displays the irrelevant search results to the user.

Proposal: *In this work, content and link structure of a publication is taken into account for ranking the search results. Instead of considering the total number of citations (i.e. incoming links), the proposed method computes the relevancy between the publications and their citations by matching their bookmarks and ranking the results accordingly.*

- **Efficient Result Representation Schemes:** In response to user queries, a digital library search engine generally returns a large number of results presented to the user in the form of a ranked list. To search for the desired information, user keeps on navigating between the papers and thus making extra efforts. Some more efficient representation scheme is needed to reduce the search space.

Proposal: *A more efficient way of organizing the publications can be a combination of clustering and ranking, where clustering can group the publications and ranking can be applied for ordering the publications within each cluster. Based on this approach, a mechanism based on link structure of publications and query similarity has been proposed. It provides ordered results in the form of clusters in accordance with user's query.*

1.7 ORGANIZATION OF THESIS

The chapter wise organization of the dissertation is shown in Fig. 1.2 and a brief outline of the remainder of this dissertation is given as:

Chapter II: Information Retrieval and Digital Library System: A Review: This chapter reviews the technology behind general digital library search engines and presents in detail the prevalent crawling and indexing techniques in use by current digital library search engines. At the end, the chapter enumerates various issues which must be considered in the design of effective and scalable digital library search engines.

Chapter III: State-of-The-Art Techniques in Digital Libraries: This chapter presents in detail the prevalent document ranking algorithms in use by current digital library search engines. This chapter also describes current state-of-the-art techniques such as

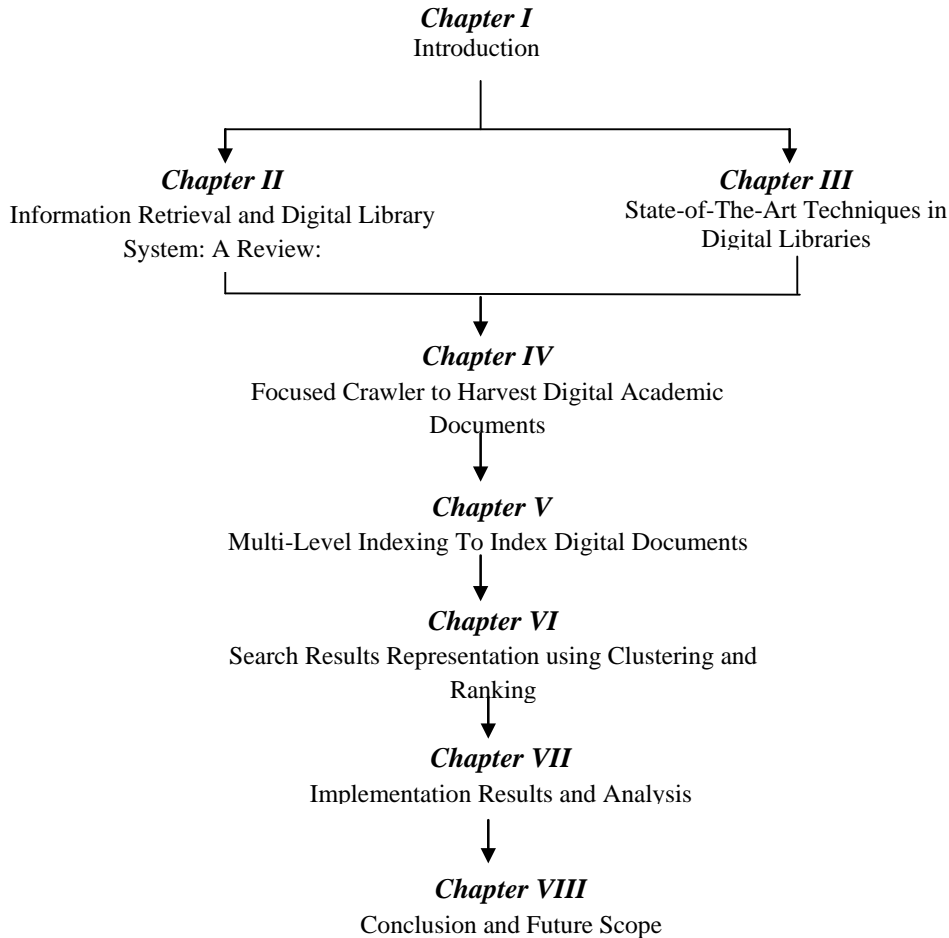


Fig. 1.2 Chapter-wise Organization of the Dissertation

Cluster Analysis, Document Categorization and Keyword Extraction in detail. Several issues regarding information retrieval through digital library search engines have also been identified.

Chapter IV: Focused Crawler to Harvest Digital Academic Documents: This chapter proposes a novel approach to crawl digital library search system. A set of data structures for crawling have also been proposed. This chapter also describes a Document Categorization technique in detail for efficiently retrieving the relevant results.

Chapter V: Multi-Level Indexing to Index Documents: This chapter proposes an indexing scheme for efficient digital library system. Its two main modules: pre-processing module and query processing module are discussed in detail along with

algorithm used. A set of data structures for indexers have also been proposed in this chapter.

Chapter VI: Search Results Representation using Clustering and Ranking: This chapter describes the proposed clustering and ranking mechanism for efficiently retrieving the documents from the digital library search system as per the user interest.

Chapter VII: Implementation and Result Analysis: This chapter discusses the implementation aspects of proposed techniques. This chapter also includes the snapshots and results of experiments. Performance of various techniques has been measured in terms of precision, recall and F-measure. The results so obtained have been compared with the outputs of existing systems.

Chapter VIII: Conclusion and Future Scope: This chapter concludes the work and provides a description of potential future work in the area under consideration.

The digital library search engine's technology and a comprehensive review of some prevalent state-of-the-art techniques employed by existing digital library search engines is presented in next two chapters.

Chapter II

INFORMATION RETRIEVAL & DIGITAL LIBRARY SYSTEMS: A REVIEW

2.1 INFORMATION RETRIEVAL

Information retrieval [3, 4] is fast becoming the dominant form of information access. It can be defined as:

Information Retrieval (IR) is the task of finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

In an abstract sense, IR deals with the representation, storage, organization of, and access to information items. The representation and organization of the information items should provide the user an easy access to the information in which he is interested. The general IR process is depicted in Fig. 2.1, wherein *Index* provides an efficient representation of the information items stored in the database. The *Query Engine* is responsible for taking user queries, retrieving the matched results/records from the index and representing them to the user in an understandable manner.

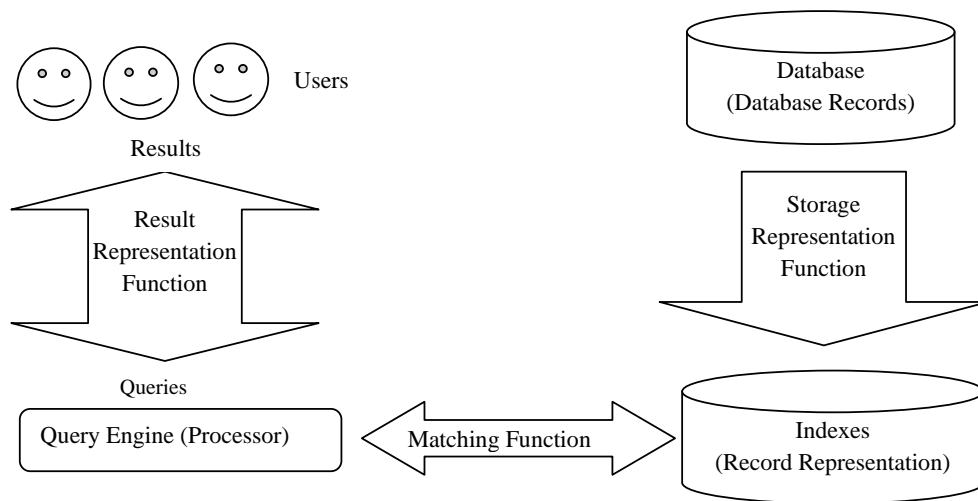


Fig. 2.1 Basic Process of Information Retrieval

Earlier, IR used to be an activity that only a few people like reference librarians and similar professional searchers were engaged in. But, in current scenario, hundreds of millions of people are engaged in this process in their daily routines, for example when they use a web search engine to access the Web documents on WWW or search their emails [22].

WWW [12] is an interlinked collection of documents. These documents contain hyperlinks to other documents. The links can point to a document on the same machine or to one on the other side of the world. It is becoming a challenging task to find the specific information in the WWW or Web, because of the rapid growth of the Web and the diversity of the information offered through the Web. Therefore, the field of information retrieval covers a broad spectrum of techniques and applications that aim to satisfy the user's information needs. An ideal information retrieval system [23] must be able to

- Determine the information needs of a user,
- Search the information available,
- Return the relevant information that is generally compiled from multiple sources, in a language and format that can be easily understood by the users.

In IR, the information need of the user is expressed as a bag of keywords [24]. The results are returned in the form of a list of documents that contain one or more of those keywords. The user accesses the information from the web through the search engine, which is discussed in the next section.

2.2 SEARCH ENGINES

The plenteous content available on the WWW is useful to millions of people World Wide. Some simply browse the Web through entry points such as *Yahoo*, *MSN* etc, but many information seekers use a Web search engine to begin their Web activities. A *Search Engine* [14, 25] is an automated information retrieval system designed to help minimize the time required to search for desired information on the WWW. The search results are generally presented in an ordered list or in groups, and are often called hits. The results may consist of web pages, images, information, blogs and other types of files. A list of some prevalent search engines is given in Table 2.1.

Table 2.1: List of some Popular Search Engines

<i>Year</i>	<i>Engine</i>	<i>Current Status</i>
1983	AOL[26]	Active, Web portal and online services
1994	Web Crawler[27]	Active, Aggregator
1995	AltaVista [28]	Defunct, Domain has redirected to Yahoo!'s own search site
1998	Google [5]	Active
2004	Yahoo! Search [28]	Active, Launched own web search (see Yahoo! Directory, 1995)
2005	GoodSearch [29]	Active
2008	DuckDuckGo[30]	Active, Protecting searchers' privacy and avoiding the filter bubble of personalized search resultP
2009	Bing [31]	Active, Launched as rebranded Live Search
2011	YaCy [32]	Active, P2P web search engine
2015	Cliqz [33]	Active, Browser integrated search engine

Following subsections describe the general architecture of search engines and basic terminologies used by them.

2.2.1 GENERAL ARCHITECTURE OE A SEARCH ENGINE

The architecture of a typical search engine [13, 14] is shown in Fig.2.2. The most important component of the search engine is a crawler [34] also called a robot or spider that traverses the hypertext structure in the web, downloads the web pages, and stores them in page repository. The downloaded pages are then routed to an indexing module [35] that parses them and builds the index based upon the keywords present within the pages. Index is generally maintained alphabetically considering the keywords. When a user fires a query in the form of keywords on the interface of a search engine, the query processor after matching the query keywords with the index returns the URLs of the

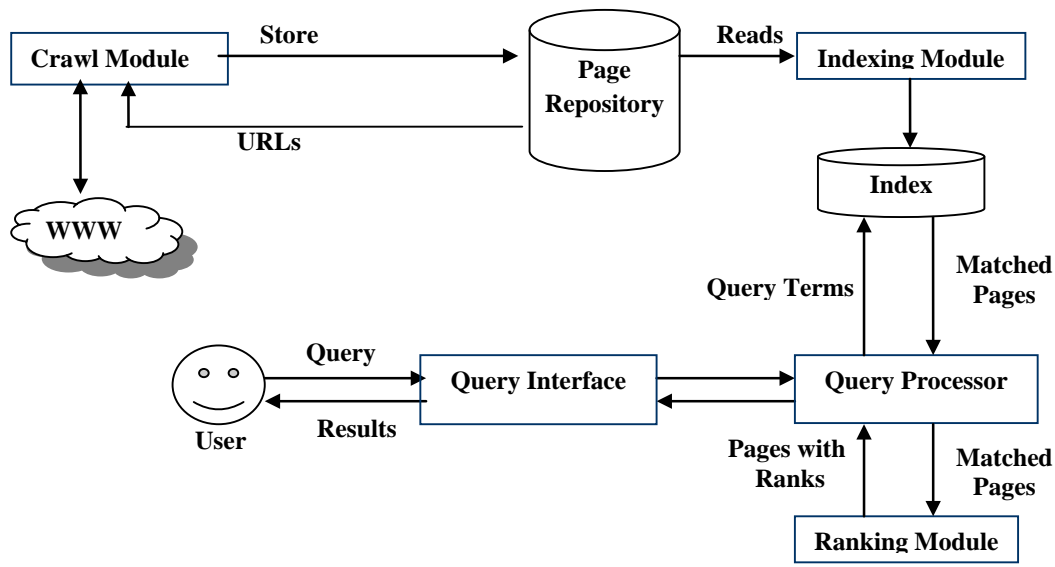


Fig 2.2 Architecture of Web Search Engine

pages to the user. A search engine generally returns a large number of web pages in response to user queries and users have to spend much time in finding their desired information from this long list resulting in information overload problem.

Before representing the results to the user, some ranking mechanism [36] either in back end or in front end is used by most of the search engines to make the user search navigation easier within the search results. Important pages are displayed on the top of results leaving the less important pages downwards.

2.2.2 TYPE OF ISSUES WITH THE CURRENT WEB

Data on the Web is not in the form as desired by the users. It contains a lot of issues that should be looked upon by the search engines. These are described as under.

- **Large volume:** WWW contains huge collection of data. Also, the growth of data over the WWW is exponential. Increase in the amount, poses scaling issues that are difficult to cope with.
- **Distributed data:** Data is distributed widely over the WWW. It is located at

different sites and platforms. The communication links between computers vary widely.

- **Unstructured and redundant data:** The data on the Web is highly unstructured [37]. It is impossible to organize and add consistency to the data and the hyperlinks. Also, there exists semantic redundancy that can increase traffic.
- **High percentage of volatile data:** The data on the Web is highly volatile. Documents can be added, removed or updated easily on the WWW. These changes to the documents are usually unnoticed by users.
- **Quality of data:** The data available on the Web is not of high quality. A lot of Web pages do not involve any editorial process. That means data can be false, inaccurate, outdated, or poorly written.
- **Heterogeneous data:** Data on the Web is heterogeneous. Documents are written in different formats, media types, and natural languages.
- **Dynamic data:** The content of Web documents change dynamically [12] with some web pages being highly dynamic and some less. The web pages that changes dynamically need to be noticed, so that the user gets an updated page on visit.

Web is massive, much less coherent; it changes more rapidly, and is spread over geographically distributed computers. This requires new information retrieval techniques, or extensions to the old ones, to deal with the gathering of the information, to make index structures scalable and efficiently updateable, and to improve the discriminating ability of search engines.

In spite of recent advances in search engine technologies, there still occur situations where the user is presented with non-relevant search results. For example, when a user inputs a query for some scientific literature, book or periodical on general purpose search engine such as Google [5], it returns a long list of search results consisting of tutorials, news, articles, blogs etc. This is because these search engines are not completely capturing the vast amount of information available in the digitization projects on books and periodicals that are occurring locally, nationally and internationally.

Now days, researchers are making their work available online in the form of postscript or PDF documents, therefore, amount of scientific information and the number of electronic journals on the Web is increasing at a fast rate. But the access to the growing body of scientific literature on the publicly indexable Web is limited by the lack of organization of such information [11]. To overcome this problem, digital libraries [7, 8] have been introduced to make retrieval mechanism more effective and relevant for researchers or users. As general purpose search engines do not take into consideration the vital information available in digital repositories/ libraries, there is a need to design a separate retrieval system over the online digital libraries that satisfies user's information needs and returns only relevant results.

2.3 DIGITAL LIBRARIES: AN INTRODUCTION

Libraries have always strived to collect, process and disseminate information. But information today exists in many forms than just a printed matter and this has lead to the evolution of Digital Libraries. A Digital Library [7, 8, 15] is an integrated set of services for capturing, cataloging, storing, searching, protecting, and retrieving information, which provides coherent organization and convenient access to typically large amounts of digital information. Digital libraries break the barrier of physical boundaries and strive to give access to information across varied domains and communities. Digital libraries are experiencing rapid growth with respect to both the amount and richness of available digital content. As a consequence of the huge amount of digital content becoming available, modern search engine technologies are now being introduced in digital libraries to retrieve the relevant content [38]. The list of some existing digital libraries is given in Table 2.2.

2.3.1 Benefits of Digital Libraries

Digital libraries bring significant benefits [47, 48] to the users through the following features:

- ***No Physical Boundaries:*** The users can access the digital libraries virtually at any time and from anywhere without having to wait and going to physically

Table 2.2 List of some Existing Digital Libraries

<i>Name</i>	<i>Discipline</i>	<i>Access Cost</i>
<i>CiteSeer [39]</i>	<i>Computer Science</i>	<i>Free</i>
<i>Google Scholar [40]</i>	<i>Multidisciplinary</i>	<i>Free</i>
<i>IEEEExplore [41]</i>	<i>Computer Science Engineering, Electronics</i>	<i>Subscription</i>
<i>ScienceDirect [42]</i>	<i>Multidisciplinary</i>	<i>Subscription</i>
<i>Open Access Journals Search Engine(OAJSE) [43]</i>	<i>Multidisciplinary</i>	<i>Free</i>
<i>Academic Publications eJournal [44]</i>	<i>Multidisciplinary science</i>	<i>Free</i>
<i>Academic Search [45]</i>	<i>Multidisciplinary</i>	<i>Subscription</i>
<i>SpringerLink [46]</i>	<i>Multidisciplinary</i>	<i>Free abstract & preview; Subscription full-text</i>

anywhere.

- ***Wider access:*** The same resources or documents can be used simultaneously by a number of users at a same time. It can also meet the requirements of a larger population of users easily.
- ***Improved information sharing:*** Through the appropriate metadata and information exchange protocols, the digital libraries can easily share information with other similar digital libraries and provide enhanced access to users.
- ***Improved preservation:*** Since the electronic documents are not prone to physical wear and tear, their exact copies can easily be made, thus the digital libraries facilitate preservation of special/ rare documents and artifacts by providing access to digital versions of these entities.
- ***Information Retrieval:*** Digital Libraries can provide very user friendly interface to users for searching required documents by clicking search terms.
- ***Structured Approach:*** Digital libraries provide access to much richer content in a more structured manner, i.e. we can easily move from the catalog to the particular

book, then to a particular chapter and so on.

2.3.2 Principles for Digital Library Design

The main objective of a digital library is to provide coherent organisation and convenient access to typically large amounts of digital information. The following principles [8, 49] guide the development of the architecture of digital library system:

- **Service Driven:** The architecture for the DLs must be driven by the services it provides and tools required for delivering the service.
- **Open Architecture:** The architecture must be open, extensible and support interoperability among heterogeneous, distributed systems.
- **Scalability:** The architecture must be robust, scalable and reliable in a high transaction rate production setting thousands of patrons with a wide variety of backgrounds and information needs.
- **Preservation:** The architecture must ensure persistent access to the collection of DL, addressing issues such as naming, digital archiving and digital preservation.
- **Privacy:** The architecture must be sensitive to privacy issues and support both anonymous and customized access to resources.
- **Practicality:** The architecture should represent a flexible and practical approach to standards, recognizing the need to balance the level of information collection with economic constraints.

The general architecture of a digital library system is described in the next section.

2.4 GENERAL ARCHITECTURE OF DIGITAL LIBRARY SEARCH SYSTEM

Fig 2.3 depicts the general architecture of a digital library search engine [15]. The main functions carried out by the system are described as:

Document Acquisition: When the user wishes to explore a new topic, a new instance of the agent is created for that particular topic. The user invokes this sub-agent by giving it broad keywords. The sub-agent uses search engines and heuristics for searching the Web

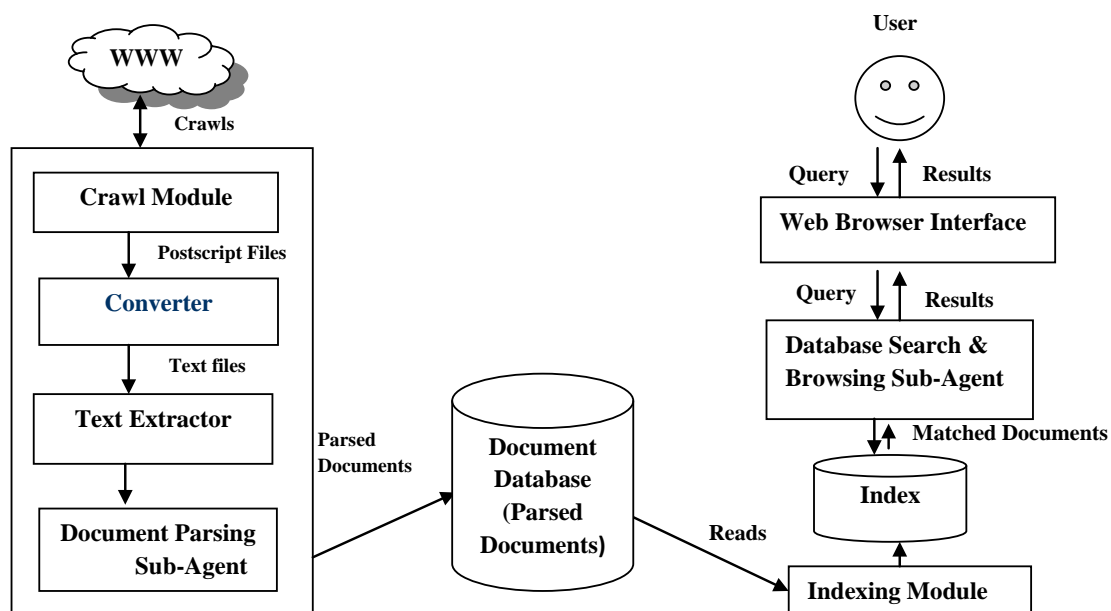


Fig. 2.3 The Architecture of a Digital Library

pages which are likely to contain links to research papers of interest. The agent locates and downloads postscript files identified by “.ps”, “.ps.Z”, or “.ps.gz” extensions.

Document-Parsing: This module extracts the semantic features and citations from the downloaded documents and places them in a database. Each citation is parsed using heuristics to extract the fields like document text, document words and further citations etc.

Database Browsing: This component consists of a query processing sub-agent which takes a user query of proper syntax and returns an HTML formatted response. Typically, the query program is not used directly, but through a Web browser interface.

The detailed description of each component of digital library search engine is described in following sections.

2.5 CRAWLER

Web Crawlers [50, 51] are one of the main components of digital library search engines. Web crawling is the process by which system gather a corpus of digital documents from

the Web resources, in order to index them and support a digital library search engine that serves the user queries. The primary objective of crawling is to gather as many useful documents as possible quickly, effectively and efficiently, together with the link structure that interconnects them.

A general structure of crawler [50, 51] is shown in Fig. 2.4. Crawling starts with a set of seed URLs stored in a queue structure, called “URL queue”. Then multiple threads of crawler execute simultaneously and each thread gets the URL from the queue which further fetches the corresponding web pages from the server. Later, this page is parsed to extract links/ URLs and these links are appended to the URL queue to be fetched later. A real life crawler is much more complex than this structure to consider issues like politeness policy also i.e. do not request many web pages from the same server at the same time.

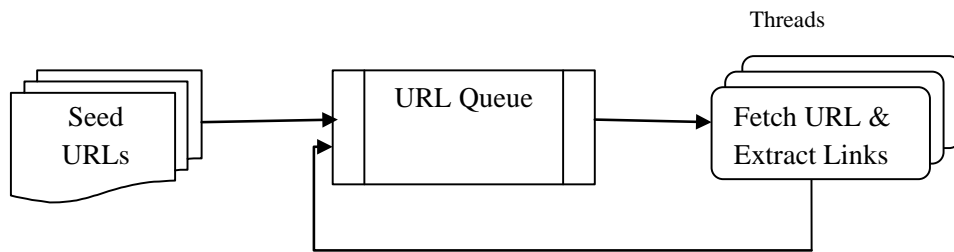


Fig 2.4 Structure of Crawler

2.5.1 Types of Web Crawler

Different strategies are being employed in web crawling. These are as follows.

a) Simple Crawler: It is a single –process information crawler, which was initially made for his desktop. Later, it was extended on to the internet. It had a simple structure, and hence had limitations of being slow and having limited efficiency.

b) Focused Crawler: A Focused Crawler [52, 53] finds, acquires, indexes, and organizes the documents based on specific topics. The focused crawlers are designed to efficiently extract documents based on different parameters which

identify or check the relevancy of extracted documents as per the user interest; priority criteria for deciding in which order to pursue based on previously downloaded information and save all information and extracted documents into the local database. There are various applications of the focused crawler including generating web based recommendations and retrieving domain/topic relevant scientific paper/publication etc. They are also useful to update topic relevant indexes where specific information is required to fulfill the community’s information need, in comparatively much lesser time [54].

Fig. 2.5 represents the structure of the focused web crawler. The only difference compared to the general crawler is the *Topic Classifier* which makes it more precise [55]. Each fetched page is classified to predefined target topic(s). If the page is predicted to be on-topic, then its links are extracted and are appended into the *URL Queue*. This type of focused web crawler is called “full-page” focused web crawler since it classifies the full page content. In other words, the context of all the links on the page is the full page contents itself.

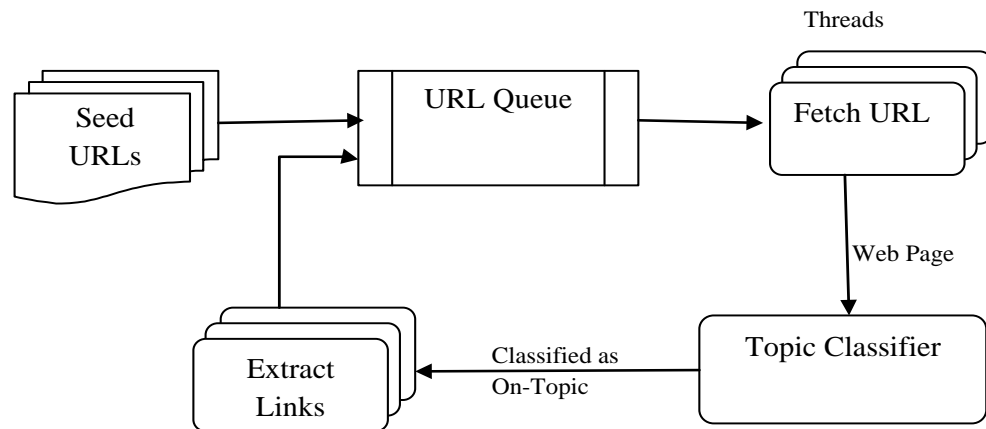


Fig 2.5 Structure of Focused Web Crawler

c) Distributed Crawler: Distributed web crawling [56] is a distributed computing technique whereby Internet search engines employ many computers to index the Internet via web craw. It reduces the overload on server by spreading the load of tasks on different computers. The main focus is on distributing the

computational resources and the bandwidth to the different computers and the networks.

d) Parallel Crawler: Initially given by Junghoo Cho in 2002 [57], this approach relied on parallelizing the crawling process, which was done by multi-threading. A parallel crawler [58, 59] consists of multiple crawling processes. Each process performs the basic tasks that a single-process crawler conducts. It downloads web pages from WWW, stores the pages locally, extracts URLs from the downloaded pages and follows links. Some of the extracted links may be sent to other processes depending on how the processes split the download task. The processes may be distributed either on the same local network or at geographically distant locations. This type of crawler has faster downloading, less time consuming but requires more bandwidth and computational power for parallel processing.

e) Incremental Crawler: A traditional crawler, in order to refresh its collection, periodically replaces the old documents with the newly downloaded documents. On the contrary, an incremental crawler [60, 61] incrementally refreshes the existing collection of pages by visiting them frequently based upon the estimate as to how often pages change. It also exchanges less important pages by new and more important pages. It resolves the problem of the freshness of the pages. The benefit of incremental crawler [62] is that only the valuable data is provided to the user, thus network bandwidth is saved and data enrichment is achieved.

2.5.2 Study of Existing Web Crawlers for Digital Libraries

A literature analysis of various web crawlers for digital library search engines has been done in this section. A few existing crawling techniques used for digital library search engines are discussed below:

a) Locating Online Copy of Scientific Documents: This system [63] makes use of citation information to locate and crawl copies of articles available throughout the Web. The heuristic-based crawling and distance-based title matching algorithms are used to find online copies of scientific papers more effectively. This system solves the problems of involving human browsing in order to get the

final online copy, and incomplete coverage. But, the main disadvantage of this system is that it can find target documents more efficiently than Google, but it does so at the cost of time i.e. the elapsed time per citation.

b) Finding Scientific Papers with HomePageSearch and MOPS: An approach [64] to seek scientific papers relevant to a pre-defined research area was proposed in this system. It searches for web pages which are created by scientists and are active in the research area under consideration. A list of names of scientists is obtained from electronic computer science bibliographies. The HomePageSearch system finds the Home Pages according to the names, and Mops uses the homepages as starting URLs and finds research papers close to the Home Pages. It creates an index of these papers and makes it accessible on the web. The quality of the MOPS index depends on the list of starting points for the search. If the starting URLs are too far away from the documents, the system either will not found them, or the search takes too much time.

c) Missing Content Analysis: In this [65], popular information needs are identified by proposing a tool which dynamically analyze the query log of the system, identify missing content queries, and then direct the system to enrich its data. Thus, this tool is able to satisfy the user's needs. First, system finds topics or information needs that have low coverage within the system and then reduces the knowledge gaps by using an alternative sources. This system uses the query logs to represent knowledge requirements set by users in the past; thus, the process may improve the quality of recurrent queries that were once identified as Multiple Choice Questions (MCQs) but unable to predict future knowledge demands.

d) A Meta-Search Enhanced Focused Crawling: In this [66], a set of seed URLs is taken as an input by the crawler for fetching relevant pages based on the content and link-based analysis results. If the fetched page is relevant, then all the outgoing links in the fetched page are extracted and forwarded to the URL queue for further crawling. At the mean time, a meta-searching component keeps drawing queries from a domain-specific lexicon, retrieving diverse and relevant URLs by querying multiple search engines, and combining their top results. The

main advantage of this system is that it requires only domain specific seed URLs. But, with the rapid growth of Web, the effect of increasing the number of starting URLs could be very limited.

e) An Academic Document Search Engine: Whitelists and Blacklists: This system replaces blacklist with a whitelist [67]. A *blacklist* means the crawl is forbidden from a certain list of URLs whereas a *whitelist* means only certain domains are considered and others are not crawled. The whitelist is generated based on domain ranking scores URLs harvested by the CiteSeerX crawler. In this system, whitelist policy includes two essential factors: a ranked seed list, and a domain constrained crawling rule. The main advantage of this system is to use of whitelist which significantly reduces the number of useless URL requests and unnecessary downloads. The system results in increasing the fraction of useful documents. But, while this policy reduces crawling irrelevant URLs, it could miss opportunities to discover new resources as well.

f) Focused Crawling for Educational Materials: The system [68] proposed domain ontology concepts based query method for searching educational documents from Web and categorizing them by topic. It has also proposed concept and term based ranking system for obtaining the ranked seed documents which is then used by a concept-focused crawling system. This system first ranks the seed documents before start crawling for effective results. It also relays on background knowledge of concepts and associated topic learning terms, which are compared with the contents of the crawled documents. The disadvantage of this system is that it does not evaluate the retrieved documents from the point of view of structure of learning content.

g) Automatically Acquiring Scientific Documents: In this system, publicly-available research paper titles and author names are used as queries [69, 70, 71]. Research papers and sources of research papers are identified from the search results using accurate classification modules. This proposed framework crucially depends on accurate paper classification (into categories such as book, paper, thesis etc.) and researcher homepage identification modules. This system uses

“Web Search” to obtain seed URLs for initiating crawls in an open-access digital library. The disadvantage of this system is that trained naïve Bayes classifiers are used for training data which are not easy to use for naïve users.

h) Focused Crawling for Missing Documents: The proposed system [72] uses the publication metadata to guide the crawler towards authors’ homepages to harvest what is missing from a digital library collection. The system first identifies the missing papers that are not indexed by CiteSeer and then proposed a fully automatic heuristic-based system that has the capability of locating authors’ homepages. These author homepages are used further for focused crawling to download the desired papers.

A brief comparison of the various crawlers used in digital libraries described above is given in the next section.

2.5.3. Comparison Study of Various Web Crawlers

By going through the literature survey, a comparison study of various existing web crawlers in digital libraries was done and is shown in Table 2.3 and Table 2.4. The comparison is done based on different parameters such as techniques used, input parameters, types of crawling used, importance and limitations.

A critical look at the available literature indicates the following issues which need to be addressed while designing an efficient crawler for digital library search engines:

- There is a need of a new approach to seek scientific papers relevant to a pre-defined research area. As traditional digital libraries, search for documents based on content similarity only with the query keywords irrespective of their topic/domain/context which results in the irrelevant list of search results.
- Most of the focused crawlers use local search algorithms to gather or build the domain-specific collections that are not comprehensive and diverse enough to scientists and researchers.
- The vast amount of digital documents is available on WWW, but, no single search engine has crawled and indexed the entire Web. The factor that decreases digital

library search engine efficiency is the missing information or paid access of some documents. If the crawler can crawl or harvest the Web by using metadata information of such documents, then user can get most of the relevant and desired documents through digital library search engines.

Table 2.3 Comparison of various Web Crawler for Digital Libraries

Techniques →	An Academic Document Search Engine: Whitelists and Blacklists [67]	Focused Crawling for Educational Materials [68]	Homepage Search and MOPS [64]
Measures ↓			
Description	This system replaces blacklist with a whitelist. Whitelist policy includes two essential factors: a ranked seed list, and a domain constrained crawling rule.	The system proposed domain ontology concepts based query method for searching educational documents from web and categorized by topic. It has also proposed concept and term based ranking system.	An approach to seek scientific papers relevant to a pre-defined research area. This system searches for web pages which are created by scientists who are active in the research area under consideration.
Input Parameters	Seed URLs list, whitelist	Domain-Ontology concepts which are given as queries to search engine.	A list of names of scientists who are active in the research area under consideration.
Need of the User's Support	No Need	No	User interface for manually send correct or wrong personal homepage data.
Type of Crawling Used	Focused Crawling	Concept-Focused Crawling	Focused crawling with topic oriented knowledge
Importance	Use of whitelist significantly reduces the number of useless URL requests and unnecessary downloads. Increases the fraction of useful documents.	This system firstly ranks the seed documents before start crawling for effective results. The system also rely on background knowledge of concepts and associated topic learning terms, which are compared with the contents of the crawled documents.	It searches the given address at the depth of 1 or 2 for finding scientific papers.
Limitation(s)	While this policy reduces crawling irrelevant URLs, it could miss opportunities to discover new resources as well.	It does not evaluate the retrieved documents from the point of view of structure of learning content.	If the starting URLs are too far away from the documents, then the search takes too much time.

Table 2.4 Comparison of various Web Crawler for Digital Libraries

Techniques →	Locating Online Copy of Scientific Documents [63]	Automatically Acquiring Scientific Documents [69, 70, 71]	Missing Content Analysis [65]	A Meta-Search Enhanced Focused Crawling [66]
Measures ↓				
Description	The heuristic-based crawling and distance-based title matching algorithms are used along with citation information in order to find online copies of scientific papers more effectively.	This framework crucially depends on accurate paper classification and researcher homepage identification modules.	Popular information needs are identified by dynamically analyzing the query log of the system, identify missing content queries, and then direct the system to enrich its data	In this system, a meta-searching component keeps drawing queries from a domain-specific lexicon, retrieving diverse and relevant URLs by querying multiple search engines, and combining their top results.
Input Parameters	Citation information of the document.	A publicly-available research paper titles and author names are used as queries to a Web search engine.	Dynamically analyze the query log of the system and alternative external sources to reduce the knowledge gaps.	A set of starting URLs
Need of the User's Support	No need of human browsing	No need	User's support is required to manually build the taxonomies.	No Need
Type of Crawling Used	Focused Crawling	Focused Crawling	User driven Focused Crawler	Meta-Search based Focused Crawling
Importance	Solve the problems of involving human browsing to get to the final online copy, and incomplete coverage.	The system uses "Web Search" to obtain seed URLs for initiating crawls in an open-access digital library.	Increase the probability to find the proper answer for future queries by reducing the knowledge gaps.	Advantage over the Tunneling technique.
Limitation(s)	Needs to spend time on additional crawling and citation matching.	An incorrectly predicted homepage as a seed URL may result in crawling irrelevant documents and extra processing load.	There is no guarantee that the external sources can satisfy the user's specific needs.	More starting URLs one uses in the crawling process, the more comprehensive the final collection will be.

A brief discussion about the indexing process in digital library search engines is described in next section.

2.6 INDEXER

With the huge corpus of digital information present on the WWW, the need to efficiently find some specific piece of digital information as per user interest becomes crucial [73]. In digital libraries, the index structure [74] has been considered as the important component for supporting fast searching. Indexing is an assistive technology mechanism which helps to optimize the speed of digital library search engine in finding the relevant documents against the user query. Indices are used to provide a framework for researchers to locate the documents quickly and efficiently.

2.6.1 Architecture of Indexing Process

The architecture of a typical indexing system [75, 76] is shown in Fig.2.6. The main component of this system is a *Text Acquisition* that identifies and acquires the documents for indexing. This component is responsible to feed the real time streams of documents (e.g. articles, research papers, blogs, videos etc.) and convert the variety of documents into consistent text plus meta-data format. For example, if some documents are in HTML, Word, XML or in PDF format, then this component converts all the documents types into

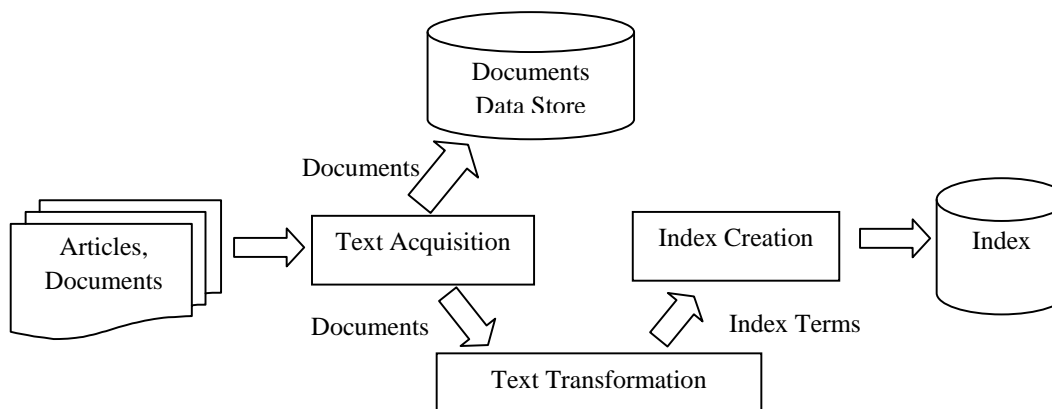


Fig 2.6 Architecture of Indexing Process

XML format. Metadata is the information about documents such as document type, title, author and creation date. This identified and crawled documents are forwarded to be saved in a *Data Store*. This component stores the details of the document in the form of document type, structure, features, size etc. *Text Transformation* component takes the crawled documents as an input and transforms documents into index terms or tokens. It processes the sequence of text tokens in document to recognize structural elements of the documents. The index terms or tokens are further forwarded to *Index Creation* component which is responsible to create the data structure or index in order to support the fast searching of information.

2.6.2 Types of Indexing Techniques

Today, the main challenge for digital library search engines is to efficiently crawl or harvest the scientific literature present on the WWW and index them in an ordered way for efficient retrieving and presenting relevant results to the researcher. Some of the common indexing techniques have been discussed here as follows.

a) Inverted Indexing: Inverted Index is the most commonly used data structure to index documents into the database. It is also named as postings file or inverted file [74, 77]. It is called “inverted index” because it maps the each word of document with its locations in a document or set of documents. There are several variable features or field on inverted indexes that the users can use as per their need. These variations are as:

- *List 1:* A term’s inverted list only stores list of documents in which the word appears in.
- *List 2:* A term’s inverted list stores the list of documents and frequency of occurrence in the documents which it appears.
- *List 3:* A term’s inverted list stores the list of documents and the location (or word positions) of each occurrence of the term in the document in which it appears.

In this case, the search system scans each word of every document that is crawled and creates an inverted index. When the end-user hits the query by using some keywords, then the system fetches the inverted list of terms that match with the query terms.

Illustrative Example: Consider small fragments of two sample documents d_1 and d_2 as given below:

d_1 : An apple a day keeps the doctor away. Apple is red in color. Golden apples are very juicy.

d_2 : Apples are sweet and sour in taste. Apples are very good for health everyone should eat apple

The set of terms with their term frequencies in respective documents is depicted in Table 2.5.

Table 2.5 Term Frequencies in respective Documents

Terms	List 1	List 2	List 3
Apple	1,2	1:3,2:3	1:(2,9), 2:(1,8)
Day	1	1:1	1:(4)
Keeps	1	1:1	1:(5)
Doctor	1	1:1	1:(7)
Away	1	1:1	1:(8)
Red	1	1:1	1:(11)
Color	1	1:1	1:(13)
Golden	1	1:1	1:(14)
Very	1,2	1:1, 2:1	1:(17), 2:(10)
Juicy	1	1:1	1:(18)
Sweet	2	2:1	2:(3)
Sour	2	2:1	2:(5)
Taste	2	2:1	2:(7)
Good	2	2:1	2:(11)
Health	2	2:1	2:(13)
Everyone	2	2:1	2:(14)
Eat	2	2:1	2:(16)

The indexes can be described by following lists:

- *List 1*: Only the documents are listed. This list is represented in the format as: (d_1, d_2, \dots) , where d_i denotes the document number i.e. document identifier.
- *List 2*: Documents are listed with their word frequencies in document. The format is $(d_1: f_1, d_2: f_2 \dots)$, where d_i represents the document identifier and f_i denotes the word frequency.

- List 3: Documents are listed with their word positions and word granularity. The format is $(d_1: (w_1, w_2, \dots), d_2: (w_1, w_2, \dots), \dots)$, where d_i is the document identifier and w_j are the word positions.

Advantages:

- Generally, this approach for indexing the documents is used in Full-text searching. It is very easy method for users who are not sure what or which type of documents they want to retrieve.
- Computing the frequency of occurrence of each word in every document provides the basis for optimizing query execution and recommendation.

Limitations:

- This method results in a huge number of irrelevant result lists of documents.
- Major drawback of this method is to rebuild the inverted list instead of updating the existing list while adding a new document to collection. This results in high cost in terms of time and space.
- This approach results in indexing individual words only, whereas researchers often uses domain names, topic phrases, title of documents etc. while searching for desired results in digital libraries.

b) Signature Files: Signature Files [77, 78], also named as word-oriented index structures, which process each word of the document separately by using a hashing function (or also called signature). For generating the signature, pre-processing of document is done (i.e. applying stemming and stop words removal etc.) to get the indexable tokens. A binary pattern is generated by setting a constant number of 1s (say m) in the range between $[1..V]$. This pattern is named as the word signature. This method divides the text of the document in a number of blocks and each block is having b non-common, distinct words. Now, each word is mapped into bit vector. The length of each bit vector is V bits. A block signature in the form of V -bit pattern is generated by superimposing (i.e. bit ORed) each word signature present in a single block and saved in

the signature file. There are various applications of this approach like in office filing, hypertext systems, as well as in data mining.

Illustrative Example: Consider small fragment of documents containing a block having 3 words (say $D=3$), the length (V) of each signature is=12 and $m=4$ (as shown in Table 2.6)
 $D1=$ “SGML”, $D2=$ “Database”, and $D3=$ “Information”

When a user hits a query, the system first generates query signature s_q from query keywords. Now, the comparison is done between the query signature s_q and every block signature s_b present in signature files. The possible comparison outcomes are shown in Table 2.6.

- If $s_b \cap s_q = s_q$, The block is matched with the query.
- If $s_b \cap s_q \neq s_q$, The block is not matched with the query.
- The result of comparison comes out to be matched but there is a false drop i.e. the block signature is not matched with the query signature. In order to overcome false drops, the further examination of block must be done.

Table 2.6 Signature Generation and Comparison

Word Signature		Queries & Signatures		Results
SGML	010 000 100 110	SGML	010 000 100 110	Match with Block Signature
Database	100 010 010 100	XML	011 000 100 100	No Match
Information	010 100 011 000	Intonation	110 100 100 000	False drop
Block Signature	110 110 111 110			

Advantages:-

- It is more efficient approach if the users use phrases-type and proximity-type queries.
- Unlike Inverted Index, this method handles new insertions and queries more efficiently.

Limitations:-

- Results in a more number of false drops, which can be eliminated by doing only sequentially search on every block signature which results in a false drop output.
- In the case of large databases, signature files result in slow execution because their response time is linear on the number of items in the database.
- This approach allows insertions with more cost and needs significant space overhead.

c) Citation indexing: This type of indexing was proposed in 1950s by Institute for Scientific Information (ISI) which is also named as Thomson Reuters [79]. This method assumes that, there are three strategies generally used by the researchers in finding their interest of research work [80]:

1. Follow the references or citations of the known document made by their authors.
2. Searching through bibliographies or indexing services by using subject words.
3. Consult a subject expert of the area who gives a direction to the researcher about other information like tools, techniques, authors, citations of that area which helps the researcher.

This indexing makes a link between article and their citations i.e. who cite that article for reference. It is a technique which allows us to trace all articles or idea from the older publication to recently published who have cited the older publications. For making the relationship between an older publication and recently publication, this technique considers the references or footnotes or endnotes (citations) in the recently published article. There are numerous advanced methods which are proposed for searching the article that are related to each other based on citations, text, and usage information. An Autonomous Citation Indexing (ACI) system was invented by CiteSeer [15] which automatically extracts the context of the citations and creates a citation index in electronic format. This system enables to:

- autonomously search articles,
- automatically extract references or citations,

- identify citations to the same article but written in different formats, and
- identify the context of citations.

Illustrative Example: Let’s take a look how citation index [81] works? When a research hits a query on a query interface, a list of citations which is matched with the query is returned by the system. The researcher can further browse the articles by tracing the references between the articles made by citations. As shown in Fig. 2.7, the system

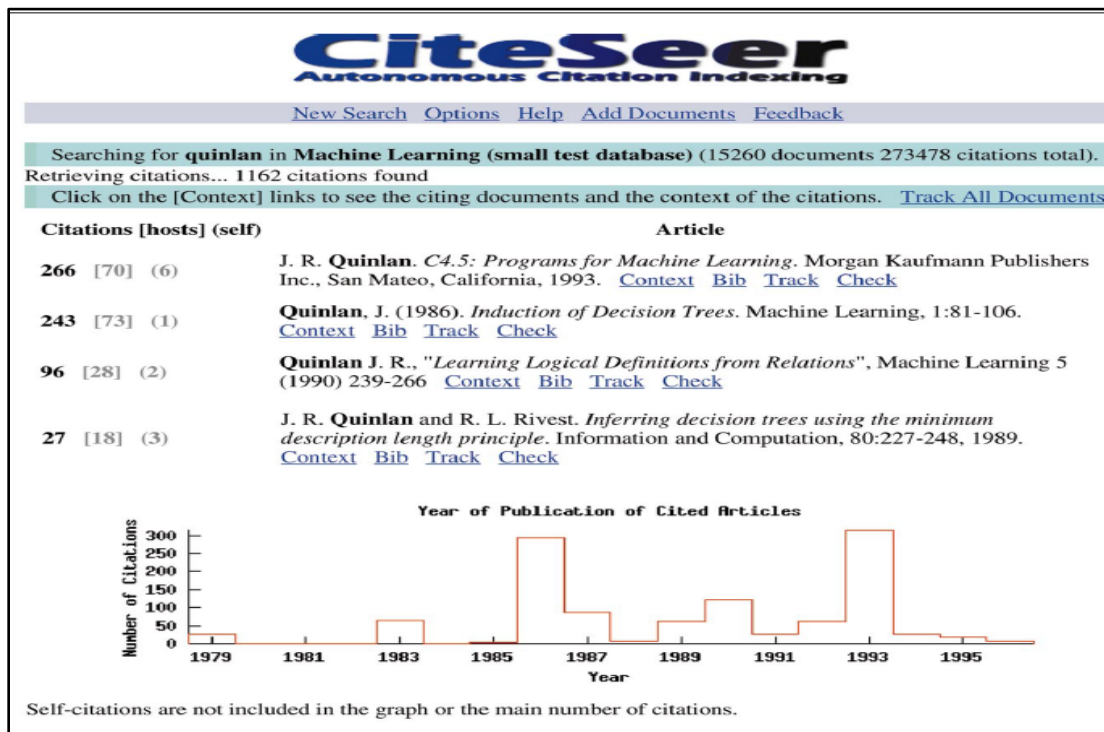


Fig.2.7 An Example of Citation Indexing

returns a number of citations to each article against the query “Quinlan” hit by the researcher [81]. The “hosts” column defines the total number of unique hosts. The “self” column represents the total number of self-citations of given paper. The graph shows the total number of citations on the vertical side versus the year of publication for each cited article.

Advantages:

- This method helps to reveal relationships among various publications.

- This method helps the researchers to draw attention towards various important corrections or retractions related to publish work of their interest area.
- The method enables the researchers to identify significant improvements or criticisms made on the previous work of a particular publication.

Disadvantages:

- Today, three type of citation indexes are available as: Science Citation Index (SCI), Social Sciences Citation Index (SSCI), and Arts & Humanities Citation Index (AHCI). But, the main drawback is that they need manual efforts for selective indexing.

d) Keyphrase Indexing: Conventional systems often provide the indexing at term or word level that appears in the document. But, when a researcher is interested to search for an article or document in terms of topics/domains, then conventional indexing based system returns a long list of documents. As a result, it is difficult for the researcher to find whether the returned list fully covered his/her interested area or which kind of refinement in queries will provide the fruitful results. Thus, to overcome the above problem, an indexing scheme, named as Keyphrase based Indexing was proposed in [82]. Keyphrases are topical words or phrases from the document which provide the concise description of the document content. This approach automatically extracts the keyphrases from the document which form the basic unit for indexing. This method allows the users to interact with the collection at the level of topics and subjects rather than words and documents. A system named as keyphind [83] was proposed that allows browsing, exploring, and searching large collections of text documents. This system uses the keyphrase indexing for retrieving documents. There are total three type of indexes are generated by the system [82] (as shown in Fig. 2.8). These are as follow:

- 1) *Word-to-phrase index:* This type of index results in creating a list of keyphrases against all words or terms which appear in the document collection. For example, “crawling” word presents in “focused crawling,” “incremental crawling,” “crawling techniques,” etc.

The screenshot shows the Keyphind application window with a search for the term "text". The interface is divided into two main panes. The left pane lists phrases and their frequency in documents, while the right pane lists phrases that occur with the search term "text retrieval" and their frequency in those documents. Below these panes is a section for document previews.

Phrase	# Docs	Occurring with "text retrieval"	# Docs
text editor	12	information retrieval	4
text compression	11	full text	2
text retrieval	10	natural language processing	2
full text	8	retrieval system	2
program text	8	document retrieval	2
text database	7	signature file	2
text generation	7	engineering sector	1
structured text	6	signature length	1
input text	5	office information system	1
plain text	5	search string	1
text element	5	signature size	1
text entry	5	machine learning	1
text line	5	schema-independent retrieval	1
text processing	5	term clustering	1
text widget	5	feature selection	1
text window	5	navigation tool	1
free text	4	office system	1
text editing	4	relational database system	1
text file	4	feature set	1
text search	4	recognition error	1
block of text	3	recent development	1
handwritten text	3	information retrieval system	1
natural language text	3	natural language indexing	1
parallel text	3	feature extraction	1

Document previews for "text retrieval"

- Natural language processing for information retrieval David D. Lewis AT&T Bell Laboratories Karen Sparck Jones Computer Laboratory, University of Cambridge July 199 (107)
- Document Retrieval Using Signature Files Jaber Al-Merri and D. R. McGregor Department of Computer Science University of Strathclyde 26 Richmond Street Glasgow, G1 1XH Scotl (2731)
- Querying, Navigating and Visualizing an Online Library Catalog Aravindan Veerasamy Scott Hudson Shamkant Navathe feerasam, hudson, shang@cc.gatech.edu College of Computing (2976)
- CS-TR-3514 August, 1995 A Survey of Information Retrieval and Filtering Methods P Christos Faloutsos Department of Computer Science and Douglas W. Oard Electrical Engin (5778)
- Text Search Using Database Systems Revisited | Some Experiments | Helmut Kaufmann and Hans-JPorg Schek Swiss Federal Institute of Technology (ETH ZPurich) Department of Computer Sc (6966)
- Schema-Independent Retrieval from Heterogeneous Structured Text Charles L. A. Clarke G. V. Cormack F. J. Burkowski Dept. of Computer Science? University of Waterloo, Waterloo, Canada (7271)
- Project Note MULINDEX Multilingual Indexing, Navigation and Editing Extensions for the World-Wide Web Gregor Erbach, G?nter Neumann, Hans Uszkoreit DFKI GmbH Language Techno (13944)
- Speech Retrieval Based on Automatic Indexing Martin Wechsler, Peter Sch?auble Swiss Federal Institute of Technology (ETH) ZPurich, Switzerland Abstract We present a system that (15481)
- REPRESENTATION AND LEARNING IN INFORMATION RETRIEVAL A Dissertation Presented by David Dolan Lewis Submitted to the

Fig.2.8 An Example of Keyphrase Indexing

- 2) *Phrase-to-document index*: This type of index results in creating a list of all the phrases in the collection along with the list of documents in which the phrase appears.
- 3) *Document-to-phrase index*: This type of index results in listing every document by a number and indicates all phrases that were extracted from that document.

Advantages:

- Topical orientation: When a researcher hits a query, then the system returns a list of keyphrases instead of document list.
- Phrase-based clustering: The system groups the documents that contain the same keyphrases. When a keyphrase is selected by the researcher, then a list of documents (or document cluster) is returned to the researcher instead of document based searching.
- Query refinement: When a user hits the query, then a list of keyphrases displayed by the system provides all possible ways to extend the query.

Disadvantages:

- Biasing clusters on keyphrases.
- Keyphrase-based clusters are variable in sizes. Some clusters are so large in size containing number of documents and some are too small.

e) **Latent Semantic Indexing:** Scott Deerwester et al. [84] proposed a new approach for indexing documents, named as Latent Semantic Indexing (LSI) which extracts the document from the collections based on the concepts. Unlike word-based approach, this method helps to extract more relevant and desired documents efficiently. Most retrieval systems check the similarity between the query terms and the terms present in documents; whereas LSI model retrieves the information or document based on the similarity of concept or semantic structure for finding the more relevant results. The Singular Value Decomposition (SVD) method [85] is used for performing the concept – based mapping. This method states that if two document vectors show the same topic, then they also have some number of words or keywords in common. To compute the semantic structure between these semantic similar documents, truncated SVD method is used. LSI method is also known as dimensionality reduction technique as it converts the high-dimensional space representation of terms of the documents into a low dimensional space representation.

In the SVD, a large term-by-document matrix $A(t \times d)$ is decomposed into product of three matrices. The SVD of a matrix A is written as:

$$A = U_{t \times n} * \Sigma_{n \times n} * (V_{d \times n})^T \quad (2.1)$$

where t represents the number of terms, d denotes the number of documents, n represents the unique dimension which is $\leq \min(t, d)$, U and V are orthogonal matrices i.e. $UU^T = VV^T = 1$ and Σ represents a diagonal matrix where the values on the diagonal of Σ are called the singular values.

Now, for computing the latent semantic representation, choose only top k values of Σ (say Σ_k). The remaining singular values are then set to 0. Matrix U is turned into U_k by keeping only first k columns and V matrix into V_k by keeping only the first k rows.

When the user hits the query for purpose of retrieving the information from the collection, then query is represented in the form of query vector using:

$$q = q^T U_{t \times k} \Sigma_{k \times k}^{-1} \quad (2.2)$$

After that, similarity between the query vector and documents vectors is computed by using cosine similarity coefficient. Based on this similarity value, the result list of documents is ranked.

Illustrative Example: Consider small fragments of two sample documents d_1, d_2 and query q as given below:

d_1 : Delivery of silver arrived in a silver truck

d_2 : Shipment of gold arrived in a truck

q : Gold silver truck

Step 1: Compute term-document matrix, A and Query matrix q .

$$\begin{array}{l} \text{Terms} \\ \text{Delivery} \\ \text{Silver} \\ \text{Arrive} \\ \text{Truck} \\ \text{Shipment} \\ \text{Gold} \end{array} = \quad A = \begin{array}{cc} d_1 & d_2 \\ \left[\begin{array}{cc} 1 & 0 \\ 2 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{array} \right] \end{array} \quad q = \begin{array}{c} q \\ \left[\begin{array}{c} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{array} \right] \end{array}$$

Step 2: Find the SVD for matrix A . First, compute the singular values σ_i by finding the eigen values of $A^T A$.

$$A^T A = \begin{bmatrix} 1 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 2 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 7 & 2 \\ 2 & 4 \end{bmatrix}$$

$$\begin{aligned} \det(A^T A - \lambda I) &= \begin{vmatrix} 7-\lambda & 2 \\ 2 & 4-\lambda \end{vmatrix} = \lambda^2 - 11\lambda + 24 \\ &= (\lambda - 8)(\lambda - 3) = \lambda = 3, 8 \end{aligned}$$

Thus, Singular values are $\sigma_1 = \sqrt{3}$ and $\sigma_2 = \sqrt{8} = 2\sqrt{2}$

Step 3: Find the right singular vectors (the columns of V) by finding an orthonormal set of eigenvectors of $A^T A$.

For $\lambda = 3$

$$(A^T A - \lambda I) \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0 \Rightarrow \begin{bmatrix} 4 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

Thus,

$$V_1 = \begin{bmatrix} 1 \\ \sqrt{5} \\ -2 \\ \sqrt{5} \end{bmatrix} = \begin{bmatrix} 0.4472 \\ -0.8944 \end{bmatrix}$$

Similarly, For $\lambda = 8$

$$\begin{bmatrix} -1 & 2 \\ 2 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$$V_2 = \begin{bmatrix} 2 \\ \sqrt{5} \\ 1 \\ \sqrt{5} \end{bmatrix} = \begin{bmatrix} 0.8944 \\ 0.4422 \end{bmatrix}$$

Thus,

$$V = \begin{bmatrix} 0.4472 & 0.8944 \\ -0.8944 & 0.4472 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1.732 & 0 \\ 0 & 2.828 \end{bmatrix}$$

As, $u_i = \frac{1}{\sigma} A v_i$

$$\text{Thus, } u_1 = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 0 \\ 2 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.4472 \\ -0.8944 \end{bmatrix} = \begin{bmatrix} 0.2581 \\ 0.5163 \\ -0.2581 \\ -0.2581 \\ -0.5163 \\ -0.5163 \end{bmatrix}$$

$$\text{Similarly } u_2 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 0 \\ 2 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.8944 \\ 0.4472 \end{bmatrix} = \begin{bmatrix} 0.3162 \\ 0.6324 \\ 0.4743 \\ 0.4743 \\ 0.1581 \\ 0.1581 \end{bmatrix}$$

$$\text{So } U = \begin{bmatrix} 0.2581 & 0.3162 \\ 0.5163 & 0.6324 \\ -0.2581 & 0.4743 \\ -0.2581 & 0.4743 \\ -0.5163 & 0.1581 \\ -0.5163 & 0.1581 \end{bmatrix}$$

Step 4: Find the query vector.

$$q = q^T U_K \Sigma_K^{-1}$$

$$q = [0 \ 1 \ 0 \ 1 \ 0 \ 1] \begin{bmatrix} 0.2581 & 0.3162 \\ 0.5163 & 0.6324 \\ -0.2581 & 0.4743 \\ -0.2581 & 0.4743 \\ -0.5163 & 0.1581 \\ -0.5163 & 0.1581 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1.7320 & 0 \\ 0 & 1 \\ & 2.828 \end{bmatrix}$$

$$q = [-0.1490 \ 0.4472]$$

Step 5: Compute the cosine similarity between query vector and documents by using the equation as shown below:

$$Sim(q, d) = \frac{q * d}{|q||d|}$$

$$Sim(q, d_1) = \frac{-0.1490 * 0.4472 + 0.4472 * 0.8944}{\sqrt{(-0.1490)^2 + (0.4472)^2} \sqrt{(0.4472)^2 + (0.8944)^2}} = 0.7074$$

$$Sim(q, d_2) = \frac{-0.1490 * (-0.8944) + 0.4472 * 0.4472}{\sqrt{(-0.1490)^2 + (0.4472)^2} \sqrt{(-0.8944)^2 + (0.4472)^2}} = 0.7070$$

Here, it is concluded that document d_1 scores higher than d_2 . Its vector is closer to the query vector than other vectors.

Advantages:

- This approach considers the semantic structure of the terms with documents for retrieving more relevant results as per user interest.
- LSI helps to reduce the problems of lexical matching by using conceptual indices instead of literal terms of the documents for retrieval.
- This method helps to find the best similarity between small groups of terms, in a semantic way (i.e. in a context of a knowledge corpus).
- LSI is able to handle the two major problems of keyword based queries i.e. synonymy and polysemy.

Disadvantages:

- This method needs relatively high computational performance and memory in comparison to other information retrieval techniques.
- More complexity exists in determining the optimal number of dimensions for computing the SVD.
- To perform SVD on huge corpus is not feasible as the complexity of this method increases with increasing the number of terms and documents.

A brief summary of the indexing techniques [169] described above is shown in Table 2.7. Along with this, a list of few digital library search engine is also shown in Table 2.8 with their features and the information on what type of indexing is used by them.

Table 2.7 Summary of Indexing Techniques

Indexing Type	Features	Applications	Advantages	Disadvantages
Inverted Indexing [74, 77]	<ul style="list-style-type: none"> • Full-Text Search. • Stores the frequency of occurrence of a term in documents in which the term appears, in the form of a term's inverted list. 	<ul style="list-style-type: none"> • Used in Query Optimizing. 	<ul style="list-style-type: none"> • Easy to locate the information when the researcher is not aware of what he want to search for. 	<ul style="list-style-type: none"> • Results in high number of irrelevant list of documents or information. • Rebuild the inverted list while adding a new document to collection which results in high cost.
Citation Indexing [79, 80, 81]	<ul style="list-style-type: none"> • Citation analysis is done by considering citations or hyperlinks between documents. • Compute the rank or priority of the publication based on the number of time it has been cited. • Find papers that cite earlier papers. 	<ul style="list-style-type: none"> • Analyze research trends. • Identify emerging areas of science, and find out where and how often a particular article is cited. 	<ul style="list-style-type: none"> • Identify relationships among various publications. • Identify significant advancement made on the previous work of particular publication. • Evaluate the prestige of an author more accurately and quickly. 	<ul style="list-style-type: none"> • The citations to one article show typical variations in their format. It is very difficult to recognize that all of these citations refer to the same article.
Keyphrase Indexing [82, 83]	<ul style="list-style-type: none"> • Keyphrases are considered as a basic unit for indexing the documents. 	<ul style="list-style-type: none"> • Provides labels for text documents. • Provides a concise description of a document's content. • Used in document categorization, clustering. 	<ul style="list-style-type: none"> • Topical orientation. • Phrase-based clustering. • Query refinement. • Easy to build. 	<ul style="list-style-type: none"> • Basing clusters on keyphrases. • The sizes of keyphrase-based clusters are variable.
Latent Semantic Indexing (LSI) [84, 85]	<ul style="list-style-type: none"> • Document-term matrix is used. • SVD method is used for performing the concept –based mapping. 	<ul style="list-style-type: none"> • Automated document classification • Text summarization. • Used for electronic document discovery (eDiscovery). 	<ul style="list-style-type: none"> • Find the best similarity between small groups of terms, in context of a knowledge corpus. • Solve the problem of synonymy and polysemy. 	<ul style="list-style-type: none"> • The expensive complexity involved in computing truncated SVD.

Signature files indexing [77, 78]	<ul style="list-style-type: none"> • Word-oriented index structures based on hashing. • Preprocessing of the document is done (lexing, stemming and stop words removal etc.) for getting the indexable tokens. 	<ul style="list-style-type: none"> • Used in text indexing methodology. • Utilized in office filing, hypertext systems, relational and object-oriented databases, as well as in data mining. 	<ul style="list-style-type: none"> • It is more efficient approach if the users used phrases-type and proximity-type queries. • Handles new insertions and queries more efficiently. 	<ul style="list-style-type: none"> • Results in huge number of False Drops which can be eliminated by doing only sequentially search on every block signature.
--	--	--	--	---

Table 2.8 Different Indexing Methods used by Digital Library Search Engines

Digital Library Search Engine	Type of Indexing	Features
GOOGLE SCHOLAR [80]	<ul style="list-style-type: none"> • Full-Text Indexing • Citation Indexing 	<ul style="list-style-type: none"> • Indexes the full text or metadata of scholarly literature.
YAHOO [89]	<ul style="list-style-type: none"> • Humans are relied upon for indexing. 	<ul style="list-style-type: none"> • Hierarchically taxonomy is used to organize the collection. • Robots.txt file is used to explore more sites. • Indexing the document features includes URL, HTML title tags, and short description.
ALTAVISTA [86]	<ul style="list-style-type: none"> • Meta tags are used for indexing. 	<ul style="list-style-type: none"> • Indexes every word in every page but does not retrieve stop words. • Allows the proximity searching with the connector "NEAR".
EXCITE [90]	<ul style="list-style-type: none"> • Full Text Indexing 	<ul style="list-style-type: none"> • Uses concept extraction approach. • Clustering of words is used to find the concept. • Uses robots to do full text indexing. • Multi-level indexing is used.
INFOSEEK [88]	<ul style="list-style-type: none"> • Meta descriptor tags are used for Indexing. 	<ul style="list-style-type: none"> • Uses robot to do full text indexing. • Indexes third and fourth level also
CiteSeer ^x [15]	<ul style="list-style-type: none"> • Autonomous Citation Indexing (ACI) • Full text Indexing 	<ul style="list-style-type: none"> • A web-based scientific literature digital library. • Computing the citation count and re related articles for all documents cited in the collection. • Provides improvements in terms of cost, availability and efficiency. • Facilitates the researchers by providing easy navigation and evaluation of citations by linking the references automatically in research articles.
Academic Search [87]	<ul style="list-style-type: none"> • Full-Text Indexing • Citation Indexing (Related Article feature) 	<ul style="list-style-type: none"> • Monthly indexing service

2.6.3 Study of Recent Indexing Techniques

A literature survey of various indexing techniques used by digital library search engines has been done in this section. Few existing indexing techniques proposed by researchers are discussed below:

a) Indexing Technique using Hierarchical Clustering: An approach to index documents more efficiently by using hierarchal clustering is being proposed by Deepti Gupta et al (2009) [91]. This method uses the Agglomerative Hierarchical clustering algorithm in order to index the information based on similarity measure and fuzzy string matching. The system employs both Euclidean metric and Levenshtein metric [92] for similarity calculation and fuzzy string matching respectively. This technique keeps the related documents in the same cluster so that searching of documents becomes more efficient in terms of time complexity.

b) Context based Indexing using Ontology: In this method [93], index is built on the basis of context of the document rather than on the basis of terms. The ontology-based collection method is presented in this paper which uses context to describe collections and search engines. The context of the documents being collected by the crawler in the repository is being extracted by the indexer using the context repository, thesaurus and ontology repository. The documents are then indexed according to their respective context.

c) Trie Structure based Indexing: An improved indexing mechanism to index the web documents is being proposed by Pooja Mudgil et al. [94] that keep the context related information integrated with the frequency of the keyword. The structure is implemented using Trie. The proposed contextual based indexing has considered the presence of keywords in various HTML tags of web documents such as head, title, keyword, description, body and link. The weight is assigned to each of these tags and stored using Trie structure. This will help to optimize the speed and performance in finding the relevant documents for a search query.

d) Concept-Based Semantic Annotation and Indexing: Sasa Nesic et al. [95] presented an ontology-driven approach to semantic annotation, indexing and

retrieval of fine-grained units of document's data. In this approach, the document units and the user query are both represented by weighted vectors of ontological concepts. To determine the relevance of the document units to given query, similarity between their concept vectors is measured. The key part of this approach that distinguishes it from similar existing approaches is the concept exploration algorithm, which calculates the semantic distances between concepts in the ontology based on the ontology relationships.

e) **Sentence Context Ontology based Indexing:** The author [96] proposed a conceptual framework for modeling contexts associated with sentences in research articles. The system also presented the Sentence Context Ontology, which is used to convert the information extracted from research documents into machine-understandable data. The system presented a linked data application which uses a new semantic publishing model for providing value added information services for the research community. The system provides a feature of classifying the citations based on the reasons used in the articles and also evaluated the citation analysis based on different contexts of citations to the cited works and the author timeline.

2.6.4 Comparison of Different Indexing Techniques

After extensive study of some of prevalent indexing schemes, it is concluded that each approach has some relative strengths and limitations. A detailed comparison of various indexing approaches such as Hierarchical Clustering based indexing, Trie structure based indexing, Context based indexing and Sentence Context Ontology based indexing used by different digital library search systems is shown in Table 2.9. Comparison is done on the basis of some measures such as main features, data structure used, type of indexing, applications in various fields, their advantages and disadvantages

In next section, the working of query processing is described in detail.

Table 2.9 Comparison of Indexing Techniques

Techniques	Indexing Technique using Hierarchical Clustering [91]	Trie Structure based Indexing [94]	Context based Indexing using Ontology [93]	Sentence Context based Indexing [96]
Main Technique Used	Agglomerative Hierarchical clustering algorithm is used by the system in order to keep the information based upon similarity measure and fuzzy string matching.	This method keeps the context related information integrated with the frequency of the keyword.	An index is built on the basis of context of the document rather than on the terms basis using ontology.	A linked data application is developed which provides intelligent information services using the extracted information from research articles using Citation Context Analysis, Conditional Probabilistic Models and Semantic Web for modeling Scientific Discourse
Type of indexing	Agglomerative Hierarchical clustering based indexing	Contextual based indexing	Context based indexing using Ontology	Citation Indexing
Data Structure Used	Inverted Index	Trie type tree structure	Simple inverted index	Graph based Structure
Advantage	The related documents are grouped in the same cluster so that searching of documents becomes more efficient in terms of time complexity.	It helps to optimize the speed and performance in finding relevant documents for a search query	Fast access to documents.	Classification of the citations. Evaluation of the citation analysis based on the different contexts of citations to the cited works and the author timeline.
Limitation(s)	The complexity of this method is $O(n^3)$ which makes it very slow for large databases.	More space is required to store Trie structure for large dataset.	No consideration of ambiguity if the user is not aware of the context.	A larger training dataset is required with a focus on achieving a higher accuracy,

2.7 QUERY PROCESSING

With the rapid growth of document database, there is a rapid increase in the number of users and consequently, in the number of queries submitted by the users to information retrieval systems. As document collections grow larger, it becomes more challenging and expensive task to manage them by an information retrieval system [97]. Furthermore, as the number of queries increases, it becomes even more important to provide high query processing rates on these collections.

Query processing mainly consist of following phases [98, 99]:

Step 1: Tokenizing: When a user inputs a query, the query processing engine must tokenize the query stream, i.e., break it down into understandable segments.

Step 2: Parsing: Since users may employ special operators in their query such as Boolean or proximity operators, the system needs to parse the query first into query terms and operators.

Steps 3: Stop word removal and stemming: Stop-words are language-specific functional frequent words that carry no information (i.e., pronouns, prepositions, conjunctions). Examples of such words include 'the', 'of', 'and', 'to'. The first step during preprocessing is to remove these Stop words. Stemming techniques [100] are used to find out the root/stem of a word. Stemming converts words to their stems, which incorporates a great deal of language-dependent linguistic knowledge. For example, the words, *user*, *users*, *used*, using all can be stemmed to the word 'USE'.

Step 4: Creating the query: How each particular search engine creates a query representation depends on how the system does its matching. If a statistically based matcher is used, then the query must match the statistical representations of the documents in the system. If a Boolean matcher is utilized, then the system must create logical sets of the terms connected by AND, OR, or NOT.

Step 5: Query Expansion: Since users of search engines usually include only a single term in a query, thus it becomes highly probable that the information they need may be expressed [101] using synonyms, rather than the exact query terms, in the documents which the search engine searches against. Therefore, more sophisticated systems may

expand the query into all possible synonymous terms and perhaps even broader and narrower terms.

Step 6: Query Term Weighting: The Query processing involves computing weights for the terms in the query. Sometimes the user controls this step by indicating either how much to weight each term or simply which term or concept in the query matters most and must appear in each retrieved document to ensure relevance.

Step 7: After this step, the expanded, weighted query is searched against the index by matching the constituent terms with index terms. In response, a set of matched documents are retrieved.

Step 8: Ranking: Before displaying the results, a ranking mechanism is applied to the list of result out documents.

But before discussing the various state-of-the-art techniques in digital libraries in the next chapter, some of the major issues pertaining to the design of effective and efficient digital library search engines have been described in the next section.

2.8 DESIGN ISSUES IN DIGITAL LIBRARY SEARCH ENGINES

Although, the current digital library search engines come up with advanced crawling and indexing techniques which efficiently gather and index the documents, there still exist many issues which need to be addressed as described below:

- ***Large Volume:*** Today's web consists of billions of digital documents, the extraction of desired content from which is a tedious task. Digital Library Search Engines should be able to index most of the information available on WWW in an efficient manner.
- ***Distributed nature of Data:*** The documents on the web are distributed across various servers employing different platforms such as different digital libraries, author homepages etc. Digital library Search engine must be designed in a way to cope up with this distribution and accumulate the content in its local repository.
- ***Relevancy of Results:*** As most of the digital library search engines are keyword

based, the retrieval of relevant documents is a challenging task. Digital library Search Engines generally return so many search results that user wastes most of the time sifting between them for uncovering the desired information, thus leading to the problem of Information Overkill. Digital library search engines must be capable of returning desired documents at least on the top of result list.

- ***Extensibility***: Digital Library Search engines should be extensible in the sense so as to support third party functional modules e.g. mining modules [102], ranking modules and query expansion [101] modules etc. to make them more efficient.
- ***Efficient data structures***: The existing data structures employed in the indexing process of digital library search engines lack valuable information related to relevant document retrieval and are generally keyword based. Therefore, a large number of irrelevant documents are returned posing the problem of Information Overkill. This problem can be resolve by indexing the document using multi-level index structure which provides better efficiency and effectiveness of digital library search engines.

In order to resolve these challenges, some state-of-the-art techniques in digital libraries play an important role. The next chapter is devoted to the survey of techniques such as ranking, cluster analysis and document categorization etc. in context of digital libraries.

Chapter III

STATE-OF-THE-ART TECHNIQUES IN DIGITAL LIBRARIES

3.1 INTRODUCTION

WWW [3, 4] is a vast source of dynamic and unstructured information repository covering almost every possible digital document. These digital documents contain rich textual information, but the exponential growth of the WWW has made it rapidly difficult for researchers to find the desired and relevant content in a fast manner on the Web. Thus, to retrieve effective and relevant digital information, many digital library search engine technologies [15] are now used as automated tools in order to find, extract, filter, and evaluate the desired information and resources. A lot of algorithms and approaches have been reported in the literature. These approaches and techniques are well studied and implemented for different applications and scenarios by researchers. In the next sections, some prevalent PageRanking, Clustering and Document Categorization techniques have been described.

3.2 PAGE RANKING

Today, the main challenge in front of search engines is to efficiently harness scientific work present on the WWW and present relevant results to the user. Web mining techniques are used in order to extract the relevant documents and order them. To represent the documents in an ordered manner, Page ranking methods are being applied which can arrange the documents in order of their relevance and importance. Some of the common page ranking algorithms for online digital libraries have been discussed in this section.

3.2.1 Citation Count Algorithm

This is one of the most frequent used ranking algorithms for measuring a scientist's reputation, and named as Citation Count (CC) [103]. This method uses the citation graph of the web to determine the ranking of scientific work. In citation graph, the nodes

represent publications, whereas an edge from node i to node j represent a citation from paper i to paper j i.e. a vote from paper i to paper j . This method states that if a publication has more number of citations (incoming links) to it, publication becomes important. Therefore, it takes backlinks into account to order the publications. Thus, a publication obtains a high rank if the number of its backlinks is high. Citation Count is defined in (3.1):

$$CC_i = |I_i| \quad (3.1)$$

where CC_i represents the citation count of publication i , $|I_i|$ denotes the number of citations (in-degree) of the publication i .

Example Illustrating Working of CC: To explain the working of Citation Count, let us take an example of citation graph as shown in Fig. 3.1, where A, B, C, D, E and F are six publications.

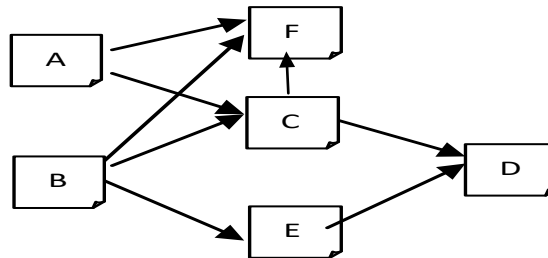


Fig. 3.1 Example of Citation Graph

The Citation Count for publications A, B, C, D, E and F can be calculated by using (3.1):

$$CC(A)=0, CC(B)=0, CC(C)=3, CC(D)=2, CC(E)=1, CC(F)=2$$

The ranking of publications based on Citation Count become:

$$CC(C) > (CC(D), CC(F)) > CC(E) > (CC(A), CC(B))$$

Limitations of CC: There are a number of cases where this method fails to reveal the good picture of influence of publications in its domain [103]. Few of reasons for this are:

- It does not take into account the importance of citing paper i.e. citation from a reputed journal gets the equal weightage as the citation from A non-reputed one.
- If two papers have similar citation count e.g. the publication D and publication F

shown in Fig 3.1, but interestingly publication F is almost 20 years younger than the publication D, thus it had a much smaller time window to accumulate citations. Thus, it does not take into consideration different characteristics of the citations, like their publication date.

3.2.2 Time dependent Citation Count Algorithm

Ludmila Marian [104, 105] proposed an extension to standard Citation Count method named as Time Dependent Citation Count (TDCC). It is a time-dependent approach which takes into account time of the citation. This method assumes that the freshness of citations and link structure are factors that need to be taken into account in citation analysis while computing the importance of a publication. Thus, Citation Count algorithm is modified by initially distributing random surfers exponentially with age, in favor of more recent publications. The method introduces the effect of time in the citation graph by applying a time-decay factor to the citation counts. The weight of a publication i is denoted as $Weight_i$ as given in (3.2)

$$Weight_i = e^{-w(t_p - t_i)} \quad (3.2)$$

where t_i denotes the published year of publication i , t_p denotes the present time (i.e. year), and w denotes the time decay parameter ($w \in (0, 1]$), which quantifies the notions of “new” and “old” citations (i.e. publications with ages less than the time decay parameter would be considered “new”; publications with ages larger than the time decay parameter would be considered “old”) citations (in-degree) of the publication i .

Example Illustrating Working of TDCC: To illustrate the working of TDCC, let us refer again to Fig 3.1 and the Table 3.1. By using (3.2) weight scores of publications can be calculated as:

Table 3.1 Data of Citation Graph

Publication	Publication year
A	2011
B	2008
C	1998
D	1980
E	2007
F	2000

$$\text{Weight}_A = 0 \quad (3.2a)$$

$$\text{Weight}_B = 0 \quad (3.2b)$$

$$\text{Weight}_C = e^{-w(2012-2011)} + e^{-w(2012-2008)} + e^{-w(2012-2000)} = e^{-w(1)} + e^{-w(4)} + e^{-w(12)} \quad (3.2c)$$

$$\text{Weight}_D = e^{-w(2012-1998)} + e^{-w(2012-2007)} == e^{-w(14)} + e^{-w(5)} \quad (3.2d)$$

$$\text{Weight}_E = e^{-w(2012-2008)} = e^{-w(4)} \quad (3.2e)$$

$$\text{Weight}_F = e^{-w(2012-2011)} + e^{-w(2012-2008)} = e^{-w(1)} + e^{-w(4)} \quad (3.2f)$$

where w is time decay factor. Let us take the threshold age = 6 years. Here $w=0$ for the publications with the ages less than 6 years (considered new publications) and $w=1$ for publications with ages more than 6 years (considered old publications). By calculating the above equations, the rank score of publications become:

TDCC (A) = 0, TDCC (B) = 0, TDCC (C) = 2.0000006144, TDCC (D) = 1.000000832, TDCC (E) = 1, TDCC (F) = 2

Here,

$$\text{TDCC}(C) > \text{TDCC}(F) > \text{TDCC}(D) > \text{TDCC}(E) > (\text{TDCC}(A), \text{TDCC}(B))$$

It may be noted that the resulting ranking of citations obtained by CC and TDCC are different.

Advantages and Limitations of TDCC: After adding a time decay parameter, the time-dependent ranking can differentiate between an old publication that acquired a large number of citations over a long period of time, and a new publication [104, 105]. The main disadvantages of this method are as:

- Adding a weak or strong time decay factor to a ranking method will have an impact on the final ordering of the documents. For example, adding a strong time decay factor to ranking will reveal the most popular publications at the current moment in time.
- Like CC, this method does not take into consideration the different importance of each citation.

3.2.3 PageRank Algorithm

Surgey Brin and Larry Page [10, 36] proposed a ranking algorithm, named as PageRank (PR) which extends the idea of citation analysis. In citation analysis, the incoming links are treated as citations which provide importance to a page but this technique could not provide fruitful results. In turn, PageRank [10] provides a better approach which is based on the fact, that the importance of a research paper can be judged by the number of citations the paper has from other research papers. This algorithm states that if a link comes from an important paper then this link is given higher weightage than those which are coming from non-important papers. These links are called as backlinks. The PageRank of a paper u can be calculated as:

$$PR(u) = (1-d) + d \sum_{v \in B(u)} \frac{PR(v)}{N_v} \quad (3.3)$$

where u represents a paper, $B(u)$ is the set of papers that point to u , $PR(u)$ and $PR(v)$ are rank scores of papers u and v respectively, N_v denotes the number of outgoing links of paper v , and d is a normalization factor.

Example Illustrating Working of PR: Let us take a previous example as shown in Fig 3.1 in order to explain the working of PageRank algorithm. The PageRanks for papers can be calculated by using (3.3):

$$PR(A) = (1 - d) + d(0) \quad (3.3a)$$

$$PR(B) = (1 - d) + d(0) \quad (3.3b)$$

$$PR(C) = (1 - d) + d \left(\frac{PR(A)}{2} + \frac{PR(B)}{3} + \frac{PR(F)}{1} \right) \quad (3.3c)$$

$$PR(D) = (1 - d) + d \left(\frac{PR(C)}{1} + \frac{PR(E)}{1} \right) \quad (3.3d)$$

$$PR(E) = (1 - d) + d \left(\frac{PR(B)}{3} \right) \quad (3.3e)$$

$$PR(F) = (1 - d) + d \left(\frac{PR(A)}{2} + \frac{PR(B)}{3} \right) \quad (3.3f)$$

Let us assume the initial PageRank as 1, d is set to 0.85 and do the calculation. The rank values of papers are iteratively substituted in above page rank equations to find the final values until the page ranks get converged as shown in Table 3.2.

As can be observed from Table 3.2, the page ranks of papers become:

$$\text{PR (D)} > \text{PR (C)} > \text{PR (F)} > \text{PR (E)} > (\text{PR (A)}, \text{PR (B)})$$

Table 3.2 Iteration Method for PageRank

Iterations	PR (A)	PR (B)	PR (C)	PR (D)	PR (E)	PR (F)
0	1	1	1	1	1	1
1	0.15	0.15	1.106	1.090	0.192	0.256
2	0.15	0.15	0.474	0.552	0.192	0.256
3	0.15	0.15	0.474	0.552	0.192	0.256

Advantages and Limitations of PR: One of the main advantages of this method is that it ranks the publications accordingly to the importance of their citations, bringing to light some very insightful publications that would not have been discovered with the Citation Count method. On the other hand, there are some shortcomings of this ranking method also as listed below [106]:

- The rank score of publication is equally distributed among its all references irrespective of assigning the larger rank values to more important papers.
- A page rank of a publication is mostly affected by the scores of the publications that point to it and less by the number of citations. For example, in Fig. 3.2, node F gets higher score than node E, although node E gets 4 citations and node F gets 1 citation.

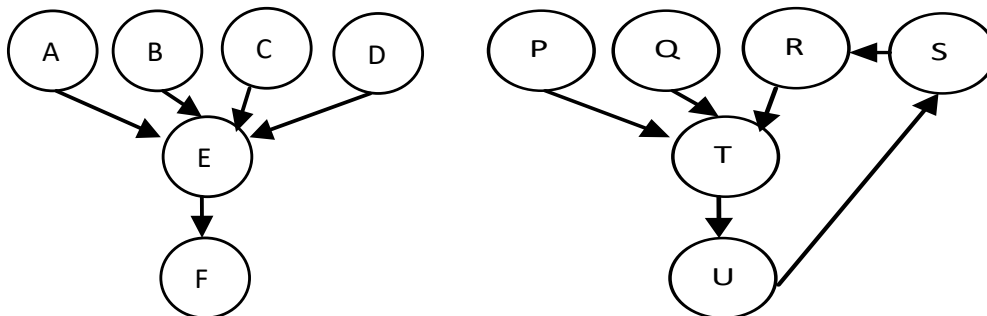


Fig. 3.2 Example of a Graph

- PageRank gives high score to a node u , if it contained a cycle. For Example, Table 3.3 shows the rank results of graph shown in Fig 3.2. In this, node E gets 4 citations, whereas node T gets 3 citations. However, the PageRank score of node T is about 2 times higher than that of node E . This happens because node T is a part of citation cycle. But in bibliometrics, cycles represent the self-citations which do not occur in citation graph. Thus, PageRank does not provide fruitful results in bibliometrics.

Table 3.3 Rank Results of Example Graph

Node	A	B	C	D	E	F	P	Q	R	S	T	U
CC	0	0	0	0	4	1	0	0	1	1	3	1
PR	0.15	0.15	0.15	0.15	0.66	0.71	0.15	0.15	1.15	1.28	1.38	1.32

3.2.4 Popularity Weighted Ranking Algorithm

Yang Sun and C. Lee Giles [107] gave a new ranking method based on PageRank with significant improvement for ranking academic papers, named Popularity Weighted Ranking algorithm. This method combines the concepts that seem to be important for analyzing the importance of publication. The publication importance is determined on the basis of the weighted citations from the other papers and a popularity factor of its publication venue i.e. quality of the publication venue where a publication is published. Unlike impact factor, it does not differentiate between journals, conferences and workshop proceedings. The popularity factor of a publication venue v in a given year is defined by (3.4)

$$PF(v, t) = \frac{n_v}{N} \times \sum_{i \in P} \frac{PF(i, t) \times w(i)}{N(i)} \quad (3.4)$$

where $PF(v, t)$ represents the popularity factor of publication venue v in a given year t , P represents the set of publication venues i which cite v in that year, n_v denotes the number of papers published in venue v in that year, $w(i)$ is the weight which represents the frequency that venue i cites venue v and $N(i)$ denotes the total number of references generated by venue i . Considering the importance of popularity factor of publication venue, the ranking score of publication p at a previous time t is given in (3.5).

$$R(q_t) = PF(v_{p_t}) + \sum_{t>T, q_t \in D} \frac{R(q_t)}{N(q_t)} \quad (3.5)$$

where $R(q_t)$ represents the ranking score of a paper q_t , which is published at time t and cite paper p_T , D represents the set of papers which cite p_T , $N(q_t)$ denotes the number of references in paper q_t , $PF(v_{p_T})$ denotes the popularity factor of the publication venue v where paper p_T is published.

Advantages and Limitations of Popularity Weighted Ranking Algorithm: One of the main advantages of this method is that it overcomes the limitations of impact factor by considering the impact of all publication venues and the probability of reader access.

- This algorithm works well for most queries but it does not work well for others.
- This method assumes that ranking score of a previously published paper will not have any impact on later published ones i.e. it does not take into consideration the time of publication.
- This method also does not differentiate between the popular and prestigious authors who published the papers.

3.2.5 HITS Algorithm

Kleinberg [108, 109] proposed a more refined notion for the importance of the web pages called Hyperlink Induced Topic Search (HITS). This method identifies two different forms of Web pages called hubs and authorities. Authorities are pages having important contents and hubs are pages that act as resource lists, guiding users to authorities as shown in Fig 3.3. A good authority is a page pointed to by good hubs, while a good hub

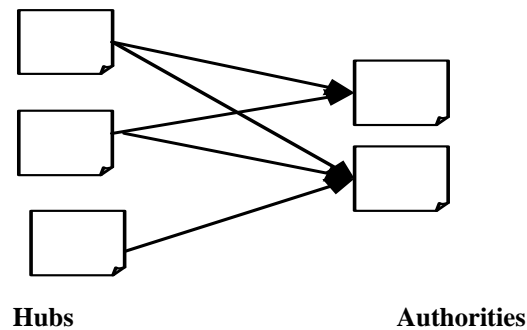


Fig 3.3: Hubs and Authorities

is a page that points to good authorities. A page may be a good hub and a good authority at the same time.

HITS functions in two major steps.

1. *Sampling Step:* In this step, a set of relevant pages for a given query are collected i.e. a sub-graph S of G is retrieved which is high in authority pages [110]. The algorithm starts with a root set R selected from the result list of a digital library search system. Starting with R , a set S is obtained keeping in mind that S is relatively small, rich in relevant pages about the query and contains most of the high authorities. HITS algorithm expands the root set R into a base set S by using the algorithm (see Fig. 3.4).

Algorithm: HITS(R)

Input: Root set R ;

Output: Base set S

Let $S = R$

1. For each page $p \in S$, do Steps 3 to 5
 2. Let T be the set of all pages S points to.
 3. Let F be the set of all pages that point to S .
 4. Let $S = S + T + \text{some or all of } F$.
 5. Delete all links with the same domain name.
 6. Return S
-

Fig 3.4: Algorithm to Determine Base Set

2. *Iterative Step:* This step finds hubs and authorities using the output of the sampling step. In this [111], each page is associated with two values: an authority weight a_i , and a hub weight h_i . Pages with a higher a_i value are considered as better authorities and pages with a higher h_i value as better hubs.

Let A be the adjacency matrix of the graph S (output of sampling step), v denotes the authority weight vector and u denotes the hub weight vector. The weights a_i and h_i of all the nodes in S are dynamically updated as follows:

$$v = (A^t \times u) \tag{3.6}$$

$$u = (A \times v) \tag{3.7}$$

If we consider that the initial weights of the nodes as

$$u_0 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

Then $A^t \times \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$

After applying k steps we get the equations as:

$$v_k = (A^t \times A) \times v_{k-1} \quad (3.6a)$$

$$u_k = (A \times A^t) \times u_{k-1} \quad (3.7a)$$

Example Illustrating Working of HITS: The adjacency matrix of the graph is:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$A^t = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Assume the initial hub weight vector is: $u = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$

We compute the authority weight vector by:

$$v = (A^t \times u)$$

$$v = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 3 \\ 2 \\ 1 \\ 2 \end{bmatrix}$$

Then, the updated hub weight is

$$u = (A \times v)$$

$$u = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 3 \\ 2 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 5 \\ 6 \\ 2 \\ 0 \\ 2 \\ 3 \end{bmatrix}$$

By using (3.6a) and (3.7a), the authority weights and hub weights are iteratively calculated until the values get converged as shown in Table 3.4.

By calculating the above equations iteratively, the page ranks of papers become:

$$\text{HITS (C)} > \text{HITS (F)} > \text{HITS (E)} > \text{HITS (D)} > (\text{HITS (A), HITS (B)})$$

Table 3.4 Iteration Method for HITS

Iterations	PR (A)		PR (B)		PR (C)		PR (D)		PR (E)		PR (F)	
	v	u	v	u	v	u	v	u	v	u	v	u
0	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0.56	0	0.67	0.70	0.22	0.47	0	0.23	0.22	0.47	0.33
2	0	1.6	0	2.55	0.82	0.04	0.23	0	0.35	0.042	0.64	0.51
3	0	0.58	0	0.75	0.73	0.00	0.08	0	0.32	0.005	0.58	0.30
4	0	0.58	0	0.73	0.73	0.001	0.03	0	0.32	0.001	0.59	0.32
5	0	0.58	0	0.73	0.73	0.001	0.03	0	0.32	0.001	0.59	0.32

Limitations of HITS: Following are some constraints of HITS algorithm [108, 110]:

- *Distinction between Hubs and authorities:* It is not easy to distinguish between hubs and authorities because many sites act as hubs as well as authorities.
- *Topic drift:* Sometime HITS may not produce the most relevant documents to the user queries because of equivalent weights.
- *Automatically generated links:* HITS gives equal importance to automatically generated links which may not have relevance for the user query.

3.2.6 PaperRank Algorithm

Zhang Guangqian [112] gave a new ranking method for publications ranking named PaperRank based on Google's PageRank. In this method, publication's rank score is determined on the basis of the reading value and other factors because it considers that the reading value of same papers may be different due to different readers. The reading value of a paper is related to its content, the periodical in which it was published, and the

author of the paper. Thus, this method considers the factors such as content, journal, author, published time etc. in order to measure the reading value of papers. PaperRank of the publication p can be calculated as:

$$PaperRank = BR \times AR \times IF \times D \quad (3.8)$$

where BR represents the base rank, AR denotes the AuthorRank, IF denotes the impact factor of the journal in which it was published and D represents the published time of publication p . Various parameters used in the PaperRank calculation are explained below.

BaseRank: The BaseRank (BR) calculates the rank of the publication by using the PageRank algorithm. It considers the quoted time of cited publication and the importance of the citing publication. The BaseRank formula is given as:

$$BR(u) = c \sum_{v \in B(u)} \frac{BR(v)}{N_v} \quad (3.9)$$

where u represents a publication, $B(u)$ is the set of citations that point to u , $BR(u)$ and $BR(v)$ are rank scores of publications u and v respectively, N_v denotes the number of publications cited by publication v (i.e. number of references), c is a factor used for normalization.

AuthorRank: This parameter assumes that if paper A is cited by paper B and C at the same time, then, being cited by paper B authored by a popular and prestigious author contributes more to the Rank value of A than being cited by paper C with an unimportant author. Thus, it calculates the AuthorRank by considering the authors' contribution in a certain academic field. The AuthorRank can be computed by using an author citation network [113] which is a directed and weighted graph where nodes represent authors, edges represent citing relationships from author A to author B, and edge weights represent the number of times that author A cites author B. The AuthorRank can be calculated as:

$$AR(a) = d \sum_{b \in B(a)} \frac{AR(b)}{N_b} \quad (3.10)$$

where a represents an author, $B(a)$ is the set of author's "citing" author a , $AR(a)$ and $AR(b)$ are AuthorRank of author's a and b respectively, N_b denotes the number of authors cited by author b , d is a normalization factor.

Impact Factor of Journal: This parameter assumes that if paper A is cited by paper B and C , and paper B was published in the core journal, and paper C was from unimportant journal, then the vote from paper B to A contributes more rank value to paper A than a vote from paper C to paper A . Thus, it considers the impact factor of journal to represent the weight of each journal. The formula for calculating the impact factor of the journal is defined as follow:

$$IF(j) = \frac{C}{A} \quad (3.11)$$

where $IF(j)$ represents the impact factor of journal j , A denotes the total number of papers published in journal j in the previous two years, and C denotes the quoted times of papers in the current year.

Published Time: This parameter considers the time of the publication. It assumes that sometimes a recently published paper having only one or two citations due to small time window may be important to reader in a certain field. Thus, it introduces the time factor D as follow:

$$D(p) = \frac{(t - \min\{T(k)\} + 1)}{(\max\{T(k)\} + 1)} \quad (3.12)$$

where $D(p)$ represents time factor of paper p , t is the year in which p was published, $B(p)$ denotes the set of all the papers, T is a $n \times 1$ matrix composed by all the years in which all the papers were published, and n is the total number of all the papers.

Limitations of PaperRank: Researchers have shown that scientific publications naturally form a network on the basis of citation relationships. This algorithm can do well for the direct relationships i.e. citation and cited relationships, but it may not adequately reflect the lineage of scientific works. In such scenario, counting the indirect citation, indirect co-citation, and indirect co-reference, which are feasible in the Web environment may be considered.

3.2.7 Popularity and Similarity based PageRank Algorithm (PSPR)

Phyu Thwe [114] proposed a PageRank like algorithm for conducting a web page access prediction named as *Popularity and Similarity Based Page Rank Algorithm (PSPR)*. This

method highlights an improvement in the prediction of web page access by a user [115]. It is based on Web Usage Mining and processes the web server log files to analyze the user's browsing pattern for predicting user's next click. This method ranks the result list of a search engine by taking into consideration the popularity and similarity among web pages as well as the user's navigation behavior pattern.

PSPR functions in two major steps:

1. *Build Markov Model*: In this step, Markov model [114, 115] is used for predicting the behavior of a web user. It is the most widely used web usage mining algorithm for modeling sequences or processes of browsing behavior of a user using finite-state structure. This model takes web pages in the sequence accessed by a user as input parameter and output a model that predicts the user next access/click. Let us assume P be a set of web pages in a web site, P can be written as $P = \{p_1, p_2 \dots p_n\}$, W be a user session of a website. Then, the probability of visiting the next page p by the user is denoted by conditional probability $P = (p_i/W)$. Assuming that i number of pages has already been visited by the user. From here, it can be said that the prediction of next page access does not depend on all the pages in a web session rather can be restricted to small number of k pages. The number k also marks the order of the Markov model. Thus, it can be judged that the web page p_{i+1} will be accessed next using (3.13),

$$P_{i+1} = \operatorname{argmax}_{p \in P} \{P(P_{i+1} = p | p_i, p_{i+1}, \dots, p_{i-(k-1)})\} \quad (3.13)$$

2. *Similarity Calculation*: The popularity of page and transitions plus similarity among the web pages is determined to calculate the importance of the web pages. Similarity is computed based on the contents of the page URL. Following steps are taken in this method:

- Select the URLs of the two pages so as to calculate similarity among them.
- The URLs are sorted in a string array being separated by a special character '/' and their length is calculated.
- Weights are assigned to each array starting from the longest array to the smallest one.

- The matching substrings are identified and their corresponding weights are added and the sum is divided by the total weight to give the similarity measure between the two.

The similarity of two web pages lies between 0.0 and 1.0. If similarity comes out to be 1, it indicates that the two web pages are exactly same. But, if it comes out to be 0, then it is concluded that the web pages are totally different.

Example Illustrating Working of PSPR:

Building Markov model: Let us assume a sample web session of any website as shown in Table 3.5 for building Markov model where Session ID represents the different users and Transitions represents the sequence of pages access by particular user.

Table 3.5 Web Session for a Website

Session ID	Transitions
ID1	C, B, A
ID2	D, E, B, A, E, D
ID3	A, D, E, B, D
ID4	A, D, B, E, C

Next, 1^{st} order Transition Probability Matrix (TPM) (i.e. first order Markov Model) is evaluated, where each state is composed of only single page as depicted in Table 3.6.

Table 3.6. 1st Order Transition Probability Matrix

	A	B	C	D	E
s1=A	0	0	0	2	1
s2=B	2	0	0	1	1
s3=C	0	1	0	0	0
s4=D	0	1	0	0	2
s5=E	0	2	1	1	0

Then, second-order Markov model is evaluated. In this each state will be composed of two web pages and this is decided by the entries in the first-order TPM as shown in Table 3.7 and so on.

Table 3.7 2nd Order Transition Probability Matrix

	A	B	C	D	E
{A,D}	0	1	0	0	1
{A,E}	0	0	0	1	0
{B,A}	0	0	0	0	1
{B,D}	0	0	0	0	0
{B,E}	0	0	1	0	0
{C,B}	1	0	0	0	0
{D,B}	0	0	0	0	1
{D,E}	0	2	0	0	0
{E,B}	1	0	0	1	0
{E,C}	0	0	0	0	0
{E,D}	0	0	0	0	0

This transition probability matrix can be now used to predict the next click for the given session. For example, consider a user’s navigation sequence as **D** → **E** → ? To predict the next page after D and E, firstly the state {D, E} is identified in the second-order TPM and then the page with highest probability is selected. Here, B has the highest probability among rest of the pages as seen from Table 3.7.

Therefore, **D** → **E** → **B** is obtained.

Similarity Calculation: Consider two pages A and B with their respective page URLs as shown in Table 3.8. Thus, the similarity of the two pages is calculated by using the steps as described above and it comes out to be $(4+2+1) / (4+3+2+1) = 0.7$. It indicates that the pages are somewhere similar but not exactly same.

Table3.8 Example of Similarity Calculation

Page	Page URL
A	/project/creators/order-23/madeasy.html
B	/project/creators/artificial/madeasy.html

Advantages and Limitations of PSPR: The main advantage of this method is that it improves the prediction of web page access by analyzing web users' navigational patterns. It can be applied to any web site’s navigational graph for improving browsing orders. But, this method fails to predict directly one more step ahead.

3.2.8 SIMRANK: PageRank Approach Based on Similarity Measure

Shaojie Qiao et. al [116] proposed a better and promising approach to rank the query results of web pages based on similarity measure from the vector space model named as SimRank. This method computes the similarity of pages and applies it to partition a web database into several web social networks (WSNs). This method utilizes the concept of social annotations [117] named as SimRank. The web annotators associate some set of textual content with every web page so as to provide a prior knowledge regarding the web page to the web user without reading the internal contents of that page. In other words, they provide a brief overview about the web page and thus make the user's navigation fruitful. These set of textual contents are known as annotations. The annotations are parsed contents holding the important keywords of a web page. As seen, traditional algorithms do not take into account the impact of content of web pages. They only employ the link structure of web pages to determine the importance of web pages. But, the contents of a web page, which is the required information a user is looking for, could provide a better accuracy in ranking the result list. Thus, this method considers the similarity measure from vector space model to compute the rank of pages. It also improves the traditional PageRank [10, 36] algorithm by taking into account the relevance of page to a given query.

SimRank works in the following manner:

- First, it computes similarity among the web pages of the complete web database.
- Then, it uses the similarity measure as the distance between the pages and apply k-means algorithm [118] to form clusters with pages holding similar contents, and
- Finally, it computes the similarity with respect to the query and assigns a relevance score to each web page. But, this method has issue that its efficiency gets affected by the capabilities of the web crawler being utilized.

The term frequency of a term t_i in the page d_j is calculated by using (3.14),

$$tf_{ij} = \frac{f_{ij}}{\max\{f_1, f_2, \dots, f_{|V|}\}} \quad (3.14)$$

where f_{ij} denotes the frequency of the term t_i in the page d_j and $|V|$ is the vocabulary size.

The inverse document frequency of term t_i is given by using (3.15),

$$\text{idf}_i = \log\left(\frac{N}{df_i}\right) \quad (3.15)$$

where N is the total number of web pages in the web database, df_i denotes the number of web pages in which the term t_i appears atleast once.

Now, the overall term weight is computed as in (3.16):

$$w_{ij} = \left\{ 0.5 + \frac{0.5 \times f_{ij}}{\max\{f_1, f_2, \dots, f_{|V|}\}} \right\} \times \log \frac{N+1}{df_i} \quad (3.16)$$

The similarity measure of a query $Q = \{t_1, t_2, \dots, t_n\}$ and a page p_j denoted as $p_j = \{w_{1j}, w_{2j}, \dots, w_{nj}\}$ where n is the number of terms in the query. So, similarity between two pages p_a and p_b is computed by using (3.17):

$$\text{Sim}(p_a, p_b) = \frac{\sum_{i=1}^n w_{ipa} \times w_{ipb}}{\sum_{i=1}^n w_{ipa}^2 + \sum_{i=1}^n w_{ipb}^2 - \sum_{i=1}^n w_{ipa} \times w_{ipb}} \quad (3.17)$$

Example Illustrating Working of SimRank: To illustrate the working of SimRank, let us consider two papers with their contents as shown below,

$P1$ = The project objectives are laid down as per the required project.

$P2$ = Organisation signs a project yesterday.

Let the query entered is $Q = \{\text{project}\}$

Based on the equations (3.14), (3.15), (3.16) and (3.17), the values obtained are:

$$\text{tf}_{p1} = \frac{2}{11} = 0.181 \quad \text{tf}_{p2} = \frac{1}{5} = 0.2$$

$$\text{idf}_{p1} = \frac{4}{2} = 2 \quad \text{idf}_{p2} = \frac{4}{2} = 2$$

$$w_{p1,p2} = (0.5 + 0.5 \times 0.181) \times 2 = 1.181$$

$$w_{p2,p1} = (0.5 + 0.5 \times 0.2) \times 2 = 1.2$$

$$\text{Sim}(p_1, p_2) = \frac{1.181 \times 1.2}{1.181^2 + 1.2^2 - (1.181 \times 1.2)} = 0.99$$

Thus, it shows that the contents of both the papers are relevant as per the query entered. Hence, SimRank judges with better accuracy about the papers against the content interested to the user.

Advantages and Limitations of SimRank: The main advantage of this method is that it uses similarity measures to effectively cluster and score the publications. This method uses k-means clustering approach to divide a web database into several WSNs as well as clean up the unrelated pages that can help reduce the cost of computation. But, this method has issue that its efficiency gets affected by the capabilities of the web crawler being utilized.

3.2.9 Page Ranking using Social Annotation based on Language Model

Kunmei Wen et. al. [119] proposed an extension to SimRank named *optimizing the results with social annotations based on a language model*. This method uses social annotations to re-rank search results. This method uses the combination of two ranking strategies:

- (a) Query-annotation similarity, and
- (b) Query-document similarity in order to optimize retrieval ranking method.

This method works in the following phases:

- To build the statistical language model of social annotation.
- Calculated the similarity among query and annotation using the language model.
- Initial results of a search engine are re-ranked on the basis of combined score of both the similarity measure.

Statistical language model: In this method, the input parameters considered for constructing a language model [120] are as:

- Set of K initial search results denoted as $D = \{(R_1, A_1) \dots (R_k, A_k)\}$ produced by a search engine where R_k denotes the page and A_k denotes a set of annotations against a specific R_k .
- Set of social annotations in the top K initial search results (also refer as a temporary corpus) denoted as $V_A = \{W_j \mid j = 1 \dots L\}$ where L denotes the size and W_j is a social annotation.
- Set of the social annotations of a specific page denoted as $A_i = \{a_i \in V \mid i = 1, \dots, n\}$

The steps involved in the language model construction include:

- Identify the annotations associated with the web pages and initialize the set A_k accordingly.
- Derive temporary corpus (or the collection of all the social annotations) from the K initial search results.
- Calculate the probability of a term denoted by w_i in the set of annotations A_i for a specific web page using the formula as shown in (3.18),

$$P(w_j|A_i) = \frac{C(w_j, A_i) + 1}{\sum_w C(w, A_i) + L} \quad (3.18)$$

- Thus, it results in the K language models of the annotations for top K initial results.

Query-annotation similarity: User enters the query in the form of keywords, therefore a query can be denoted as $Q = \{q_1, q_2, \dots, q_m\}$ where q_i refers to the keywords or corpus. The probability of the existence or generation of a specific query Q in A_i 's language model is represented as $P(Q|A_i)$. This is referred as probability of query generating. The similarity computation between query and annotations involves the following steps:

- Firstly, the probability of terms appearing in specific annotation is derived from the language model of social annotation.
- A weight is assigned on the similarity measure between query and social annotation and the results are stored.
- The frequency count of a term w in the given query Q is represented as $C(w, Q)$ is taken into account to contribute in similarity score.
- Similarity weight between query and annotation is calculated by using (3.19),

$$P(Q|A_i) = \prod_{w \in Q} P(w|A_i)^{C(w, Q)} \quad (3.19)$$

Final Rank Score: This method finally calculates the rank score of paper by integrating the query-annotation similarity denoted by $P(Q/R)$ and query-document similarity denoted by $P(Q/A)$. The combined weighted rank score is calculated by using (3.20),

$$Score_i = \alpha \times P(Q|R_i) + \beta \times P(Q|A_i) \quad (3.20)$$

where α and β are weights determined experimentally and satisfy $\alpha + \beta = 1$.

Advantages and Limitations: This method uses the concept of annotations which are used as a brief summary for a publication. By using this approach the results are optimized and the newly formed search list is more accurate. But sometimes these annotations contain incomplete and unrelated terms. Such annotations are considered as sparse in nature.

3.2.10 Comparison Study

Based on the literature analysis, a comparison of some of various ranking algorithms is shown in Table 3.9 and Table 3.10. Comparison is done on the basis of some parameters such as main technique used, methodology, input parameters, relevancy, quality of results advantages and limitations. Here N denotes the number of papers. A typical digital library search engine should ranking techniques based on the specific needs of the users. After going through exhaustive analysis of the ranking algorithm [164, 165], it is concluded that existing techniques have limitations in terms of response time, accuracy of results and relevancy of results. Thus, there is scope to propose ranking algorithm which should meet out these challenges efficiently.

The next section describes an introduction to the Document Clustering techniques used by digital library search engines.

3.3 WEB DOCUMENT CLUSTERING

The process of grouping a set of physical or abstract objects into classes of similar objects is called “cluster analysis” or “clustering” [121, 122]. It is an unsupervised learning technique and has been widely used in numerous applications including market research, data analysis and image processing. In the context of document clustering [118, 123, 124], objects are replaced by web documents and are grouped together based upon some measure like similarity of content or of hyperlinked structure. As most of the digital library search engines return a large and unmanageable list of documents containing the user specified query keywords, finding the user required documents from such a large list is usually tedious, often impossible. As a solution, the digital library search engines could

Table 3.9 Comparison of Various Page Ranking Algorithms

Algorithms → Measures ↓	PaperRank [117]	PSPR [114, 115]	SimRank [116, 117]	Social Annotation based on language model [119, 120]
Main Technique used	Web Structure Mining, Web content Mining	Web Usage Mining, Web structure mining	Web Content Mining	Web content mining
Description	Computes new score of the top 'n' pages. Pages returned are more relevant.	The search result list is ranked based on Markov model output and frequency of transition and similarity of papers.	Papers are ranked according to the content similarity rather than the link structure of the pages.	Result are ranked based on the weighted scored determined by calculating similarity score between query and annotation as well as query and document
Input Parameters	Backlinks, authors, Impact factor, time of publish.	Web sessions (Sequence of pages accessed).	Papers and query contents.	Initial search result list, set of tags and papers.
Complexity	$O(\log N)$	$O(N)$, where N denotes the number of pages (or states).	$O(N^2)$ where N denotes number of papers.	$O(K*L)$ where K denotes size of the K initial result list and L denotes size of the temporary corpus.
Relevancy	More	More relevant than traditional PageRank Algorithm.	Results obtained are relevant than the traditional PageRank and other extensions of PageRank.	More relevant results than in SimRank approach.
Quality of results	High	Markov models are highly vulnerable to the data set being used.	Increased efficiency and accuracy in ranking of pages in result list.	This method highly optimizes the initial search results that use only query-document similarity.
Importance	The pages are sorted according to the importance of citations, author journal.	Improves the prediction of web page access & can be applied to any web site's navigational graph for improving browsing orders.	Effectively analyze pages or documents with little contextual information.	It optimizes the ranking of initial results by integrating query-annotation similarity with query-document similarity.
Limitations	Extra calculations to find the author ranking and time impact of citations.	It fails to predict directly one more step ahead.	Its efficiency gets affected by the capabilities of the web crawler being utilized.	In some web pages the annotations may be sparse and incomplete; hence it creates a gap between annotations and queries.

N*= Number of Paper

Table 3.10 Comparison of Various Page Ranking Algorithms

Algorithm →	CC [103]	TDCC [104]	PageRank [10, 36]	Popularity Weighted PageRank [107]	HITS [108, 109]
Measures ↓					
Main Technique Used	Web Structure Mining	Web Structure Mining	Web Structure Mining	Web Structure Mining	Web Structure Mining, Web content Mining
Description	Results are sorted based on number of incoming citations.	Results are sorted based on time dynamics of the citation graph i.e. age of the citations	Computes scores at indexing time. Results are sorted by taking into account the importance of citing papers.	Results are sorted according to weighted citations as well as popularity factor of publication venue of paper.	Computes hub and authority scores of 'n' highly relevant pages on the fly. Relevant as well as important pages are returned.
I/P parameters	Backlinks	Backlinks, publishing time of paper	Backlinks	Backlinks, Publication venue	Backlinks, forward links, Content
Working Levels	1	1	N	N	< N
Complexity	O(N)	O(N ²)	O(log N)	O(MN)	<O(log N)
Relevancy	Less	Less(More than CC)	Less(more than CC, TDCC)	More (less than PaperRank)	More (less than PaperRank)
Quality of Results	Less	Higher than CC	Medium	Higher than PR	Less
Importance	Simplicity of computation.	This method considers the freshness of citations by differentiating between the old and new citations.	It statistically analyses whole citation graph at once. It captures not just quantity, but also quality of citing papers.	This method overcome the limitation of impact factor and considers the popularity of publication venue.	This method provides good results by considering Hubs and Authorizes scores and also considers the content of the paper.
Limitations	Unweighted ranking i.e. it treats all the citations equally.	It does not take into consideration the different importance of each citation.	Results come at the time of indexing. Results are sorted based on importance of citations.	It does not take into account the time of publication.	Topic drift and efficiency problem.

N*= Number of Paper, M= Average Citations of a Paper

apply some tools to group a set of documents returned in response to a query with the aim of finding meaningful clusters, rather than a list of ranked documents.

3.3.1 Major Categories of Clustering

In general, the major clustering methods can be classified into the following categories.

Partitioning methods: Given a database of n objects, a partitioning method constructs K ($K \leq n$) partitions of the data, where each partition represents a cluster. The clusters satisfy the following requirements:

- Each group must contain at least one object, and
- Each object must belong to exactly one group.

The general criterion of a good partitioning is that objects in the same cluster are “closed” or related to each other, whereas objects of different clusters are “far apart” or very different. K-means, K-medoids [118] are few popular algorithms based on partitioning method. The K-means algorithm is given in Fig. 3.5.

Algorithm: K-means(D, k)

Input: A dataset D , a user specified number k

Output: k clusters

```
{
  Randomly Initialize cluster centroids;
  While not convergent
  {
    For each object  $o$  in  $D$  do
      Find the cluster  $c$  whose centroid is most close to  $o$ ;
      Allocate  $o$  to  $c$ 
    For each cluster  $c$  do
      Recalculate the centroids of  $c$  based on the objects allocated to  $c$ ;
  }
}
```

Fig. 3.5 The K-means Algorithm

The key idea of K-means is simple and is as follows: In the beginning, the number of clusters i.e k is determined. Then, the algorithm randomly assumes the centroids (or centers) of these K clusters. If the number of objects is less than the number of clusters, then each object is treated as the centroid of a cluster and allocated a cluster number. Otherwise, the algorithm computes the distance (i.e., Euclidean distance)

between each object and all centroids to get the minimum distance. Because the location of the real centroid is unknown during the process, the algorithm needs to revise the centroid location with regard to the updated information. After updating the values of the centroids, all the objects are reallocated to the K clusters. The process is repeated until the assignment of objects to clusters ceases to change, or when the centroids move by negligible distances in successive iterations.

Hierarchical methods: Hierarchical clustering [126] constructs a hierarchy of clusters that can be illustrated in a tree structure which is also known as a dendrogram. Each node of the dendrogram, including the root, represents a cluster and the parent-child relationship among them enables to explore different levels of clustering granularity.

There are mainly two types of algorithms for hierarchical clustering:

- Agglomerative
- Divisive

The *Agglomerative approach*, also called the bottom up approach, starts with each object forming a separate group. It successively merges the objects or groups that are closed to one another, until all of the groups are merged into one, or until a termination condition holds. The Hierarchical Agglomerative Clustering (HAC) algorithm is presented in Fig. 3.6.

Algorithm: HAC(D)

Input: A Dataset D

Output: A hierarchy tree of clusters

```

{
  Allocate each object  $o$  in  $D$  as a single cluster;
  Let  $C$  be the set of the clusters;
  While  $|C| > 1$  do
    For all clusters  $X, Y \in C$  do
      Compute the between-cluster similarity  $S(X, Y)$ ;
       $Z = X \cup Y$ , where  $S(X, Y)$  is the minimum;
      Remove  $X$  and  $Y$  from  $C$ ;
       $C = C \cup Z$ ;
}

```

Fig. 3.6 The Hierarchical Agglomerative Clustering (HAC) Algorithm

The *Divisive approach*, also called the top-down approach, starts with all of the objects in the same cluster. In successive iterations, a cluster is split up into smaller clusters, until eventually each object is in one cluster, or until a termination condition holds.

Density based Methods: The partitioning and hierarchical methods can find only spherical-shaped clusters and encounter difficulty at discovering clusters of arbitrary shapes and sizes. Other clustering methods have been developed [125] based on notion of density, wherein if a number of data objects in the "neighborhood" exceeds some threshold, then they are grouped together. It means, for each data point within a given cluster, the neighborhood of a given radius has to contain at least a minimum number of points. The DBCCOM [126] algorithm and its extension OPTICS are typical density-based methods that perform clustering according to a density-based connectivity analysis.

3.3.2 Similarity Measures

The key problem underlying document clustering is to determine an adequate similarity function so that truly similar documents can be grouped together using a clustering algorithm. In this section, some similarity functions have been discussed, which are used for finding similarity between two documents, two queries, or one document and one query. There are different ways to represent document contents: keywords, words in their order, and phrases. They provide different measures of similarity, each with its own useful information.

Document Representation: The depiction of a set of documents as vectors in a common vector space is known as the Vector Space Model (VSM) [127]. This representation is used for many IR operations ranging from scoring documents on a query, document classification and document clustering. A document vector captures the relative importance of the terms in a document, wherein each term is assigned a weight depending on its number of occurrences in the document.

In VSM, each document can be viewed as a vector with one component corresponding to each term in the dictionary. Let Y be the set of terms in the document collection whose

size is give by n . For each term y_i there exist a vector y_i in the vector space that represents it. It then considers the set of all term vectors $\{y_i\}$ ($1 \leq i \leq n$) to be the generating set of the vector space, thus the space basis. A document vector x_i is given by:

$$x_i = (y_{i,1}, y_{i,2}, \dots \dots y_{i,n}) \quad (3.20)$$

If each x_i (for $i = 1 \dots m$) represents a document vector of the collection, then there exists a linear combination of the term vectors $\{y_i\}$ which represents each x_i in the vector space. Once a vector has been defined for each document in the corpus, they can be represented by using a document-by-term matrix A in which each row represents a document and each column represents a term in the corpus. The resulting document-by-term matrix A whose element A_{ij} denotes the occurrence of a term j in document i as shown below:

$$A = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mn} \end{bmatrix} \quad (3.21)$$

There are number of schemes to assign the weight to terms in a document. The simplest approach is referred to as *Term Frequency* which assigns the weight to be equal to the number of occurrences of term t in document d . It is denoted $TF_{i,d}$ with the subscripts denoting the term and the document in order.

There are number of similarity measures have been proposed in literature, some of which is described as:

a) Cosine Similarity: When documents are represented as term vectors, the similarity of two documents corresponds to the correlation between the vectors. This is quantified as the cosine of the angle between vectors, that is, the so-called cosine similarity [128, 130].

Cosine Similarity measure between two documents d_i and d_j i.e. $Sim_{Cosine}(d_i, d_j)$ is given by (3.22):

$$Sim_{Cosine}(d_i, d_j) = \frac{d_i \cdot d_j}{\|d_i\| \|d_j\|} = \frac{\sum_{k=1}^n w_{i,k} w_{j,k}}{\sqrt{\sum_{k=1}^n w_{i,k}^2 \sum_{k=1}^n w_{j,k}^2}} \quad (3.22)$$

where k denotes the size of documents.

This similarity measure is simple and very efficient to evaluate. This measure gives the value in between [0, 1]. But, it does not consider the variation in the ratings given to the documents by the different users for the computation.

b) Jaccard Coefficient: It is a statistic used for comparing the similarity and diversity of sample sets. The Jaccard coefficient [127, 130] measures the similarity between finite set of sample. It is defined as the size of the intersection divided by the size of the union of the sample sets i.e. the number of shared terms present in documents divided by the number of all unique terms present in both documents.

Jaccard Coefficient between two documents d_i and d_j i.e. $Sim_{Jaccard}(d_i, d_j)$ is given by (3.23):

$$Sim_{Jaccard}(d_i, d_j) = \frac{|d_i \cap d_j|}{|d_i \cup d_j|} = \frac{\sum_{k=1}^n w_{i,k} w_{j,k}}{\sum_{k=1}^n w_{i,k}^2 + \sum_{k=1}^n w_{j,k}^2 - \sum_{k=1}^n (w_{i,k} w_{j,k})} \quad (3.23)$$

where k denotes the size of documents.

The main disadvantage of this measure is that it can't verify the existence of duplicate samples i.e. over-typed words were neglected in the measurement of the similarity.

c) Dice Coefficient: It is defined as two times the number of terms which are common in the compared documents and divided by the total number of terms present in both documents [128, 129].

Dice Coefficient between two documents d_i and d_j i.e. $Sim_{Dice}(d_i, d_j)$ is given by (3.24):

$$Sim_{Dice}(d_i, d_j) = 2 \frac{|d_i \cap d_j|}{|d_i| + |d_j|} = 2 \frac{\sum_{k=1}^n w_{i,k} w_{j,k}}{\sum_{k=1}^n w_{i,k}^2 + \sum_{k=1}^n w_{j,k}^2} \quad (3.24)$$

where k denotes the size of documents.

d) Overlap Coefficient: It is similar to the Dice's coefficient, but. It is also called as Szymkiewicz-Simpson coefficient [129, 131]. This method considers two strings a full match if one is a subset of the other and it is similar to Dice

Coefficient It measures the overlap between two sets by dividing the size of the intersection by the smaller of the size of the two sets:

Overlap Coefficient between two documents d_i and d_j i.e. $Sim_{Overlap}(d_i, d_j)$ is given by (3.25):

$$Sim_{Overlap}(d_i, d_j) = \frac{|d_i \cap d_j|}{\min(|d_i|, |d_j|)} = \frac{\sum_{k=1}^n w_{i,k} w_{j,k}}{\min(\sum_{k=1}^n w_{i,k}^2, \sum_{k=1}^n w_{j,k}^2)} \quad (3.25)$$

where k denotes the size of documents.

If document d_i is a subset of d_j or Vice-versa, then the overlap coefficient value is 1.

The next section is devoted to the discussion of another technique used in Digital libraries called Document Categorization.

3.4 DOCUMENT CATEGORIZATION

The benefit of digital documents is that they can be computationally analyzed, because a computer program can extract the document text and process it for further analysis. A convenient way of storing also creates the need for a convenient way of retrieval: What is the use of storing documents if they cannot be found? Naturally categorization or classification of documents [132] has been used to make it easier to find relevant information.

Document classification is the task of assigning documents to two or more predefined categories. For example, a news document generated in the Reuters news agency is classified into a number of topics, such as "crude oil", "foreign currency exchange", "acquisition" and so on. If a document can be assigned to more than one category; the process is called multi-category classification. If a document could be assigned to only one category, it is called singular-category classification. Multi-category classification is more common than singular-category classification.

A brief study about some keyword extraction techniques has been discussed.

3.4.1 Keyword Extraction

Keywords play a crucial role in extracting the correct information as per user requirements. Since keyword is the smallest unit which express meaning of entire document , many applications can take advantage of it such as automatic indexing, text summarization, information retrieval, classification, clustering, filtering, cataloging, topic detection and tracking, information visualization, report generation, web searches etc. [133, 134] Existing methods about Automatic Keyword Extraction [135] can be divided into four categories:-

- ***Simple Statistical Approach:*** It comprises simple methods which do not require any training data. These types of approaches use statistical information to identify the keywords in the document. The statistical methods include word frequency, term-frequency [136], term frequency-inverse document frequency (TF*IDF) [137], word co-occurrence [136, 138] etc.
- ***Linguistics Approach:*** These approaches use the linguistic features [139] of the words mainly sentences and documents. The linguistic approach includes the lexical analysis, syntactic analysis discourse analysis and so on. In this work, the linguistics approach is employed for keyword extraction.
- ***Machine Learning Approaches:*** Machine Learning approach considers the supervised learning from the examples. They induce a model which is trained on a set of keywords for keyword extraction. These methods require training data, and are often dependent on the domain. This approach includes Naïve Bayes [140], Support Vector Machine [141] etc.
- ***Hybrid Approaches:*** Hybrid approaches about keyword extraction mainly combine the methods mentioned above or use some heuristic knowledge in the task of keyword extraction, such as the position, length, layout feature of words, html tags around of the words, etc. Various extraction methods discussed are for single document but these can further be applied to multiple documents as per their suitability [142].

Limitations of Existing Keyword Extraction Techniques: A critical look at the available Keyword Extraction Techniques for digital libraries indicates the following limitations

which need to be addressed. Like the term frequency and inverse term frequency (tf-idf) technique can't be applied on single document to extract the keywords. Tf-idf technique can extract important keywords by comparing two or more documents. In all these techniques, a lot of processing is done to extract the keywords by scanning whole document which is very time consuming. There is a need to devise a novel technique for keyword selection and extraction.

In next section, a brief study about some document categorization techniques has been carried out.

3.4.2 Different Categorization Techniques

Major document categorization techniques are decision trees, k-nearest neighbor, Bayesian approaches, neural networks, regression based methods and vector based methods. A brief description of these methods and their relative merits are discussed below:

a) Decision Tree: Decision trees [143, 144 145] are most widely used predictive modeling approaches used in statistics, data mining and machine learning. Here classification is based on the learning of decision trees which consists of a sequence of various decision rules in the form of tree like structure where the nodes represent questions and the leaves represent the corresponding category of documents. This method is easy to interpret for naïve users. But, decision-tree learning is based on heuristic algorithms where decisions are made at each node locally and cannot guarantee to return the global optimal decision tree.

b) K Nearest Neighbor (K-NN): K-NN classifier [143, 144, 145] is a case-based learning algorithm in which the categorization is done by comparing the category frequencies of the k-nearest neighbors. The Euclidean distance or the angle between the feature vectors is computed as a similarity measure between documents. These methods are sometimes called “memory based learning” methods. This method is easy to interpret and robust to noisy training data. But, in some cases, it is biased by value of k i.e. number of clusters.

c) Naïve Bayes (Idiot Bayes) Classifier: It is a supervised learning algorithm which is based on applying Bayes' theorem [140, 149] with the “naive” assumption of independence between every pair of document features. This method is feature independent means the word order is irrelevant. A disadvantage of this method is that they can only process binary feature vectors and, thus, have to abandon possibly relevant information.

d).Neural Networks (perceptrons) Classifier: Neural network [143, 144, 145] is also called artificial neural network is a mathematical model inspired by biological neural networks. It is composed of set of parallel and distributed processing units called neurons [143, 144,145]. These neurons are interconnected by means of unidirectional or bidirectional links by ordering them in layers. This method can handle noisy and contradictory data very well. The main disadvantage of this method is that neural networks are difficult to understand by naïve users and requires high training cost due to high flexibility of neural networks.

e) Support Vector Machines (SVM): This method needs positive training documents as well as negative training documents during the categorization process [143, 144,146, 148]. This method is looking for the decision surface that best separates the positive from the negative examples in the n-dimensional space. The main advantage of this method is its simplicity, interpretability, robustness and flexible performance.

3.4.3 Study of Recent Document Categorization Techniques

A literature survey of various document categorization techniques used by digital library search engines has been done in this section. Few existing document categorization techniques proposed by researchers are discussed below:

a) Publication-level Classification System of Science: A method to classify the publications into research area at individual level of publication instead of at the journal level was proposed by Ludo Waltman and Nees Jan van Eck [150]. This method clustered the publications into research areas based on citation relations. Each publication is assigned to a single research area, and research areas are

organized in a hierarchical structure. The methodology is able to deal with very large numbers of publications. A noteworthy feature of this methodology is its transparency and relative simplicity. But, in this method, value of few numbers of parameters need to be chosen manually. The main limitation of this method is its exclusive reliance on direct citation relations between publications.

b) Automatic classification of scientific papers in PDF: Juan C. Rendón-Miranda et al. [151] proposed a method to classify scientific papers in PDF format according to the first level of the ACM Classification System and then the result is instantiated in document ontology. Once the ontology is populated, it can be used to perform inferences and obtain implicit knowledge from the papers. For document classification, Naïve Bayes classification approach is used.

c) Categorization of multilingual scientific Documents: Jarosław Protasiewicz et al. [152] proposed a three layered classification for multilingual scientific documents. Multilingual means documents containing the text parts in various languages at the same time. The three layers work as:

- (i) A preprocessing layer which generates a Vector Space Model,
- (ii) Monolingual classifiers corresponding to different text parts, and
- (iii) A decision layer which integrates the outputs of all the classifiers and generates the final prediction regarding a target class.

This method states that the classification quality is improved by integrating outputs of all multilingual classifiers that performs separately. But, the main disadvantage of this method is that monolingual classifiers are dependent on a dataset and training algorithms because it gave opposite results when the models are trained by Long-Short-Term memory algorithms.

d) Fast Categorization of Web Documents represented by Graphs: A Hybrid approach to categorize the web documents was proposed by Alex Markov et al. [153] which was built upon both graph and vector space representations. The graph approach provides the ability to capture important structural information hidden in a web document and its HTML tags. This method uses the tags for identification of hyperlinks, title, underlined, or bold text, etc. The document

representation techniques used by this system also gave weightage to the order and combination of words in the text.

3.4.4 Comparison Study

By going through the literature survey of some of existing document categorization techniques, it is concluded that each technique has some advantages and disadvantages. A tabular comparison study is shown in Table 3.11 which compares the techniques on

Table 3.11 Comparison between various Existing Document Categorization Techniques

Indexing Method	Publication-level Classification System of Science [150]	Automatic classification of scientific papers in PDF [151]	Categorization of multilingual scientific Documents [152]	Fast Categorization of Web Documents Represented by Graphs [153]
Main Technique Used	Classify the publications into research area at individual level of publication instead of at the journal level. The publications are clustered into research areas based on citation relations	ACM Classification first level system is used to classify the documents and then instantiated in document ontology.	Three layered classification system for multilingual scientific documents is proposed. It states that the classification quality is improved by integrating outputs of all multilingual classifiers that performs separately.	The Graph-Theoretic web document representation technique is used for categorizing the web documents by giving weightage to tags in the documents.
Type of Categorization	Clustering based on citation relations.	Naïve Bayes classification approach is used.	Multinomial Naïve Bayes method is used.	Vector representation, using the k-Nearest Neighbor (k-NN) classification algorithm
Advantages	Efficiently deal with deal with very large numbers of publications.	It can be used to performed inferences and obtained implicit knowledge from the papers.	It describes the documents sufficiently well and there is no need to introduce more computationally demanding algorithms.	This method captures important structural information hidden in a web document and its HTML tags.
Limitation(s)	Value of few number of parameters need to be chosen manually.	This method is under every phase of methodology used.	Monolingual classifiers are dependent on a dataset and training algorithms.	It is very difficult to be used by naive users .

various parameters such as technique used, type of indexing used, advantages and limitations.

3.5 POSSIBLE APPLICATION AREAS

The concept of document clustering, document categorization and keyword extraction has been widely used by many researchers in optimizing the search and retrieval process of digital library search engines. Some of the identified key areas where these techniques can be utilized are given under:

- ***Building Effective Indexes:*** The document clustering can be utilized in building effective index structures for digital library search engines, which in turn prompts the efficient index searching.
- ***Automatic Query Expansion:*** The extracted information from query categorization can be used as source for automatic query expansion [154, 155]. By categorizing the queries and then recommending the clusters of documents to users, there becomes an opportunity for users to take advantage of category/topic based queries and use the appropriate ones to meet his information need.
- ***Ranking:*** With the rapid growth of digital documents on WWW, the users are becoming more and more dependent on the digital library search engines' ranking schemes [103, 104, 107, 108, 112, 115, 116, 117] to discover more relevant information as per their needs. Typically, users expect the more relevant documents at the top-ranked results, and more often they do not look at the document snippets except in the first few result pages. So, there is a need of ranking schemes which take into account not only the overall page quality and relevance to the query, but also the match with the users' real search intent when they formulate the query.
- ***Better Document Representation:*** By clustering similar documents either by their content or by their category, the results of a search query can be presented to the users in a much better way than the traditional ordered representation. By this way, user can restrict his browsing to particular clusters of his interest. Thus, Information Overkill problem can be abridged and search space can be reduced to

a better scale.

The next section provides a brief summary of the limitations found in the literature survey.

3.6 REVIEW SUMMARY

A critical look at the available literature indicates the following issues, which need to be addressed in building efficient indexing and query processing systems for digital library search engines:

- ***Lack of Efficient Result Representation Techniques:*** In response to user queries, a digital library search engine generally returns a large volume of results generally presented to the user in the form of a ranked list. To search for the desired information, user keeps on sifting between the documents and thus making extra efforts. Some more efficient representation either in the form of clusters or in the form of combination of cluster and ranked representation is actually needed so as to reduce the search space.
- ***Low Precision:*** Most of the digital library search engines depict low precision. User can't browse all the documents one by one, and most documents are irrelevant to the user's interest, they are highlighted and returned by digital library search engine just because these documents possess query keywords. Even, the most relevant documents to users' query words or topic are generally not shown at the top of the search results list. Hence, the time users spend for seeking out the required information from search result list is large.
- ***Irrelevancy of Results:*** The traditional ranking methods employed by the digital library search engines are generally based either on content-oriented or on the link-oriented approaches i.e. it assigns a page score independent of users' query words. Thus, the relation between academic documents and the requirement of a researcher could not be completely matched.
- ***Inefficient Document Organisation Scheme:*** In response to user queries, a digital library search engine generally returns only those documents/publications whose contents are indexed by them. In some cases, there may exist paper p

which is linked to by papers already indexed, but is not indexed by search engines. Is it still possible to meaningfully index p and return it in search results?

- ***Inefficient Retrieving Approach:*** As most of the digital library search engines are keyword based, category or domain of keywords are generally ignored by them. For instance, the topical query “apple” given to a digital library search engine may retrieve the documents related to “apple fruit” as well as “apple computer” together, thereby unnecessarily increasing the search space. Infact, there is a need of optimizing user search by the way of using categorizes or taxonomies so as to restrict it towards the right direction either by building efficient query analyzers or by building efficient retrieving systems.

In subsequent chapters, novel approaches for Crawling, Indexing, Categorization and Document Ranking have been proposed to resolve the mentioned issues.

Chapter IV

FOCUSED CRAWLER TO HARVEST DIGITAL ACADEMIC DOCUMENTS

4.1 GENERAL

The WWW is a huge collection of digital documents wherein every second, a new piece of document is added. Finding relevant academic documents or publications indeed is a protracted task and searching required document without any explicit or implicit knowledge adds more intricacy to the process. Generic crawlers traverse complete web in order to generate indexes which are used later for searching and recommending links to users. This method leads to huge storage space requirements and usually falls short to cope up with the huge volume of digital information present on the Web. Focused crawling in such a scenarios provides a better alternate to generic crawling especially when topic specific and personalized information is required.

Digital libraries which are based on focused crawling of open-access archives (e.g. CiteSeer) often have large volume of missing publications in their collections of archived publications viz. documents of ScienceDirect [42], ACM, Springer and IEEE Explore [41] which require payment of fee to access the desired content. A question arises here- *How do the researchers be able to access these kind of missing documents from digital libraries or How do digital libraries collect or crawl such category of documents?* As a solution to this, an approach of focused crawling has been developed to improve the effectiveness of digital library crawlers.

A detailed discussion on proposed digital library focused crawler is given in the following sections:

4.2 PROPOSED CRAWLING PROCESS OF DIGITAL LIBRARIES

The architecture of proposed focused crawler is depicted in Fig 4.1, which consists of following functional modules:

1. Page Downloader
2. Categorization Process

3. Link Forecasting Process
4. Missing Document Finder Module
5. Aging Process

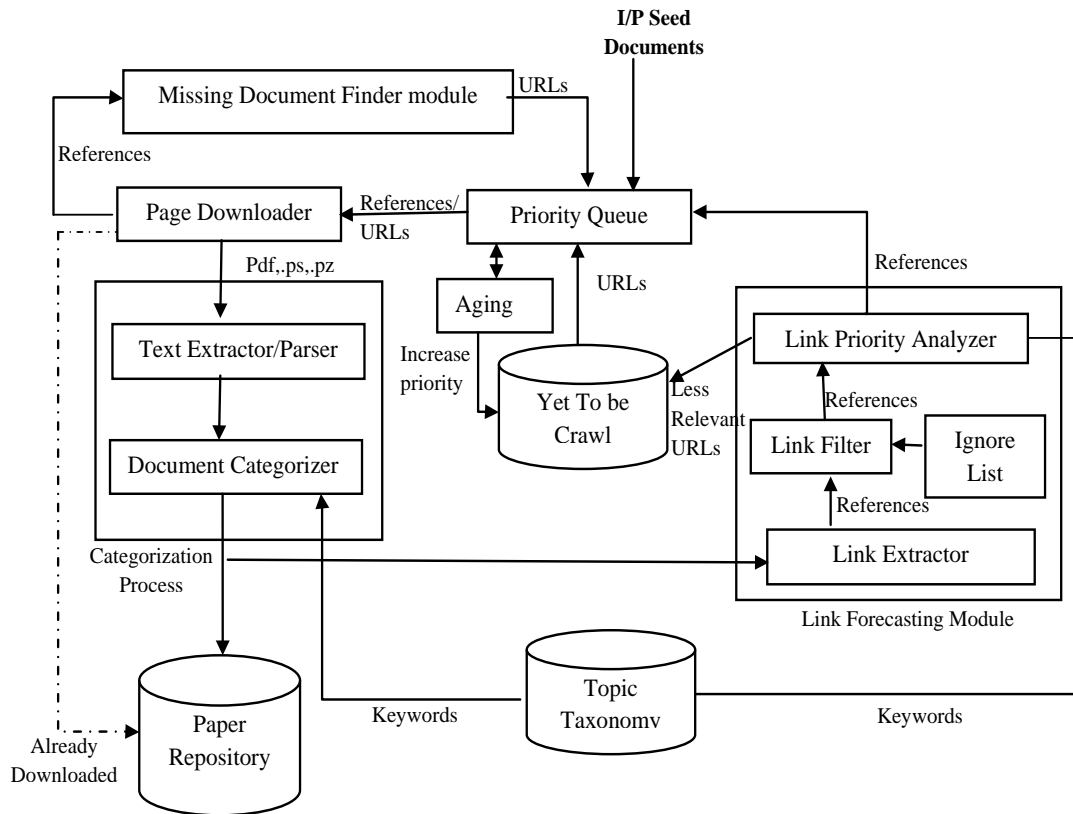


Fig 4.1 Architecture of Proposed Crawling System

When a user inputs a seed document title in the form of query, then *page downloader* fetches the document and downloads it from the internet. If the downloaded document is in pdf/.ps/.pz format, then it is forwarded to *Categorization process*, otherwise sent to *Missing Document Finder Module*. In *Categorization process*, first *text extractor/parser* parses the downloaded document i.e. extracts the information such as keywords, title, author, references in downloaded document etc. and then further forwards it to *Document Categorizer*. This component decides the category of the respective document with respect to *Topic Taxonomy*. The document is saved into *Paper Repository* and also forwarded to *Link Forecasting Module*.

In *Link Forecasting Module*, first *Link Extractor* extracts all the outgoing links (references) from the downloaded document. After that, *Link Filter* component filters the extracted references/URLs and sends all of them to *Link Priority Analyzer*. The

Link Priority Analyzer assigns the appropriate score to unvisited references/URLs. This component only sends the top 10 unvisited references to priority queue and rest unvisited references/URLs is saved to *Yet to be Crawl* temporary database. The *Priority Queue* contains a list of unvisited URLs in the order of their assigned weight. Aging component works at the backend, in order to increase the priority of low priority URLs (URLs in *Yet to be Crawl* database) with the time span.

The working for different functional modules is described below:

4.3 PAGE DOWNLOADER

This component takes the seed document titles provided by the user or references/URLs from the priority queue as an input and download the document in .pdf, .ps or .pz format from the web. Initially, as an input, user provides a list of seed documents (i.e. document titles from different domains/areas) for initiating the crawling process.

Let us take an example, the user inputs the seed document title as: “*Retrieval Evaluation with Incomplete Information*” and the page downloader download the corresponding pdf of document from the web as shown in Fig. 4.2.

But sometimes, the page downloader gets the input seed document title or URL from the priority queue which does not directly downloads the document in pdf. Instead of downloading the pdf document, the URL displays the link or button to download the pdf format of document.

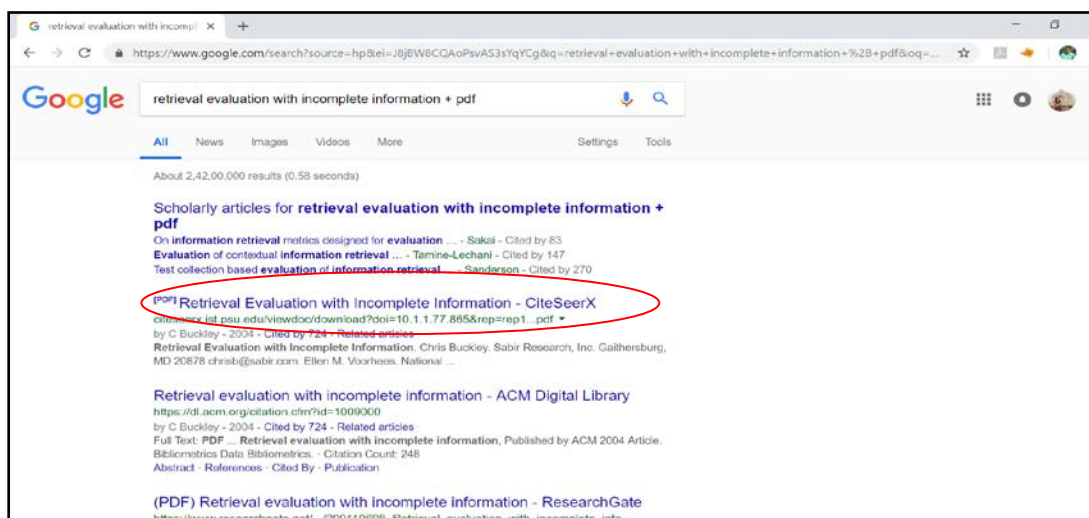
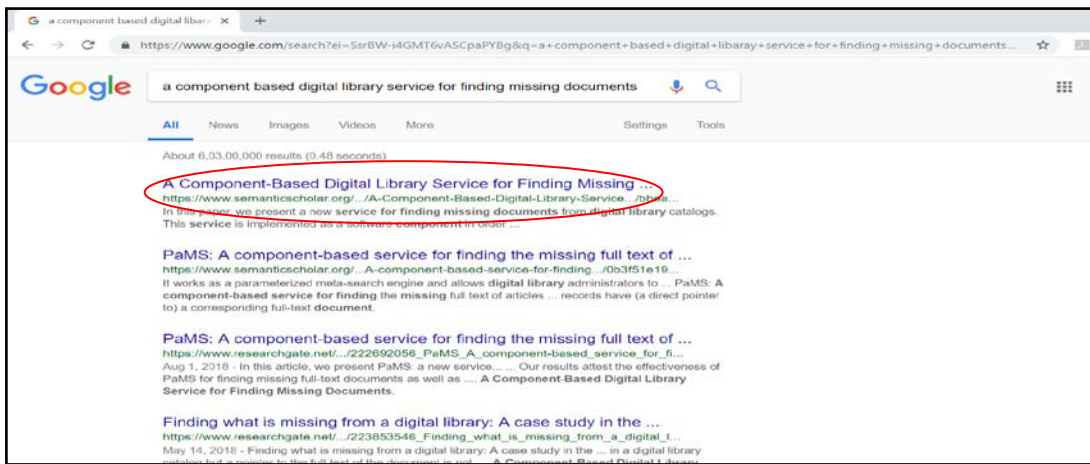


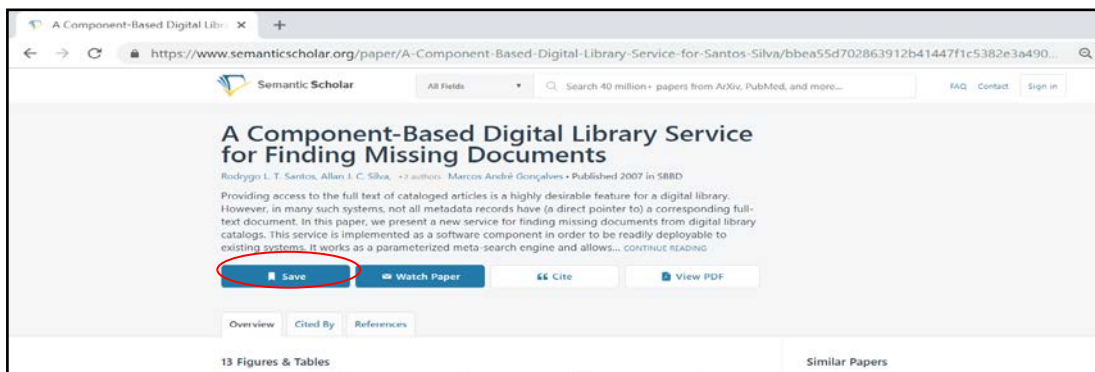
Fig 4.2 Example to Download pdf of Document

For example, if the user inputs the seed document title: “A Component based Digital Library Service for Finding Missing Documents”. Then, the user gets the indirect link to download the pdf format of seed document instead of the direct link to download the same (as shown in Fig 4.3 (a). This indirect link further provides the link or button to download the pdf format of requested document (as shown in Fig 4.3 (b)).

In these cases, this page downloader component also checks for the link or button on the given URL for downloading the document. If it fails to download the pdf format of document, then the URL is forwarded to *missing document finder module* for further processing.



(a)



(b)

Fig 4.3 Example of when Button or Link is given on the Web Page to Download pdf

4.4 CATEGORIZATION PROCESS

In this process, downloaded documents are first parsed and then categorized based upon their research area or category. After going through the detailed literature survey

[150, 151, 152, 153], few limitations were identified in existing document categorization approaches as discussed in chapter II which need to be resolved. To overcome these issues, a document categorization technique has been proposed to categorize the documents into the predefined categories.

The proposed categorization system [165] works on dynamic databases instead of static ones. The system incrementally categorizes the newly uploaded documents into the predefined categories based on various measures. In this system, a novel approach for keyword extraction is used. In this technique, keywords are extracted from research papers/documents by reading bookmarks. It has been assumed here that the bookmarks contain most important keywords of the paper. Due to bookmarks reading, system neither needs to scan the full paper nor requires storing the paper. Thus, bookmarks extraction is used to improve the efficiency of the system in terms of space and time complexity. To understand the working, the detailed process of the proposed categorization system is outlined in Fig. 4.4., where the dashed outline represents the proposed text extraction process. In order to achieve the required task, architecture is divided into two major sub-systems as given below:

- Text Extractor/Parser
- Document Categorizer

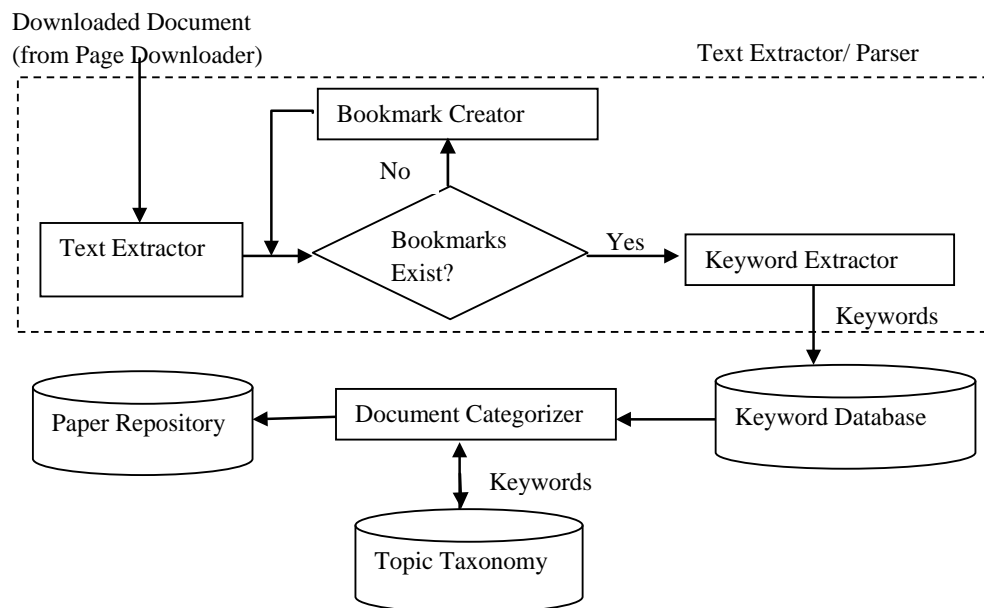


Fig 4.4 Architectural Flow of Categorization System

First, the downloaded document which is forwarded by page downloader is parsed by the *Text Extractor or Parser*. In this categorization system, *Bookmark Creator* is an inactive module. This module becomes active only when the downloaded and parsed document does not contain bookmarks. Once the bookmarks have been created by this module, *Keyword Extractor* becomes functional and extracts keywords from bookmarks and applies stop word removal techniques on extracted keywords. These keywords are saved in *Keyword Database*. Now, the *Document Categorizer* decides the category of the paper by considering the pre-defined categories stored in the form of *Topic Taxonomy* and saves the paper in main database i.e. *Paper Repository*. An algorithm for categorization process is described in Fig 4.5. The description of the various functions used in the algorithm is given below:

Algorithm: Categorizer(P, CKD, KD)

I/P: Paper P, Category keyword Database CKD (i.e. Topic Taxonomy) , Keyword Database KD
O/P: Updated CKD, Updated Paper Repository CDD

```

{
    bookmarks ← Bookmark (P)                //Extract bookmarks of a paper
    If(bookmarks == Null)
    {
        Create_bookmark(P)                  // bookmarks creation
        bookmarks ← Bookmark(P)
    }
    keywords ← Key_extract(bookmarks)        //Extract keywords from bookmarks
    CDD ← Document_category(P, keywords, CKD)
    CKD ← Incre_key_filter(keywords, CKD)
} //end of

```

Fig 4.5 Algorithm for Categorization Process.

1. *Bookmark ()*: Check whether bookmarks exist in paper or not. If they do not exist, then return null otherwise, save bookmarks in *bookmarks* variable.
2. *Create_bookmark()* : It creates the bookmarks of the paper and update existing paper.
3. *Key_extract()* : This function extracts tokens from data passed as parameter; removes stop words and performs stemming function on keywords.
4. *Document_category()* : Decides the category of the research paper based on keywords of the paper and keywords existing in different categories of topic taxonomy. Then, it uploads the paper in respective category.
5. *Incre_Key_filter()* : This function is part of topic taxonomy component which incrementally updates the keywords of categories stored in topic taxonomies. Detailed

working of Increment key filter is discussed in Section 4.4.6.

The detailed description of various modules and data structures involved in this process is explained below:

4.4.1 Text Extractor

This component takes the downloaded document in pdf format as an input and extracts all the text of the document like title, authors, keywords, bookmarks, references etc. This extracted information is forwarded to *Document categorizer*. (Subsection 4.4.4)

4.4.2 Bookmark Creator

This module comes into play when bookmarks are not present in the newly downloaded paper/ document. It creates the bookmarks and upgrades the paper. This module scans the whole paper and uses following principles or rules to create bookmarks:

- i. Words emphasized by application of bold, italic or underlined fonts,
- ii. Using headings of the research paper,
- iii. Words typed or written in upper case,
- iv. The size of the font applied,
- v. Normalized Sentence Length, which is the ratio of number of words occurring in sentence over number of words occurring in the longest sentence of the document.

After this, the selected paper is upgraded with bookmarks and processed by the next module called *keyword extractor*.

4.4.3 Keywords Extractor

This module processes only those papers which are having bookmarks. So, after checking the bookmarks in the previous step, this module extracts the bookmarks from the paper. Then, finds the keywords from these bookmarks and applies stop word removal. After this, stemming algorithm is applied on each term of text files by *porter.java* which uses Porter's Stemming algorithm [156, 157]. There are some other stemmers that are available as- Lovins stemmer [158], Dawson Stemmer [159]. But

Porter's stemmer is the prevalent stemmer in Information Retrieval and Language Processing problems because its performance is pretty good, hence is also used in present context.

On the basis of the frequency of the keywords, it chooses top ten frequent keywords and stores them in the categorized keyword database.

4.4.4 Document Categorizer

In this module, the category of each downloaded document is decided and the document in turn is saved with that category in the *Paper Repository* database. This module incrementally categorizes the newly uploaded or downloaded documents into the predefined categories based on different measures. First, it extracts the keywords of the paper under processing from the keyword database. Then, compare these keywords with the keywords of pre-defined categories at top level by using cosine similarity measure [128, 129, 130]. The category having highest cosine similarity value amongst all is the most relevant category for the selected article/paper. After this, the resultant category is explored further. This process is repeated until the most relevant category at the lowest level is found. At last, document categorizer uploads the paper in the resultant category. For keyword comparison purposes, cosine similarity measure [128, 129, 130] is used which is explained below.

Cosine Similarity: Cosine similarity is a measure of similarity between two vectors by measuring the cosine of the angle between them.

Given two vectors of same attributes, P and Q , the cosine similarity $Sim(P, Q)$, is computed as shown in (4.1):

$$Sim(P, Q) = \frac{P \cdot Q}{\|P\| \|Q\|} = \frac{\sum_{i=1}^n P_i \times Q_i}{\sqrt{\sum_{i=1}^n P_i^2} \times \sqrt{\sum_{i=1}^n Q_i^2}} \quad (4.1)$$

where P_i and Q_i are components of P and Q , n is the size of both vectors P and Q .

4.4.5 Topic Taxonomy

Taxonomies have been used to simplify studying the world by stratifying and partitioning it since ancient times. More recent examples are Yahoo! [160] and the Virtual Library [161]. The purpose of this set of categories is to provide a kind of basis (in the mathematical sense) onto which the user maps her interests (the focus

topics). Eventually, the focused crawler will assign relevance scores to each visited document based on how well it matches the categories associated with the focus topics. Thus, the use of taxonomy provides a natural mechanism to control the recall-precision trade-off.

In this proposed system, a set of multi level topic taxonomies are used to categorize the documents. For topic taxonomy, instead of using the existing canonical taxonomies, the system considers the digital document libraries or archives of some universities for taking the different types of categories. The detailed description about the data set used and the process to create categories is explained in Appendix A.

The schema for this data structure is shown in Fig. 4.6 and description regarding various fields of this schema is shown in Table 4.1.

Category_Keyword_Database

<i>Category_ID</i>	<i>Category</i>	<i>Keyword_ID</i>	<i>Keyword</i>	<i>Frequency</i>
--------------------	-----------------	-------------------	----------------	------------------

Fig 4.6 Schema for Topic taxonomy

Table 4.1 Description of Topic Taxonomy

Field	Description
Category_ID	Each category in topic taxonomy is assigned a unique serial number i.e. ID for referencing. The ID can be a sequential number or string e.g. C1. C2. C3 etc.
Category	Name of the category corresponding to category_ID.
Keyword_ID	When parser tokenizes a retrieved archive, a set of token or keyword (possible strings of characters) are produced. Punctuations are usually thrown out in this process. A token is stored in this field with a unique serial number i.e. ID.
Keyword	Name of the keywords corresponding to keyword_ID.
Frequency	It is the number of occurrences of the specified keyword in the category.

4.4.6 Incremental Keyword Filter

It is a part of topic taxonomy component. In the incremental technique, it updates the keywords of categories whenever a new research paper is uploaded by merging the new keywords with the existing processed keywords instead of starting from scratch. This not only saves the processing time but also saves the memory. The algorithm of this component is outlined in Fig 4.7.

As shown in algorithm: First, the module takes the keywords of the uploaded paper and the keywords of its sub-category. Then, selects the keywords of higher importance and keep them in the decided sub-category.

Algorithm: Incre_key_filter (PK,CKD)

I/P: Paper Keywords, Categorized_Keywords_Database

O/P: Updated keyword database

{

Step1: Take the keywords of newly downloaded/uploaded paper.

Step2: Take keywords of the sub-category in which paper is saved.

Step 3: Merge the keywords of first two steps, updating the frequency of repeated Keywords.

Step4: Sort the keywords according to updated frequencies.

Step5: Choose the top ten keywords and update the category with them.

Step 6:Return the updated keyword database

}

Fig 4.7 Algorithm for Incremental Keyword Filter Module

4.4.7 Advantages of Proposed Categorization Process

Proposed approach of categorization has the following advantages:

1. Using bookmark technique, to extract the important keywords of the document instead of scanning the whole documents, results in reducing the time complexity considerably without any adverse effect on the quality of results.
2. The mechanism works in an offline mode, thus does not affects the online query processing time of the search engine. Rather, it improves the search engine efficiency.
3. More precise and relevant results are retrieved by the users because of multi-level hierarchy of categories in topic taxonomy.

The next section describes another module, proposed in this work, towards crawling digital library documents.

4.5 LINK FORECASTING MODULE

This module extracts all the references from the document and forecasts them for further processing. The functioning of sub components of this module is described below:

4.5.1 Link Extractor

This component takes a document as an input and extracts all the references (i.e. outgoing links) of the document. These extracted links are forwarded to Link filter for further processing.

4.5.2 Link Filter

This component is optional. For limiting the boundaries of crawling area, an ignore list of URL types, references or domains is provided to crawler as per the user behavior which the user do not want to crawl. The *Ignore List* is a set of file types that

Table 4.2 Sample Ignore List

File Types	Extensions
image	.jpg, .bmp, .gif, .png, .jpeg, .mpeg
video	.flv, .avi, .mp4, .wmv, .avi

contains the types of URLs, references or domains as per the user interest to be ignored by the crawler while crawling. Thus, this component takes the extracted out-links of the documents and matches them with the ignore list of URLs. If any match is found, the corresponding link will be removed and not forwarded for further processing. Table 4.2 shows an example of Ignore List. This step helps in reducing the overall processing costs.

4.5.3 Link Priority Analyzer

This component assigns the priority to all unvisited references/URLs which are extracted from the downloaded document. It assigns the priority order to the references/URLs by analyzing the relevance of cited references with the downloaded document. It means, the unvisited cited references might be relevant to the downloaded documents. This component helps to put, on top of the ranked URL list, those URLs with higher rate of satisfying the user's needs. For computing the relevance between the downloaded document and unvisited URLs, Jaccard Coefficient measure [127, 130] is taken into account. It is defined as the size of the intersection divided by the size of the union of the sample sets i.e. the number of

shared terms present in documents divided by the number of all unique terms present in both documents. The Jaccard coefficient score is computed as:

$$J_Sim(D_i, R_j) = \frac{|D_i \cap R_j|}{|D_i \cup R_j|} = \frac{\sum_{k=1}^n w_{i,k} w_{j,k}}{\sum_{k=1}^n w_{i,k}^2 + \sum_{k=1}^n w_{j,k}^2 - \sum_{k=1}^n (w_{i,k} w_{j,k})} \quad (4.2)$$

where $Sim(D_i, R_j)$ represents the Jaccard similarity score between the downloaded document D_i and unvisited URL R_j , n represents the size of both the downloaded document and the document title corresponding to the unvisited URL, w_i and w_j denotes the weight of the term in the document i and j .

4.6 MISSING DOCUMENT FINDER MODULE

This module comes into the play when the page downloader does not find the .pdf format of document for downloading. This module is the heart of the proposed system which is responsible to find the desired document by using alternative methods and techniques. This module works as shown in Fig 4.8.

In this module, the reference/URL which is not in pdf format, is forwarded to *meta-data extractor*. The metadata extractor extracts the meta-data of the reference/URL i.e. title, author, publication venue etc. This freely available information about the URL is used to extract more related or missing information about the document. This type of information is used to frame more related queries for specific subject

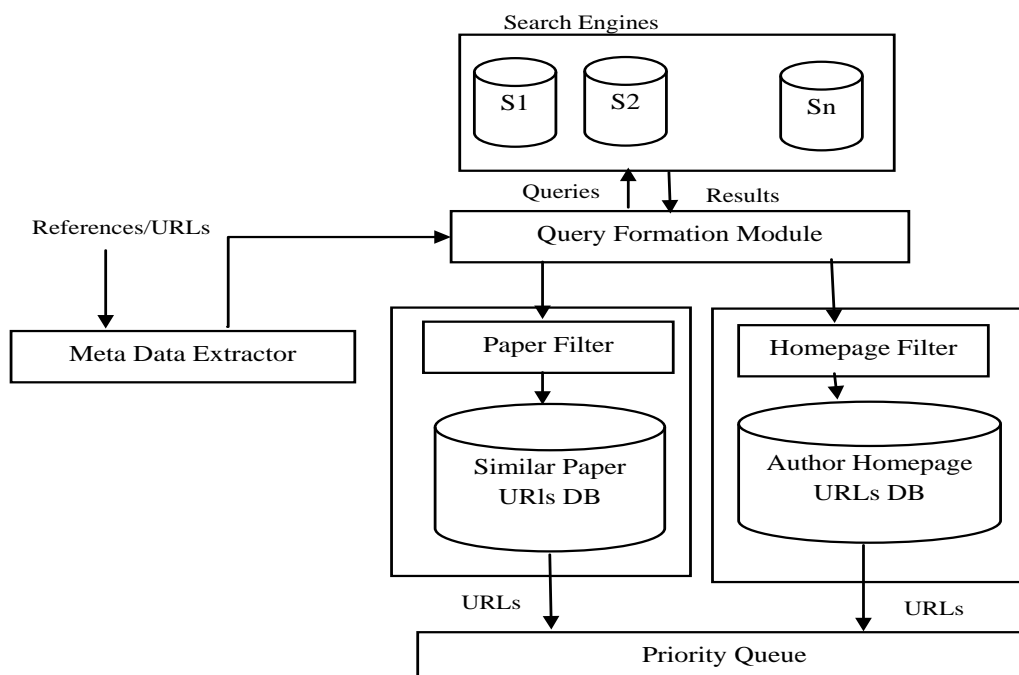


Fig 4.8 Missing Document Finder Module

disciplines. In our framework, the meta-data information is used to frame two types of queries:

Type1= Research Paper Titles, and

Type2=Author Name Queries.

By using this information, the *Query Interface* automatically generates and submits queries to *two or more search engines* (e.g. yahoo, google, google scholar etc.) requesting the more specific information. The list of search results resulting from the type1 queries are filtered by the *paper filter* and saved into a temporary database i.e. *Similar Paper URLs DB*. The results from type 2 queries are filtered by the *Homepage Filter* and saved into a temporary database i.e. *Author Homepage URLs DB*. The predicted academic author homepages and paper titles both are served as seeds and send to *Priority Queue* for further crawling.

The working of sub components is described in detail as below:

4.6.1 Query Formation Module

Meta-data record is used to generate the multiple queries which are further used for requesting the relevant document of respective query. In this proposed system, different types of queries are formed.

Let's take an example a paper having title "*Alternatives for Interconnection of Public Packet Switching Data Networks*" and whose authors are catalogued as: Vic DiCiccio, Carl A. Sunshine, James A. Field, Eric G. Manning. Various queries formed are:

- *Q1*: Unquoted title (e.g. Alternatives for Interconnection of Public Packet Switching Data Networks).
- *Q2*: Quoted title (e.g. "Alternatives for Interconnection of Public Packet Switching Data Networks")
- *Q3*: Name of first catalogued author (e.g. Vic DiCiccio)
- *Q4*: Name of all catalogued authors (e.g. Vic DiCiccio, Carl A. Sunshine, James A. Field, Eric G. Manning)

After the query formation, these queries are forwarded to different search engines for finding more desired results.

4.6.2 Author Homepages Filter

Author homepages are also known as academic homepages and form potential seed URLs for initiating crawls in digital libraries. For the system to be effective and efficient, it is imperative to identify these pages from the search results of author name queries. When a researcher hits an author name query (i.e. Q3 and Q4), the retrieved list of search results contain a lot of non-homepage URLs which are expected to be diverse with web pages ranging from commercial websites such as LinkedIn, social media websites such as Twitter and Facebook, publication listings such as Google Scholar, Research Gate, and several more. To handle this problem, filters are used to remove these types of irrelevant URLs from the search result list against author name query. Here, two types of filters are used by the system in order to find the more relevant author homepages.

- *URL Features:* Intuitively, the URL strings of academic homepages can be expected to contain, terms such as “people” “author”, or “home” and less likely to be hosted on domains such as “linkedin”, “twitter”, and “facebook”. In the proposed system, URL strings are tokenized based on the “slash (/)” separator and the domain-name part of the URL based on the “dot (.)” separator.

For the example (as shown in Fig 4.9), the URLs which contain the author name as URL string (john.blitzer.com) has more possibility to link with the author homepage as compared to the URL which contain the URL string LinkedIn (https://www.linkedin.com/in/john-blitzer-425665).

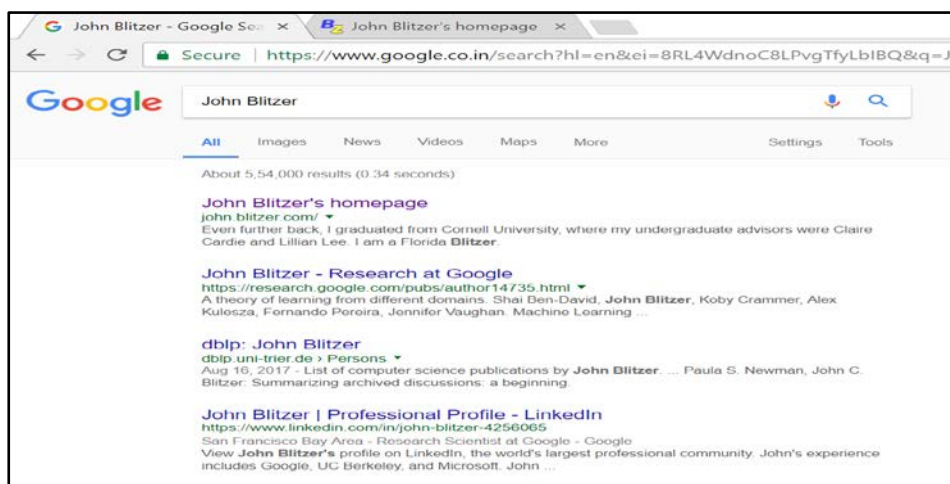


Fig 4.9 Example of URL feature

- *Name-match Features:* This feature takes the general factor into consideration that generally researchers tend to use their name or part of URL string of their homepages. Two types of match features are specified:

(1) a Boolean feature that indicates whether any part of the author name matches a token in the URL string, and

(2) a Numeric feature that indicates the extent to which name tokens overlap with the (non-domain part of) URL string given by the fraction:

$$\frac{\neq \text{ matches}}{\neq \text{ nametokens}} \quad (4.3)$$

For the example (as shown in Fig 4.10) author name “Kavi Arya” and the URL string: <https://www.cse.iitb.ac.in/~kavi/>, the two features have values as:

A Boolean feature=“true” and

A Numeric feature= $\frac{|kavi|}{|kavi\ Arya|} = \frac{1}{2} = 0.5$, respectively.

Based on these features, the system classifies the URLs into homepages and non-homepages category, filters them and further forwards them for initiating crawling in digital libraries.

HomePage of Kavi Arya at KReSIT - IIT Bombay
www.itb.ac.in/~kavi/ ▼
 Faculty. Back to CSE Homepage. **Kavi Arya** Associate Professor. **Kavi Arya**, Computer Science & Engineering Dept., Indian Institute of Technology Bombay.

Kavi Arya | LinkedIn
<https://in.linkedin.com/in/kavi-arya-05233b1>
 View **Kavi Arya**'s professional profile on LinkedIn. LinkedIn is the world's largest business network, helping professionals like **Kavi Arya** discover inside ...

Prof.Kavi Arya - Nex Robotics
www.nex-robotics.com > Testimonials ▼
Kavi Arya We used the robots to help us teach Embedded Systems course material to students - both local and through the Distance Education Program.

Kavi Arya - Google Scholar Citations
scholar.google.co.in/citations?user=Qz7H0U0AAAAJ&hl=en ▼
 Professor of Computer Science and Engineering, IIT Bombay - cse.iitb.ac.in
Kavi Arya. Professor of Computer Science and Engineering, IIT Bombay · Functional Programming ...
 SS Rode, S Vijay, P Goyal, P Kulkarni, K Arya. Electronic ...

Kavi Arya Profiles | Facebook
<https://www.facebook.com/public/Kavi-Arya> ▼
 View the profiles of people named **Kavi Arya**. Join Facebook to connect with **Kavi Arya** and others you may know. Facebook gives people the power to share...

Fig 4.10 Example of Name-Match Feature

4.6.3 Paper Filter

This component filters the URLs with no or less interest to the user against paper title query (i.e. Q1 and Q2). This step helps to reduce the processing cost of the next step.

For this, the system considers the URLs having title similarity value of the document greater than the threshold value as compared to requested document title (i.e. Q1 and Q2). For computing the similarity, the Jaccard similarity coefficient [127, 130] is computed over the keywords of the titles by using eq. (4.2).

4.6.4 Example Illustration

Let us take an example to illustrate the working of missing document finder module. Assume, the system takes the seed URL i.e. document title: “A *Heuristic-based Hierarchical Clustering Method for Author Name Disambiguation in Digital Libraries*”. First, the page downloader tries to download the pdf format of the document against the seed URL, but suppose it fails. Then the document is forwarded to missing document finder module. Here, the module first extracts the metadata of the URL (i.e. title, authors). By using this meta data, the system forms different queries and hits these queries on different search engines. As per the type 2 query (i.e. document title in quotes), when the system hits the query: “A *Heuristic-based Hierarchical Clustering Method for Author Name Disambiguation in Digital Libraries*”, the search result list (as shown in Fig 4.11) is extracted. After computing the similarity value by paper filter, top URLs are forwarded to priority queue. It is

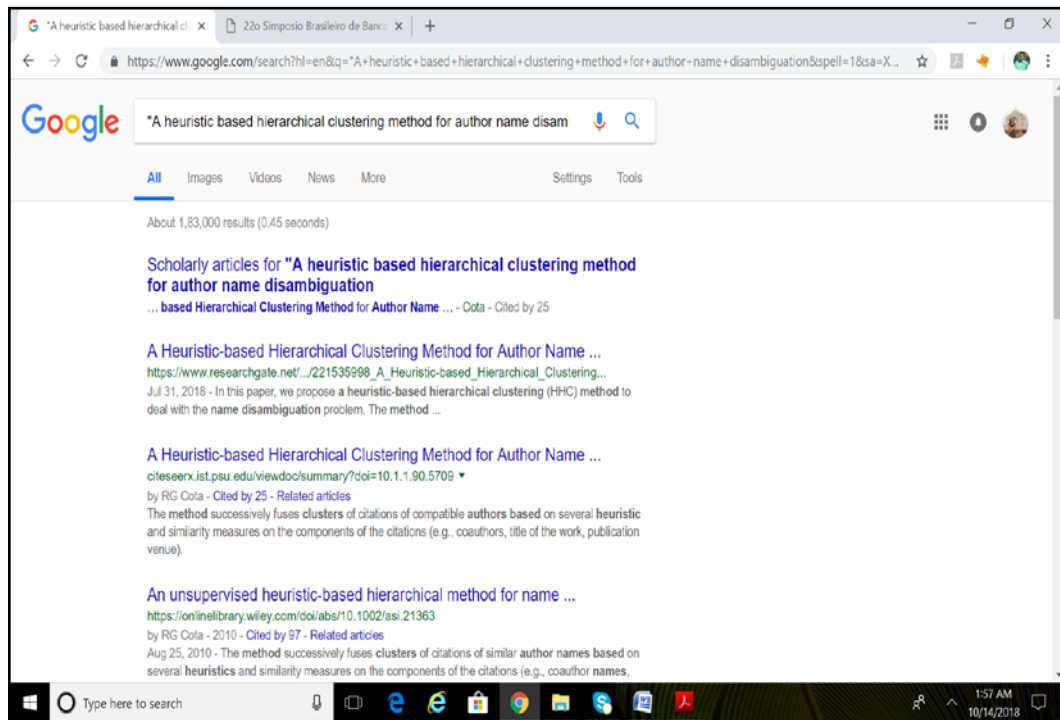


Fig 4.11 Search Result list against query on Google search engine

noted that by going through these URLs, by clicking the first URL, system will get the pdf of respective document).

4.7 AGING PROCESS

As discussed above (in Section 4.5.3), the system first computes the priority score of each unvisited URLs and then depending upon their priorities sends them to *Priority Queue* or *Yet to be Crawl Database* for further processing. But Sometimes, a situation occurs when a low priority or yet to be crawled URL never get crawled because higher priority URLs take over since the list of unvisited URLs is never empty.

Let's take an example to understand the concept of how are the URLs be processed in case of priority scheduling: Consider, A, B and C are three URLs whose priority and processing time is given as show in Table 4.3.

Table 4.3 Example of Priority Scheduling

URL	Processing Time	Priority
A	10	2
B	5	0
C	8	1

According to Table 4.3, the system first starts to crawl the URL A having highest priority 2, then C and further B having lowest priority 0 (as shown in Fig 4.12).

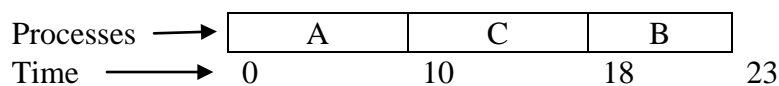


Fig 4.12 Example of Priority Scheduling

Before the processing of URL B, suppose if some high priority URL comes in the priority queue for crawling, then control is given to that URL keeping URL B behind. Sometimes, this situation can lead infinite waiting for the URLs which are having very low priority thus, creating the problem of *Starvation*. Thus, the low priority URLs tend to never be crawled unless their priority is increased by some means. To overcome this problem, the system considers the concept of *Aging*.

This concept is used to ensure that URLs with lower priority will eventually complete

their crawling. This technique can be used to reduce starvation of low priority URLs. There are many ways to implement aging, but all have the same principle that the priority of a process should increase as it waits in the ready queue. The increase in priority may or may not be equal to the waiting time of the process. In this component, the system sets up some rules and according to those rules, increases the priority level of low priority URLs. When the priority level is reached at a certain threshold value, then corresponding low priority URL will be forwarded to priority queue for crawling.

For Example, suppose a system with priority range of 0-60 with 0 means the highest priority. Consider a process with priority 40. If we increase its priority by 1 every 15 minutes, then in more than 10 hour the process will age to 0 priority and get executed.

4.8 ILLUSTRATION OF PROPOSED CRAWLING SYSTEM

Let us assume one hypothetical example to illustrate the working of the proposed system. Assume that the system takes a list of seed documents i.e. document titles provided by user as:

1. OSI Reference Module: An Overview
2. Web Usage Mining And Pattern Discovery
3. A Novel Approach for Document Ranking in Digital Libraries
4. Using Cohesion and Coupling for Software Remodularization: Is It Enough?

Let's consider only one seed document i.e. "*OSI Reference Module: An Overview*" for illustrating the working of proposed digital library focused crawler.

First, the page downloader downloads the respective document. If the document is in pdf/.ps/.pz format, then it is forwarded to text extractor/parser which parses the

Table 4.4 Parsed Documents

Title of the page/document	OSI Reference Model: An Overview
Authors	Gaurav Bora, Saurabh Bora, Shivendra Singh, Sheikh Mohamad Arsalan
Keywords with their frequency (only top 10 is taken)	Layer=11, Protocol=8, OSI= 4, Architecture= 3, Manage= 2, Connect= 1, Multiplex=1, Split=1, Transfer=1, Physical=1
Venue	International Journal of Computer Trends and Technology (IJCTT)
Year of published/upload	Jan 2014

as shown in Table 4.4, otherwise sends it to missing document finder module. Now, the document categorizer decides the category of the document on the basis of its keywords. The system considers the online topic taxonomy (as shown in Fig 4.13) for instance. The keywords of each category are shown in Table 4.5.

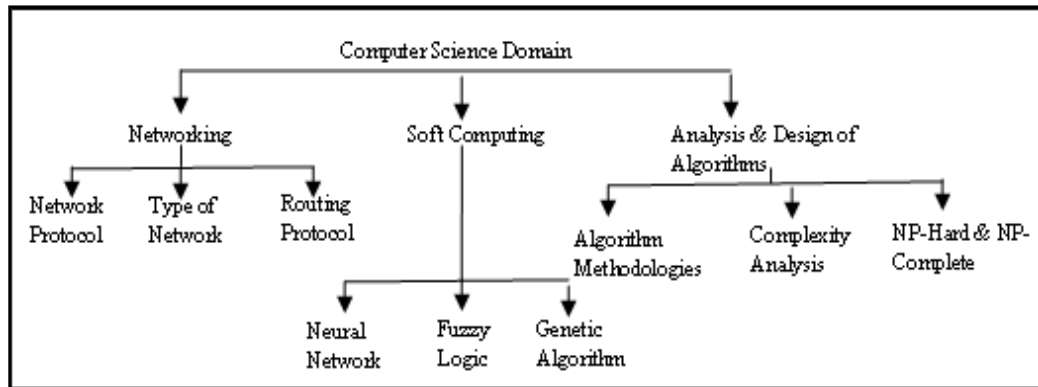


Fig 4.13 Topic Taxonomy

The categorizer compares the document keywords with the keywords of the topic taxonomy categories. Table 4.6 shows the comparison of keywords of parsed paper and keywords of the categories for deciding the category of document.

The similarity between the keywords of parsed paper P and keywords of categories C is computed by using (4.1):-

Table 4.5 Keywords

Keywords of Parsed Document		Keywords of Categories					
		Networking		Soft Computing		Analysis & Design of Algorithm	
Keywords	Freq	Keywords	Freq	Keywords	Freq	Keywords	Freq
Learning	3	Switch	15	Neural	13	Algorithm	27
Genetic	10	SNMP	10	Neuron	9	Complexity	22
Habituation	1	Wired	14	Genetic	18	Optimize	16
Architecture	3	UDP	6	Crossover	13	NP-Hard	15
Defuzzification	5	Ethernet	5	Defuzzification	15	NP-Complete	15
Connect	1	ATM	10	Expert	12	Sort	12
Expert	4	Wireless	8	Mutation	3	Space	17
Split	1	OSI	15	Learning	2	Symptotic	15
Transfer	1	Layer	21	Chromosome	1	Asymptotic	17
Mutation	1	Signaling	7	Habituation	1	Time	15

Table 4.6 Frequency of Keyword

Keywords	Frequency in category	Frequency in paper
Neural	13	0
Neuron	9	0
Genetic	18	10
Crossover	13	0
Defuzzification	15	5
Expert	12	4
Mutation	3	1
Learning	2	3
Chromosome	1	0
Habituation	1	1

$$\begin{aligned}
 \text{Similarity}(P, C) &= \frac{18 * 10 + 15 * 5 + 12 * 4 + 13 * 1 + 14 * 3 + 7 * 1}{\sqrt{1127} * \sqrt{152202}} \\
 &= \frac{365}{33.57 * 12.32} = 0.882
 \end{aligned}$$

The relevancy value of *Networking* category with the document is 0.714. There are no common keywords between the document and other two categories. So for them, the relevancy value is 0. Thus, *Networking* is selected as the document category. If the document is relevant, then link extractor extracts all the out-going links (references) as shown in Table 4.7.

Now, the link filter filters the links by referring the ignore list. It forwards all the extracted links to link priority analyzer except link number 15 due to an image link (as shown in Table 4.7). The link Priority Analyzer finds priority of extracted unvisited URLs by computing the Jaccard coefficient similarity score between the downloaded document and title of the unvisited documents/URLs. The Jaccard similarity score is also shown in Table 4.7.

The top 10 ranked unvisited URLs are forwarded to priority queue for further crawling and rest of the links are saved to *yet to be crawl* temporary database

Now, for illustrating the working of *Missing Document Finder Module*, let's take reference or link number 12 (as shown in Table 4.7).

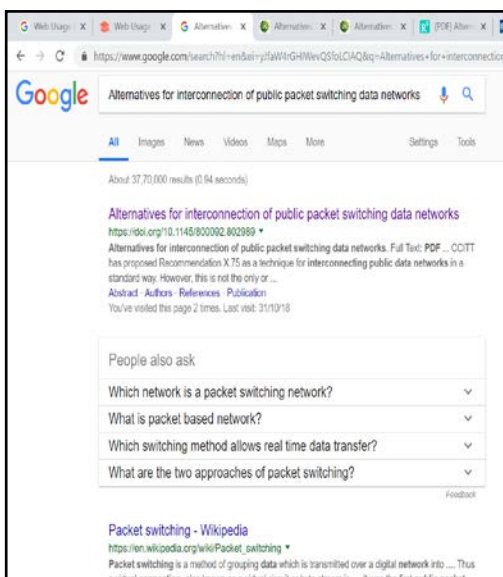
Table 4.7 Extracted Link with their Similarity Score

	Reference	Jaccard Similarity Score
1	L. G. Roberts, B. D. Wessler, "Computer network development to achieve resource sharing", Proc. SJCC, pp. 543-549, 1970.	0.719
2	L. Pouzin, "Presentation and major design aspects of the CYCLADES computer network", Proc. 3rd ACM-IEEE Commun Simp., pp. 80-87, 1973-Nov.	0.689
3	J. H. McFayden, "Systems network architecture: An overview", IBM Syst. J., vol. 15, no. 1, pp. 4-23, 1976.	1.44
4	G. E. Conant, S. Wecker, "DNA An Architecture for heterogeneous computer networks", Proc. ICCS, pp. 618-625, 1976.	1.063
5	H. Zimmermann, "High level protocols standardization: Technical and politica! issues", Proc. ICCS, pp. 373-376, 1976-Aug.	0.751
6	"ISO/TC97/SC16", Provisional model of open systems architecture, Mar. 1978.	0.902
7	"ISO/TC97/SC16", Reference model of open systems interconnection, June 1979.	0.555
8	H. Zimmermann, N. Naffah, "On open systems architecture", <i>Proc. ICCS</i> , pp. 669-674, 1978-Sept.	0.817
9	H. V. Bertine, "Physical level protocols," this issue pp. 433-444.	0.822
10	H. C. Folts, "Procedures for circuit-switched service in synchronous public data networks," and "X.25 transaction-oriented features-Datagram and fast select," this issue, pp. 489-496.	0.515
11	J. W. Conard, "Character oriented data link control protocols," this issue, pp. 445-454.	0.766
12	Vic DiCiccio, Carl A. Sunshine, James A. Field, Eric G. Manning D. E. Carlson, "Alternatives for interconnection of public packet switching data networks," Published in Proceeding SIGCOMM '79 Proceedings of the sixth symposium on Data communications Pages 120-125.	0.615
13	"IS 4335", High level data link control-elements of procedure, 1977.	0
14	"X25", Orange Book, vol. VIII-2, pp. 70-108, 197	0
15	http://media.techtarget.com/digitalguide/images/Misc/osi.gif	
16	"ISO/TC97/SC16/N23", Proposal for a standard virtual terminal protocol, Feb. 1978.	0.782
17	"EEC/WGS/165", Data entry virtual terminal protocol for EURONET.	0.766
18	"DP 6429", <i>Extended control characters for I/O imaging devices.</i>	0
19	J. Day, "Terminal protocols", this issue, pp. 585-593.	0.850

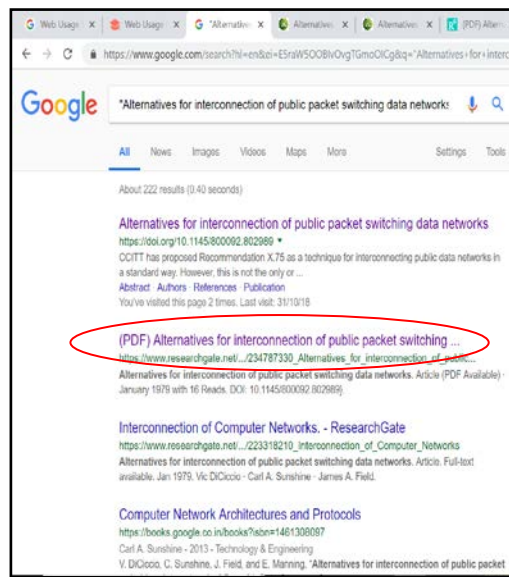
When this reference goes for further crawling, then page downloader is not able to download its pdf (as shown in Fig 4.14 (a)) and link is forwarded to *Missing*

Document Finder Module for forming different combination of queries using title and author's name on different search engine.

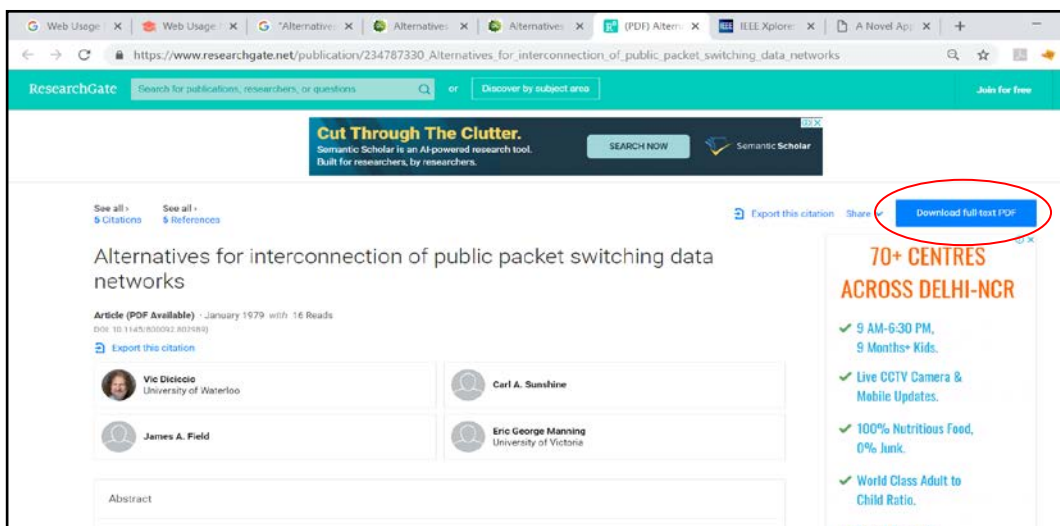
While hitting the Q2 type query i.e. document title in quoted form, then a search result list (as shown in 4.14 (b)) is returned by search engine and is forwarded to priority queue for further processing after applying paper filter. From this list, pdf downloader is able to download the pdf of respective document as shown in Fig 4.14 (c).



(a)



(b)



(c)

Fig 4.14 (a) When pdf downloader is not able to download pdf, (b) list of search result while hitting the Q2 type query i.e. Quoted Title of document and, (c) when pdf downloader downloads the document by getting the link through missing Document Finder Module

Here, it can be concluded that by using meta data information for forming different types of queries on different search engine, the proposed digital library focused crawler proves to get more efficient and effective results.

4.9 SUMMARY

The proposed approach discussed in this chapter is summarized in Table 4.8. It is observed that proposed digital library focused crawler optimizes the crawl process of the digital library search system.

Table 4.8 Summary of Proposed System

Parameters	<i>Focused Crawler</i>
<i>Module Optimization</i>	Crawler
<i>Metric</i>	Crawling topic specific papers and find the missing documents information.
<i>Mined Web Resource</i>	Web graph, various data structures and Search engines like Google, Google scholar.
<i>Type of Mining</i>	Web Structure and Web Content
<i>Advantages</i>	<ul style="list-style-type: none"> • The papers are categorized while crawling thus provide more precise and relevant results to the user. • The relevant papers which previously were not appearing in the results are made to appear.

The next chapter describes in detail proposed indexing technique i.e. the technique developed for the organization of documents at the backend in order to retrieve the results efficiently and effectively on the front end of the search engine.

Chapter V

MULTI-LEVEL INDEXING TO INDEX DIGITAL DOCUMENTS

5.1 GENERAL

With the huge corpus of digital information present on the WWW, the need to efficiently find specific piece of digital information as per user interest becomes crucial. A digital library search engine is an information retrieval system designed to find the online academic documents or article stored on WWW as per the user interest. In digital libraries, the index structure has been considered as the important component to support fast searching. Indexing [73, 74] is an assistive technology mechanism which helps to optimize the speed of digital library search engine in finding the relevant documents against the user query. Indices are used to provide a framework for researchers to locate the documents quickly and efficiently. In this chapter, a multi-level index structure is proposed.

5.2 PROPOSED APPROACH OF INDEXING

The proposed system provides a sequential as well as direct access of documents stored in the index. Also, the documents are clustered on the basis of category, which further provides more refined results to the user query. The architecture of proposed system is shown in Fig 5.1 wherein the *Web Crawler* (discussed in chapter IV) crawls or harvests the digital documents from the WWW, and these crawled documents are saved in a *Paper Repository*. From the *Paper Repository*, the documents are processed by the *Similarity Analyzer* for computing the similarity between the documents. Based on these similarity values, the *Clustering Generator* generates the clusters and stores these clusters into *clustering database*. Then, *Index Generator* generates the index structure by using the information from the paper repository (i.e. category of paper) as well as clustering database. When a user or researcher hits a query through *Search Interface*, the *Query Keyword Extractor* extracts the query terms and forwards them to *Query Analyzer*, first extracts the category of the query by using the *Topic Taxonomy* and then query category

is searched in the primary database. Once a category match is found, then the query terms are matched with the keywords associated with each clusters in order to get the clusters of documents. After matched cluster is found, the list of documents under matched cluster is retrieved and sent to the dynamic ranking component for ranking the retrieved results as per user query. Finally, a ranked list of documents is returned to the user through search interface.

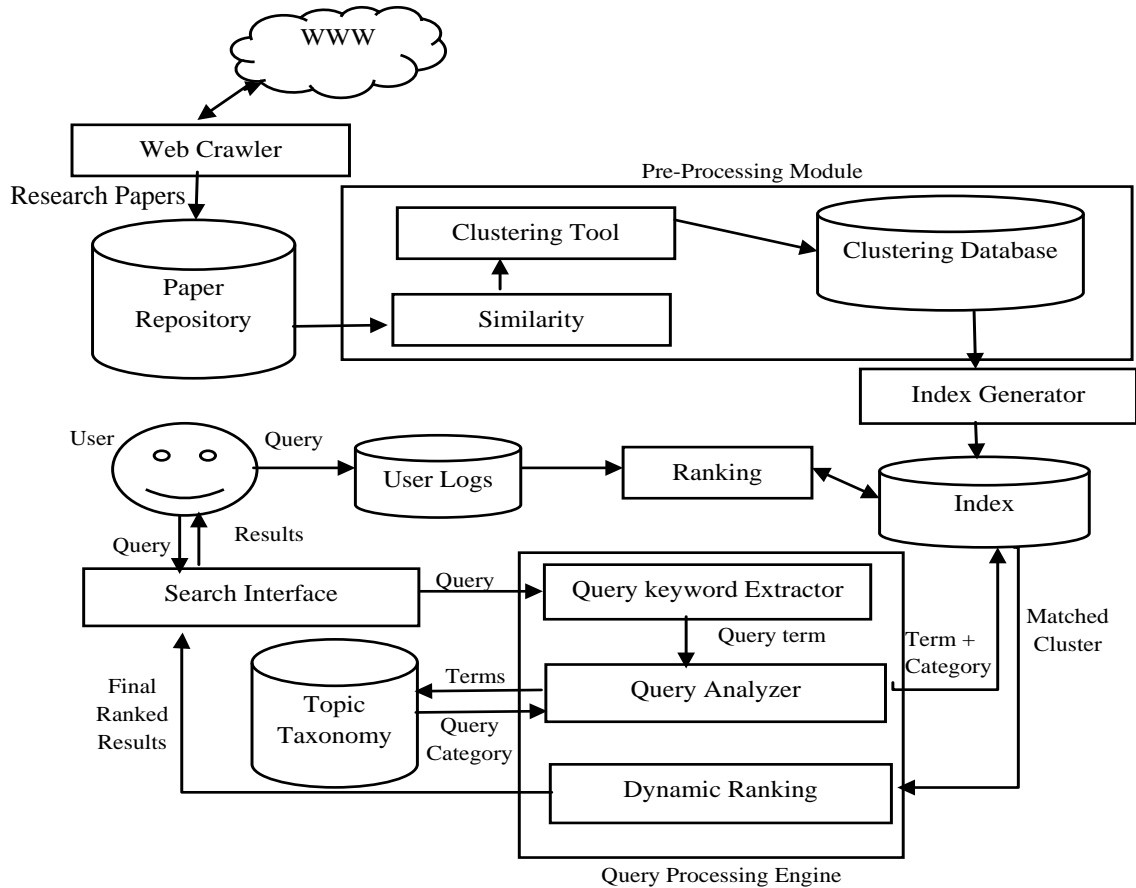


Fig 5.1 Architecture of Proposed Search System

The detailed working of the component modules is described in the following sections.

5.3 WEB CRAWLER

This module is responsible to crawl all the digital documents from the WWW and stored them into the paper repository. Before storing the digital documents, pre-processing of the digital documents is done and saved in the form of keywords, title, author, references,

their category etc. The detail working of this component has been described in Chapter IV.

5.4 PRE-PROCESSING MODULE

The pre-processing module is responsible to extract relevant information from documents so as to solve in the index. This module (as shown in Fig. 5.1) contains further two modules Similarity Analyzer and Cluster Generator which are described below in detail.

5.4.1 Similarity Analyzer

This module takes the documents from the paper repository as an input and computes the similarity between them. The computation of similarity between the documents means: which keywords or terms appear in the document, at what location and they appear in (i.e. frequency of occurrence)? There are lots of approaches that have been used to calculate the similarity between two publications, but here the proposed system takes the weight of the keywords present in the document into consideration for computing the similarity.

Similarity between the publication P and publication Q can be measured by computing cosine similarity measure [128, 130] which is denoted by $Sim(P, Q)$. The cosine similarity measure is denoted as shown below:

$$Sim(P, Q) = \cos \theta = \frac{P \cdot Q}{\|P\| \|Q\|} = \frac{\sum_{i=1}^n W_{p,i} \times W_{q,i}}{\sqrt{\sum_{i=1}^n W_{p,i}^2} \times \sqrt{\sum_{i=1}^n W_{q,i}^2}} \quad (5.1)$$

where $W_{p,i}$ and $W_{q,i}$ denote the weight of term t_i in the publication P and Q respectively.

The weight i.e. W_t of a term is computed as:

$$W_t = tf_t * idf_t \quad (5.2)$$

where tf_t denotes the term frequency and idf_t denotes the inverse document frequency. tf_t and idf_t are further described as:

Term-Frequency: In information retrieval, term frequency is defined as the raw count of a term i.e. the number of times a term appears in a document. Term frequency (tf) [137] of any term t is denoted as:

$$tf_t = \frac{\text{frequency of term in a document}}{\text{total number of terms in a document}} \quad (5.3)$$

Inverse Document Frequency: It is defined to measure how important a term is. While computing term frequency, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus, we need to weight down the frequent terms while scaling up the rare ones. The inverse document frequency (idf) [137] of any term t is computed as:

$$idf_t = \log \frac{\text{Total number of documents}}{\text{number of documents in which the term appears}} \quad (5.4)$$

5.4.2 Cluster Generator

This component is used to find out the clusters of documents which are similar in nature. The clustering of publications is done based on the category as well as similarity between them. The algorithm works as follows: initially, all publications are assumed to be individual i.e. not belonging to any cluster. First, the publications having the similar category are extracted from the paper repository, and then, the similarity between all these similar documents is calculated by using (5.1). If the similarity measure between the publications is greater or equal to the threshold value (τ), then the papers are placed into the same cluster or group. This process is repeated until all publications belong to any one of the clusters. Finally, the returned clusters are stored in the *Clustering Database*. The algorithm for computing the similarity and generating the clusters is outlined in Fig 5.2.

5.4.3 Illustrative Example:

Consider four documents with the fragment content as given below:

d_1 =A computer system is a basic functional system including hardware and software which are required to make it functional for the user.

d_2 =A system software is a type of computer program which is designed to run a computer's hardware and application programs. The operating system is example of system software.

Algorithm: Cluster (D)

Input: set of n documents with similar category $D = \{d_1, d_2, d_3, \dots, d_n\}$.

Output: k clusters of documents, $C = \{c_1, c_2, c_3, \dots, c_k\}$.

```
{
STEP1: Convert all the documents in the vector form.
STEP2: for (i=1, i ≤ n, i++)
    {
        Flag(di)= false
        C={∅}
        Ci={di}
    }
For (i=1, i ≤ n, i++)
    {
        Take document di
        For (j=2, j ≤ n, j++)
            {
                Compute Sim(di, dj)
            }
    }
STEP3: If Sim(di, dj) ≥ threshold (TH)
    {
        Ci = Ci ∪ {di, dj}
        Flag(di) = Flag (dj) = true
        If Ci ≠ ∅ then
            C = C ∪ Ci
    }
STEP4: Extract di or dj on their similarity basis and assign a cluster.
}
```

Fig 5.2 Algorithm for Clustering the Documents

d_3 = A computer system receives user input, process data and with this processed data, create information for storage and output.

d_4 = Operating system is the system software which is designed to provide a platform to other software. It co-ordinates between devices and schedules multiple tasks as per priority.

The set of terms with their term frequencies in respective documents is depicted in Table 5.1.

Let $\sum F_{d1} = 12$, $\sum F_{d2} = 16$, $\sum F_{d3} = 13$ and $\sum F_{d4} = 14$ i.e. the number of terms in d_1 , d_2 , d_3 and d_4 respectively.

Let us calculate the term frequency (tf) for the term “system” using (5.3).

Table5.1 Term Frequencies and their Weights

Terms	Term Frequencies in Documents				Weight of Terms in Documents			
	F _{d1}	F _{d2}	F _{d3}	F _{d4}	W _t in d ₁	W _t in d ₂	W _t in d ₃	W _t in d ₄
System	2	2	1	1	0.332	0.250	0.153	0.142
Computer	1	2	1	0	0.02	0.04	0.01	0
Hardware	1	1	0	0	0.02	0.01	0	0
Software	1	2	0	2	0.02	0.04	0	0.04
User	1	0	1	0	0.02	0	0.01	0
Information	0	0	1	0	0	0	0.01	0
Storage	0	0	1	0	0	0	0.01	0
Output	0	0	1	0	0	0	0.01	0
Input	0	0	1	0	0	0	0.01	0
Data	0	0	2	0	0	0	0.04	0
Program	0	2	0	0	0	0.04	0	0
Design	0	1	0	1	0	0.01	0	0.02
Operating	0	1	0	1	0	0.01	0	0.02

In d₁:

$$tf_{system} = \frac{2}{12} = 0.166$$

Let us assume that there are in total 10,000 documents crawled by the crawler and in only 100 documents the term *system* is present. The inverse document frequency for the same can be calculated using (5.4).

$$idf_{system} = \log \frac{10,000}{100} = 2$$

Therefore, by using (5.2), the weight of the term *system* in d₁ can be calculated as below:

$$W_{system \text{ in } d_1} = 0.166 * 2 = 0.332$$

Similarly, the TF and IDF for the term *system* in document d₂ can be calculated.

$$tf_{system} = \frac{2}{16} = 0.125$$

$$idf_{system} = \log \frac{10,000}{100} = 2$$

$$W_{system \text{ in } d_2} = 0.1251 * 2 = 0.250$$

In d₃:

$$tf_{system} = \frac{1}{13} = 0.076$$

$$idf_{system} = \log \frac{10,000}{100} = 2$$

$$W_{system \text{ in } d_3} = 0.076 * 2 = 0.153$$

In d_4 :

$$tf_{system} = \frac{1}{14} = 0.071$$

$$idf_{system} = \log \frac{10,000}{100} = 2$$

$$W_{system \text{ in } d_4} = 0.071 * 2 = 0.142$$

Similarly, weight for all the terms can be calculated as shown in Table 5.1.

Now, the similarity analyzer computes the similarity between the documents by using (5.1) as shown in Table 5.2.

Table 5.2. Similarity Matrix

	d₁	d₂	d₃	d₄
d₁	1			
d₂	0.290	1		
d₃	0.158	0.799	1	
d₄	0.730	0.228	0.199	1

In this system, after analyzing the similarity values between the documents, the average value of the similarity values is calculated and is considered as a threshold value. Here, in this case, threshold value is assumed as 0.5 or 50%.

Since, $0.79 > 0.50$ (i.e. which is greater than the threshold value). Hence, it can be concluded that documents d_1 and d_4 have been allotted to the same cluster.

5.5 INDEX GENERATOR

The index generator generates the index by using the information from the clustering database which consists of the cluster of similar category documents. The index structure formed here is the multilevel index structure.

5.5.1 Index Structure

In this proposed system, two types of indexes are generated i.e. primary index and sec-

-ondary index (as shown in Fig 5.3). Primary index consists of the term ID, term and their corresponding categories. Secondary index consists of the multi-level index structure. In this type of index structure, indices are constructed in levels. Multilevel index makes the search process fast and more efficient as compared to other types of index structures. In secondary multi-level index, the first level of index is category based search which consists of the category ID (C_id) and category, second level consist of the category ID, G_keywords i.e. keywords associated with each clusters and cluster ID (G_id) of the corresponding category of the terms. The last level consists of the corresponding document IDs of the documents present in that cluster.

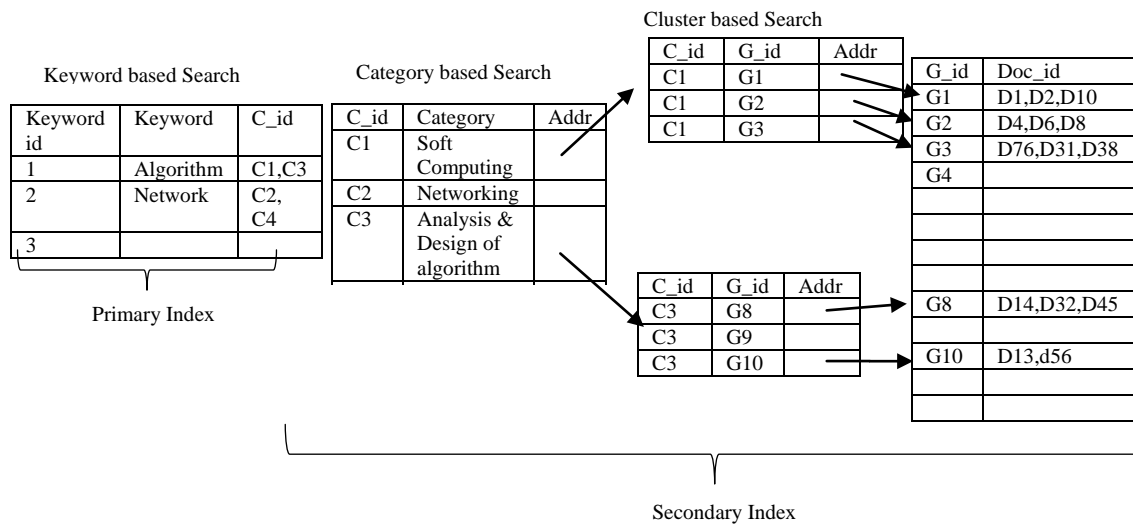


Fig 5.3 Multi-Level Index Structure

5.5.2 Illustrative Example

Let us suppose, user fires the query “*measure algorithm complexity*”. Here, first, the query processing engine analyses the query and finds the category of query term. In this case (as shown in Fig 5.3), the category comes out to be *analysis & design of algorithm* i.e. C_id -->C3. After that, this category will be searched in the secondary index; if match appears, then this category ID is now searched in the next level of secondary index for cluster based search. As in this case, system has already found out the category of the query i.e C3, next system finds the clusters by using cluster based search. At this level,

the query terms are matched with keywords of all clusters of corresponding category ID. If the match found, then corresponding cluster ID is retrieved and the documents contained in that cluster ID will be retrieved as the final result set of the search process. Thus, in this case query keywords such as *measure*, *algorithm*, and *complexity* are compared with keywords of all clusters corresponding to category *C3*. After comparison, the clusters that comes out to be *G8* and thus, documents corresponding to this clusters are *D14*, *D32* and *D45*.

Thus, final ordering of retrieval process is as:

$$C_id \rightarrow C3 \rightarrow G_id \rightarrow G8 \rightarrow D14, D32, D45$$

If, in any case, the category of the query term is not found in the topic taxonomy database, then primary index structure is used for performing the normal keyword-based search.

5.5.3 Data Structures

Data structures play an important role in the task of information accumulation. The existing data structures employed by indexing process have been updated for the sake of better interpretability, efficiency and effectiveness. Following are the modified data structures, which are used in the crawling & indexing process of the proposed Digital Library Search system.

The schema for these data structures is shown in Fig. 5.4.

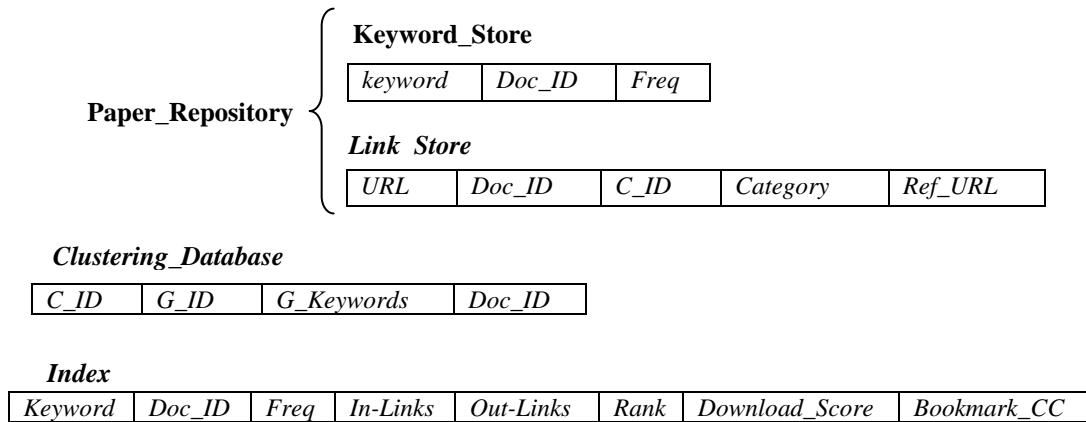


Fig 5.4 Data Structures for Crawling and Indexing Process

It may be noted that *Paper Repository* is basically a combination of two sub-schemas: *Link store* and *Keyword store*. The four data structures are described in detail as follows.

a) Paper_Repository: This repository contains entire information about all downloaded papers. It employs two data structures to store this information: the *link store* and *keyword store*. The *link store* contains structural summary of the citation graph, while *keyword store* contains information regarding the content of papers/documents. The description of various fields in these two schemas is described in Table 5.3.

Table 5.3 Description of Paper Repository

Field	Description
URL	Name of the Reference (say r_i), which has been fetched and downloaded from the web.
Ref_URL	Name of the references/URLs corresponding to out linked pages i.e. references of the fetched paper.
C_ID	The Category ID of the paper, in which the specified paper appears.
Keyword	Name of the keywords corresponding to keyword_ID.
Doc_ID	When parser tokenizes a retrieved archive, a set of token or keyword (possible strings of characters) are produced. Punctuations are usually thrown out in this process. A token is stored in this field with a unique serial number i.e. ID.
Frequency	It is the number of occurrences of the specified keyword in the category.

(b) Clustering Database: Clustering Database keeps the record of the category and the clusters of documents which belong to that particular category. The fields in the clustering database indicate the information as shown in Table 5.4.

(c) Index: Index contains information about all the keywords (terms) present in the downloaded papers. Here, a keyword means something different from a token. A

Table 5.4 Description of Clustering_Database

Field	Description
C_ID	The Category ID of the paper, in which the specified paper appears.
G_ID	The Cluster ID of the paper, in which the specified paper appears.
G_Keywords	The cluster keywords are the keywords associated with each cluster.
Doc_ID	The Document ID of the paper, in which the specified keyword appears.

normalized token after undergoing linguistic preprocessing (stemming, lemmatization processes etc.) is called as keyword. Index stores all the paper keywords alphabetically. The fields in the index indicate the information as shown in Table 5.5.

Table 5.5 Description of Index

Field	Description
Keyword	A normalized token in a paper.
Doc_ID	The Document ID of the paper, in which the specified keyword appears.
In-Links	The number of back links of the paper derived from the Link_Store repository.
Out-Links	The number of forward links of the paper derived from the Link_Store repository.
Frequency	It is the number of occurrences of the specified keyword in the category.
Download Score	It is an integer number indicating the number of times users downloaded the paper. This information is derived from the search engine logs.
Rank	It is a score provided to a paper which is generally based upon its link information e.g. Google's PageRank. The rank may also be provided on other parameters of the paper such as its content, click count etc.
Bookmark_CC	It is a score provided to a paper which is generally based upon its link and content information.

The next section describes the working of query processing engine, developed in this work, towards relevant document retrieval by taking into consideration the category of user queries.

5.6 QUERY PROCESSING ENGINE

This component takes the user query as an input and processes it for finding the desired results. The query processing engine performs the tokenization, stemming and lemmatization on the user query and finds the category of query term by comparing the keywords of query terms with the keywords of category in topic taxonomy. For computing the similarity, the cosine similarity measure [128, 130] is used by using (5.1). The matched query category and term are in turn searched in the index structure, and the clusters of documents related to the query are retrieved. The cluster of documents is

extracted on the basis of their category and their similarity with the query term. This component is further divided into sub-components as:

1. Query Keyword Extractor
2. Query Analyzer
3. Dynamic Ranking

5.6.1 Query Keyword Extractor

This component takes the submitted user query as an input, extracts the query keywords and forwards them to query analyzer for further processing.

5.6.2 Query Analyzer

Query analyzer analyses the query and performs tokenization, lemmatization and stemming on the submitted query to find out the category of the terms from the topic taxonomy. Both the query term and its category are searched in the index structure. First, the clusters of documents are extracted on the basis of their category. If, in any case, the category of the query is not present in the topic taxonomy database, then the normal keyword-based search is performed which returns the documents related to the user query in search operation, and the index structure for this is the general inverted index containing the terms and their document IDs i.e. the primary index.

For finding the category of the query, the system compares the query terms with the keywords of categories in topic taxonomy. For computing the similarity, cosine similarity measure [128, 130] is used as illustrated in (5.1).

5.6.3 Dynamic Ranking

The query processing engine retrieves the digital library search result list in the form of clusters from the index against the user query. Now, in order to rank the result list so that most relevant results get appear at the top of list, dynamic ranking is computed. The detailed description of this module is described in next chapter.

5.6.4 Illustrative Example

Suppose a user fires a query “*Mutation and crossover operator*” on the system. Query is processed by applying tokenization, stop word removal and stemming techniques and resultant keywords retrieved are mutation, crossover and operator.

Now, the query analyzer decides the category respective to the query by comparing its keywords with keywords of the categories. The comparison of these keywords with keywords of existing categories is done to decide the main category. Table 5.6 contains the keywords of main categories.

Table 5.6 Keywords in Main Categories

Keywords of Main Categories					
Networking		Soft Computing		Analysis & Design of Algorithm	
Keywords	Frequency	Keywords	Frequency	Keywords	Frequency
Switch	15	Neural	13	Algorithm	27
Node	12	Fuzzy	23	Complexity	22
Protocol	14	Genetic	18	Greedy	16
Wireless	11	Inference	13	NP-Hard	15
Multiplex	9	Defuzzification	15	NP-Complete	15
SNMP	10	Back-	12	Sort	12
TCP/IP	8	Mutation	13	Search	17
OSI	15	Regression	14	Symptotic	15
Layer	21	Chromosome	8	Asymptotic	17
Route	7	Simulate	7	Knapsack	15

Table 5.7 shows the cosine similarity values obtained after comparing the query keywords with the keywords of main categories.

After comparison, resultant category is **Soft Computing**. Now move further in the main category to decide the sub-category, the keywords of sub-categories of soft computing

Table 5.7 Cosine Similarity Values

Category	Cosine similarity value
Networking	0
Soft Computing	0.23
Analysis and Design of Algorithm	0

Table 5.8 Keywords of Sub-Category

Keywords of Sub-category					
Neural Network		Fuzzy Logic		Genetic Algorithms	
Keywords	Frequency	Keywords	Frequency	Keywords	Frequency
Neural	13	Fuzzy	23	Genetic	18
Learning	14	Inference	13	Mutation	13
Backpropagation	12	Defuzzification	15	Crossover	10
ADALINE	7	Uncertainty	8	Chromosome	8
Activation	5	Expert	5	Fitness	12
Neuron	9	Logic	18	GA	8
SVM	7	Membership	13	Selection	8
Habituation	4	Controller	8	Reproduction	7

are shown in Table 5.8.

Similarity values which are obtained by comparing query keywords with the sub-categories of soft computing using cosine similarity are shown in Table 5.9.

Table 5.9 Cosine Similarity Values

Category	Cosine similarity value
Neural Network	0
Fuzzy Logic	0
Genetic Algorithms	0.45

It can be observed from the comparison that the most relevant category is *Genetic Algorithms*.

Next, the query processing module processes the query along with the category and retrieves the matched cluster of documents as per the user query.

Let us consider some set of documents (denoted by A, B, C, D.....,J) the under same category but grouped into two different clusters or groups as per their similarity measures as depicted in Table 5.10. Let user fires a query as:

Q: Concept of page ranking algorithms in web mining.

First, the query processing engine extracts the keywords from the query which are listed below:

Concept, page, ranking, algorithms, web, mining

Now, the similarity score between the query terms and the cluster keywords is computed using (4.1) as show in Table 5.10.

Table 5.10 Similarity Value between Clusters and Query Terms

Cluster No.	S.No	Paper Title	Keywords	Similarity
I	A	Page Ranking Algorithms for Web Mining	web, mining, rank, algorithms, page, ranking, structure, link, categories, content, weighted, algorithm	0.56
	B	Comparative study of Page Ranking Algorithms for Web Mining		
	C	A Survey- Link Algorithm for Web Mining		
	D	Analysis of Various Web Page Ranking Algorithms in Web Structure Mining		
	E	Application of Page Ranking Algorithm in Web Mining		
	F	Web Mining Research: A Survey		
II	G	Web Crawler Architecture	Web, crawler, architecture, application, crawling, historical, background, foundation, key, future, directions, search	0.21
	H	How search engines work and a web crawler application		
	I	Mercator: A scalable, extensible Web crawler		
	J	Web Crawler: Extracting the Web Data		

After analyzing the Table 5.10 data, it is concluded that, the cluster I is the matched for forming the result set of the query fired. Now, the papers in the matched cluster will be rearranged according to dynamic rank which is discussed in next Chapter.

5.7 SUMMARY

A multi-level indexing method is proposed which maintains the primary and secondary index of the document corpus. Through primary index, general keyword based search is performed to retrieve the results whereas through secondary index, results are retrieved based on category and cluster. By using the multi-level approach, an efficient index structure is maintained. A comparison summary of this proposed method with existing methods is also described in Table 5.11.

The next chapter describes in detail the proposed ranking technique, the technique developed for front end of the digital library search engine for result representation.

Table 5.11 Comparison of Indexing Techniques

Technique →	Indexing Technique using Hierarchical Clustering [91]	Trie Structure based Indexing [94]	Context based Indexing using Ontology [93]	Sentence Context Ontology based Indexing [96]	Proposed Indexing Mechanism
Measures ↓					
Main Technique Used	Agglomerative Hierarchical clustering algorithm is used by the system in order to keep the information based upon similarity measure and fuzzy string matching.	This method keeps the context related information integrated with the frequency of the keyword. The structure is implemented using Trie.	An index is built on the basis of context of the document rather than on the terms basis using ontology.	A linked data application is developed which provides intelligent information services using the extracted information from research articles using Citation Context Analysis, Conditional Probabilistic Models and Semantic Web for modeling Scientific Discourse	The system provides a sequential as well as direct access of documents stored in the index. Also, the documents are clustered on the basis of category, which further provides more refined results to the user query.
Type of Indexing	Agglomerative Hierarchical clustering based indexing	Contextual based indexing	Context based indexing using Ontology	Citation Indexing	Category and Clustering based Indexing
Data Structure Used	Inverted Index	Trie type tree structure	Simple inverted index	Graph based Structure	Multi-level Inverted Index
Advantage	The related documents are grouped in the same cluster so that searching of documents becomes more efficient in terms of time complexity.	It helps to optimize the speed and performance in finding relevant documents for a search query.	Fast access to documents.	Classification of the citations. Evaluation of the citation analysis based on the different contexts of citations to the cited works and the author timeline.	Multi-level index structure is used in which first documents are classified based on category and then grouped into cluster based on similarity.
Limitations	The complexity of this method is $O(n^3)$ which makes it very slow for large databases.	More space is required to store trie structure for large dataset.	No consideration of ambiguity if the user is not aware of the context.	A larger training dataset is required with a focus on achieving a higher accuracy,	More space is required.

Chapter VI

SEARCH RESULTS REPRESENTATION USING CLUSTERING AND RANKING

6.1 GENERAL

Now a days', due to exponential growth of digital documents in digital libraries, the task performance of extracting and ranking the relevant documents as per user query is degrading gradually. Hence, there is a need to employ some methods or techniques to find, extract, filter and order the desired information. Ranking mechanism plays an important role in digital libraries as it enables the user to find the desired document easily and efficiently. Various ranking algorithms have been proposed in the literature [103, 104, 107, 108, 112, 113, 115, 116] based on different measures like number of citations to a research paper, content of paper, impact factor of publication, venue, year of publishing, bookmarks etc. But, these existing ranking algorithms (as discussed in Chapter III) sometimes provide irrelevant results due to certain shortcomings, which indicate a scope for further improvement in ranking mechanisms. In this paper, a ranking mechanism is proposed that carries out static as well as dynamic ranking to rank the documents in digital libraries. The proposed algorithm considers the citations of the paper, bookmarks of the paper, user's feedback and clustering process for ranking. This approach is explained in detail, which uses Web Content, Web Structure as well as Web Usage Mining to display an ordered search result list in cluster form in accordance with the user interest.

6.2 PROPOSED APPROACH FOR RANKING DOCUMENTS

The proposed approach considers the bookmarks and citations of the papers as an input. Bookmarks are the set of keywords that describe the complete content of the paper and citations are the references of the paper that describes the links, to the paper (backlinks). Here, a list of digital library search results is returned to the user as a hierarchy of clusters relevant to user query. Moreover, the papers within the each cluster are ranked as per their relevancy. This type of search result organization helps the user to limit his search

within the cluster having high query-cluster similarity score instead of searching in a long list of results.

The entire process (outlined in Fig. 6.1) from giving the user query to getting the results can be explained by the following steps:

- Similarity Matrix and Clusters Generation
- Static Rank Calculation
- Dynamic Rank Calculation

These steps have been explained in detailed in subsequent sections.

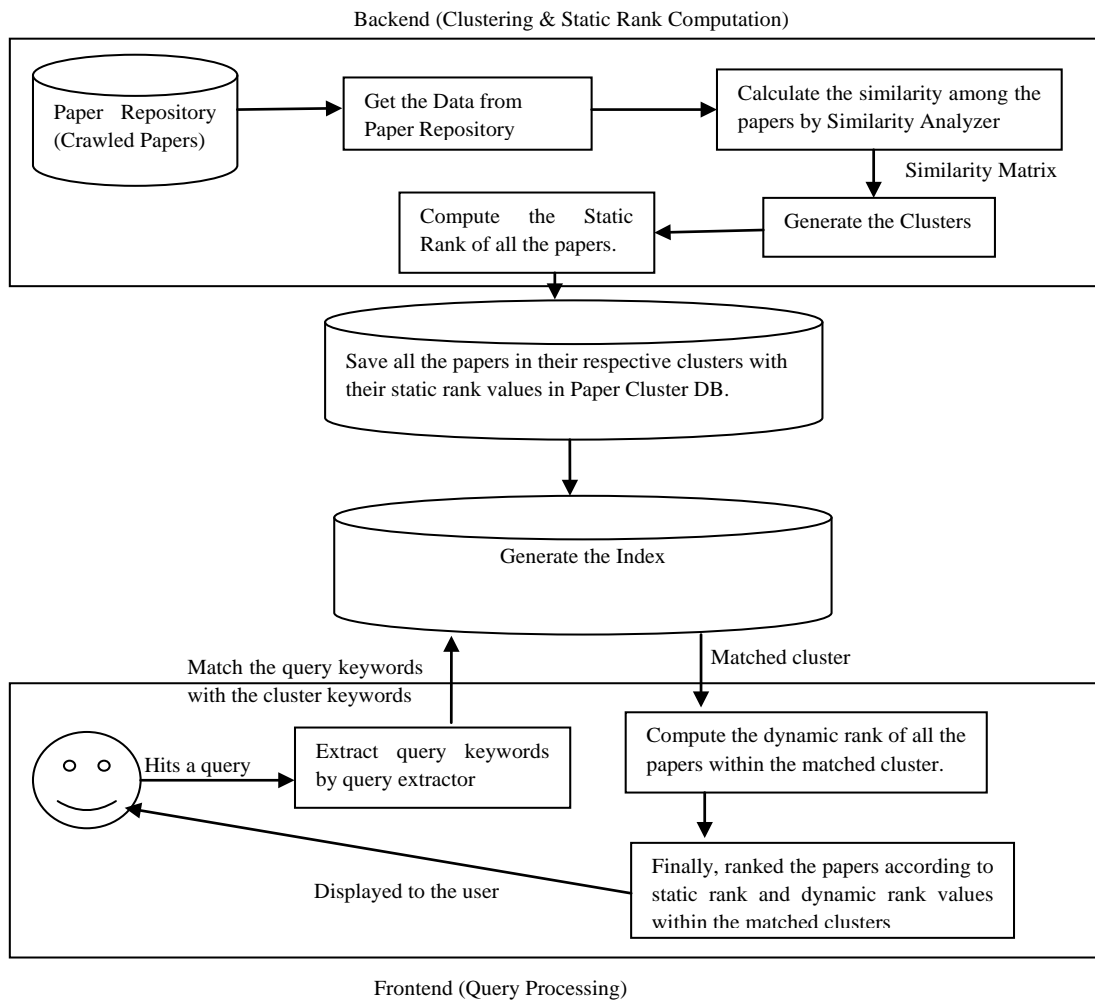


Fig 6.1 Workflow of the System

6.3 SIMILARITY MATRIX AND CLUSTERS GENERATION

In this step, the similarity between the publications is computed by considering the weight of the keywords or terms present in the documents. For comparison, cosine similarity score is used. After computing the similarity values, the method generates the groups of the similar publications, called as clusters. Finally, the returned clusters are stored in the Paper Cluster Database. The detailed working of this step has already been described in Chapter V.

6.4 STATIC RANK CALCULATION

The proposed clustering and ranking approach [166] considers three parameters named as *Download Score*, *PageRank* and *Bookmark based Citation Count* for computing the static rank of each paper or document in the cluster. The final static rank of each paper is computed by using (6.1) is stored in the clustering database along with the paper.

$$\text{StaticWeight} = \text{DownloadScore} + \text{BCC_rank} + \text{PageRank} \quad (6.1)$$

The papers within each cluster are rearranged on the basis of this static weight.

These three parameters are described in details as below:

6.4.1 Download Score

This parameter extracts the number of downloads of any paper (from user logs or search log) to compute the download score for each paper in the cluster. Download score of paper P is calculated by using (6.2):

$$\text{Download Score}(P) = \frac{\text{Number of Downloads}(P)}{\text{Maximum Downloads}} \quad (6.2)$$

Here, maximum downloads represents the number of download of any paper with highest number.

The search logs [162] constructed by the search engines act as a good resource for recording users' search histories and the necessary information about users' browsing behavior over the search results. An entry in the log records every single access made by users corresponding to their search queries. Thus, a log mainly contains users' queries

and corresponding visited documents or URLs, as well as other information about their browsing activities. The click-through or download patterns stored in the logs can capture derivative traces, which can further be utilized to characterize the users and their interests.

A typical search log [162] can be regarded as a file consisting of a series of requests, wherein a request consists of a number of fields. The format of search log is shown in Fig 6.2.

<i>UserID</i>	<i>Date</i>	<i>Time</i>	<i>Query</i>	<i>User_Agent</i>	<i>Clicked URL</i>
---------------	-------------	-------------	--------------	-------------------	--------------------

Fig 6.2 Format of Search log

The important fields are outlined below along with their description:

- *User ID*: the IP address of the client's computer. This is sometimes also an anonymous user code address assigned by the search engine server.
- *Date*: The date of the interaction as recorded by the search engine server.
- *Time*: The time of the interaction as recorded by the search engine server.
- *Query*: The query terms as entered by the user.
- *Clicked URL*: The documents or URLs clicked or downloaded from the search result list by users.

During the searching phase of the proposed system, whenever a user download any document, then a count is generated corresponding to that document which represents the number of times that document is downloaded by the users. This count is taken as an input to compute the download score.

A sample fragment of search log used by proposed digital library search system is shown in Appendix B.2.

6.4.2 PageRank

Page rank of the paper is calculated by using the *PageRank Algorithm* [10, 36, 106]. This method computes the rank of a paper by considering the number of citations (i.e.

backlinks) of the paper. This algorithm states that if a link comes from an important paper then this link is given higher weightage than those which are coming from non-important papers. These links are called as backlinks. The PageRank of a paper P can be calculated as:

$$PR(P) = (1-d) + d \sum_{Q \in B(P)} \frac{PR(Q)}{N_q} \quad (6.3)$$

where P represents a paper, $B(P)$ is the set of papers that point to P , $PR(P)$ and $PR(Q)$ are rank scores of papers P and Q respectively, N_q denotes the number of outgoing links of paper Q , and d is a normalization factor usually set to 0.85.

Illustrative Example: Let us take an example as shown in Fig 6.3 in order to explain the working of PageRank algorithm. Here, consider six papers denoted by A, B, C, D, E and F. The PageRanks for papers can be calculated by using (6.3):

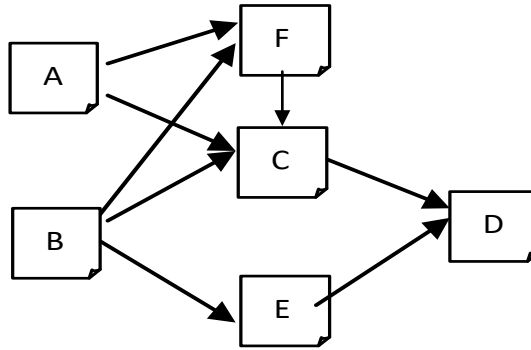


Fig 6.3 Citation Graph

$$PR(A) = (1 - d) + d(0) \quad (6.3a)$$

$$PR(B) = (1 - d) + d(0) \quad (6.3b)$$

$$PR(C) = (1 - d) + d \left(\frac{PR(A)}{2} + \frac{PR(B)}{3} + \frac{PR(F)}{1} \right) \quad (6.3c)$$

$$PR(D) = (1 - d) + d \left(\frac{PR(C)}{1} + \frac{PR(E)}{1} \right) \quad (6.3d)$$

$$PR(E) = (1 - d) + d \left(\frac{PR(B)}{3} \right) \quad (6.3e)$$

$$PR(F) = (1 - d) + d \left(\frac{PR(A)}{2} + \frac{PR(B)}{3} \right) \quad (6.3f)$$

Let us assume the initial PageRank as 1 and d is set to 0.85. The rank values of papers are iteratively substituted in above page rank equations to find the final values until the page ranks get converged as shown in Table 6.1.

Table 6.1 Iteration Method for PageRank

Iterations	PR (A)	PR (B)	PR (C)	PR (D)	PR (E)	PR (F)
0	1	1	1	1	1	1
1	0.15	0.15	1.106	1.090	0.192	0.256
2	0.15	0.15	0.474	0.552	0.192	0.256
3	0.15	0.15	0.474	0.552	0.192	0.256

The final page ranks of papers represent the following ordering:

$$PR (D) > PR (C) > PR (F) > PR (E) > (PR (A), PR (B))$$

6.4.3 Bookmark Based Citation Count Rank

This method takes the content of the paper which cited the publication or paper along with the number of citations as an input. In this algorithm [168], the relevancy score between the main paper and the paper which cited the main paper is computed on the basis of their content. To check the relevancy between papers, it uses the bookmarks instead of comparing the whole content of the papers. For comparison, cosine similarity measure [128, 130] is used as given in (4.1).

Bookmark based Citation Count i.e. BCC_ Rank of any paper P is computed as:

$$BCC_{Rank}(P) = \frac{\sum_{Q \in B(P)} Sim(P, Q)}{|B(P)|} \quad (6.4)$$

where $B(P)$ is the set of all papers which cited paper P . This rank calculates the score on the basis of ratio of total similarity score between main paper and the backlinked paper to the total number of back linked papers.

- **Illustration of Proposed Algorithm**

An example is taken to explain the bookmark based citation count ranking. The citation graph of existing papers in the database is shown in Fig 6.4. Assume that a paper B is selected to compute the BCC rank.

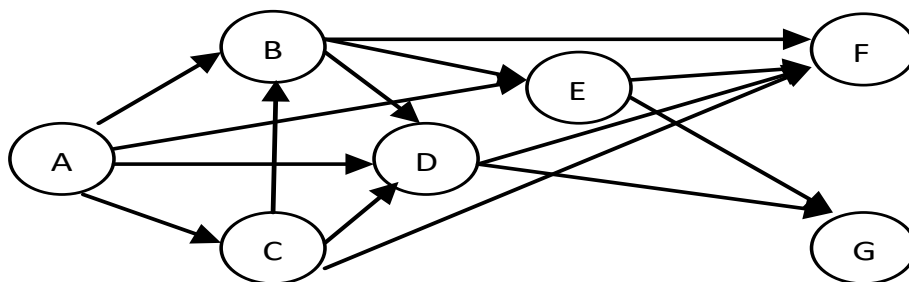


Fig 6.4 Citation Graph of Papers

First, the system extracts the bookmarks of the paper B as shown in Table 6.2. From these bookmarks, keyword extraction module selects the important keywords on the basis of their occurrence after applying stop word removal and stemming.

Table 6.2 Bookmarks of the Research Paper B

Network security: it's time to take it seriously
Introduction
Network security
Differentiating data security and network security
History of network security
Brief history of internet
Security timeline
Internet architecture and vulnerable security aspects
Ipv4 and ipv6 architectures
Attacks through the current internet protocol ipv4
Security issues of ipv6
Security in different networks
Current developments in network security
Hardware developments
Software developments
Future trends in security
Conclusion
Future scope of work
Acknowledgment
References

The top ten selected keywords along with their frequency are shown in Table 6.3.

Now, to calculate the BCC rank of the selected paper B, system extracts the research papers having citations to it. As shown in Graph, Paper A and C cited the selected paper B. Now, system extracts the top ten keywords from the bookmarks of A & C respectively with their frequency of occurrence in each paper as shown in Table 6.3.

Table 6.3 Frequency of Keywords

Keyword of B	Frequency	Keyword of A	Frequency	Keyword of C	Frequency
Security	9	Security	11	Security	2
Network	4	Network	6	Basic	2
Internet	4	Internet	5	Attacks	2
Architecture	3	History	3	Network	1
IPv4	3	IPv4	3	Tools	1
Developments	3	IPv6	3	Techniques	1
History	2	Architecture	2	Types	1
IPv6	2	Attacks	2	Tips	1
Future	2	Current	2	Introduction	1
Time	1	Protocol	2	Security	2

Following calculations are done to compare these two sets of keywords by using cosine similarity:-

$$\begin{aligned}
 Sim(B,A) &= \frac{9 * 11 + 4 * 6 + 4 * 5 + 3 * 2 + 3 * 3 + 2 * 3 + 2 * 3}{\sqrt{9^2 + 4^2 + 4^2 + 3^2 + 3^2 + 3^2 + 2^2 + 2^2 + 2^2 + 1^2} * \sqrt{11^2 + 6^2 + 5^2 + 2^2 + 3^2 + 3^2 + 3^2}} \\
 &= \frac{170}{\sqrt{153} * \sqrt{213}} = \frac{170}{180.38} = 0.924
 \end{aligned}$$

Similarly, paper B is compared with paper C also.

The cosine similarity scores obtained after comparison of bookmarks of citations with bookmarks of selected paper are shown in the Table 6.4:

Table 6.4 Cosine Similarity Values of Paper with Citation

Paper Title	Cosine Similarity score
Network Security: History, Importance, and Future (A)	0.924
Network Security Attacks Solution and Analysis (C)	0.497

The BCC rank of paper B can be calculated by using (6.4):

$$\begin{aligned}
 BCC_{rank}(B) &= \frac{\text{Total Cosine Similarity Score}}{\text{Total number of cited paper}} = \frac{0.924 + 0.497}{2} \\
 &= \frac{1.42}{2} = 0.71
 \end{aligned}$$

The BCC rank of the paper B comes out to be 0.71. Similarly, BCC ranks of other papers can be calculated.

- **Comparison Study**

The proposed Bookmark based Citation Count Ranking (BCC) method ranks the retrieved papers in order to organise them in an efficient and user friendly manner as opposed to the ordered list returned by Citation Count (CC) [103] and PageRank (PR) [10, 36, 106]. If only ranking is concerned, the proposed BCC algorithm is an iterative algorithm but unlike CC and PR, it uses both the link structure and content of the citations of documents and as a result, it returns relevant as well as important papers on the top of the digital library search result list. Rather considering similarity of the papers itself, it uses the similarity of the citations with the paper to find the rank of the concerned paper. The comparison summary of the three ranking algorithms CC, PR and BCC is given in Table 6.5.

Table 6.5 Comparison Study of CC,PR and BCC

Algorithms → Measures ↓	Citation Count (CC) [103]	PageRank (PR) [10, 36, 106]	Bookmark Based Citation Count (BCC)
Main Technique used	Web structure mining	Web structure mining	Web Structure Mining, web content mining
Description	Results are sorted based on number of incoming citations.	Papers are sorted according to the link structure of the papers and citations to the paper.	
Input Parameters	Backlinks	Backlinks	Bookmarks, query's content
Working Level	1	N*	N*
Degree of Relevancy with Query	Does not check the relevancy with the query.	Does not check the relevancy with the query.	Checks the relevancy with the query
Different Scanning Options	No Scanning.	No Scanning.	Scans only the Bookmarks the Paper.
Importance	Simplicity of computation. It is proven method which has been used for many years in scientometrics.	Traditional method that focuses on the link structure to determine relevance.	To compute the similarity, no needs to scan the whole paper only bookmarks are compared.
Limitations	Unweighted ranking i.e. it treats all the citations equally and does not take into account the time.	Results obtained at the time of indexing and not at the query time.	More space and time complexity is required because of computing ranks on the fly.

*N: number of papers

- **Retrieval of Relevant Papers by BCC**

The crawler passes the parsed downloaded papers along with their static ranks calculated using (6.1) to the indexer for indexing the papers. When some user submits his query to the digital library search interface, the query processor matches the query keywords in the index and retrieves a set of papers with pre-assigned static rank value, which are further passed to dynamic ranking module (see section 6.5) to calculate the final rank. The user now can find the more desired and relevant papers in the first few pages of the search result list.

Following are the advantages of BCC:

1. As BCC method uses content similarity along with link structure of papers and their access information, the top returned papers in the result list are supposed to be highly relevant to user information needs.
2. The rank value of any paper by PageRank method will be same either it is seen by user or not because it is totally dependent upon link structure of citation graph. While the ordering of papers using BCC is more target-oriented because it also considers the content similarity within citations.
3. In BCC, a user can not intentionally increase the rank of a paper by citing the paper itself i.e. self-citations because the rank of the paper depends on the similarity among the citations (not only on the number of citation).

6.5 DYNAMIC RANK CALCULATION

Dynamic rank means the rank given to the returned papers on the fly i.e. on the basis of submitted query. This phase takes the matched cluster as an input which is extracted by query processing engine against the user query. The dynamic rank is computed based on the similarity between the user's query and papers within the matched cluster. Thus, the dynamic rank of any document or paper is described as the similarity [128, 130] between the query q and paper d which is calculated by using (6.5):

$$\text{Dynamic}_{\text{rank}}(d) = \text{sim}(q, d) = \frac{\sum W_{q,j} \times W_{d,j}}{\sqrt{\sum W_{q,j}^2} \times \sqrt{\sum W_{d,j}^2}} \quad (6.5)$$

where $W_{q,j}$ and $W_{d,j}$ denotes the weight of term t_j in the query q and paper d respectively. These weights can be computed by calculating the frequency of occurrence of term t_j in q and d .

Finally the papers within the matched clusters are ranked and returned to the user based on the static rank and dynamic rank.

$$Rank(p) = StaticWeight(p) + DynamicRank(p) \quad (6.6)$$

An example illustration of dynamic rank is described in next section along with the static rank computation.

6.6 ILLUSTRATION OF PROPOSED CLUSTERING AND RANKING MECHANISM

Let's take an example of paper database (as shown in Table 6.6) to explain the ranking

Table 6.6 Final Rank Values

Cluster No.	Paper Id	Paper Title	Down-load Score	Page Rank	BCC	Static Weight	Sim (q,c)	Dynamic Rank	Rank
C1	A	Page Ranking Algorithms for Web Mining	0.9	0.192	0.037	1.129	0.566	0.752	1.881
	D	Empirical study of ranking Algorithms for Web Mining	0.7	0.349	0.047	1.096		0.223	1.319
	F	Analysis of Web Page Ranking Algorithms in Web Structure Mining	1	0.564	0.022	1.587		0.748	2.33
	G	Web Mining Research: A Survey	0.8	0.15	0	0.95		0.549	1.49
C2	B	Web Crawler Architecture	0.9	0.15	0	1.05	0.213	Not Calculated as C2 is not matched.	
	C	How search engines work and application of a web crawler	0.8	0.256	0.074	1.130			
	E	Mercator: A scalable, extensible Web crawler	0.7	0.502	0.054	1.256			

mechanism of the proposed algorithm. Here A, B, C etc denote the papers in the database.

First, the similarity analyzer will compute a similarity score between the already existing papers in the database and form the cluster (as described in Chapter V). On the basis of the similarity matrix, let's assume two clusters are formed as shown in Table 6.6. After the generation of clusters, clusters are saved in the cluster database along with most frequently occurred set of keywords as shown in Table 6.7.

Table 6.7 Keywords Attached to Each Cluster

Cluster No.	Keywords
C1	web, mining, rank, algorithms, page, rank, structure, link, categories, content, weighted, algorithm
C2	Web, crawler, architecture, application, crawl, historical, background, foundation, key, future, directions, search

Static Ranking: Static ranking mechanism is performed for computing the weight for each paper within a cluster. For static ranking, Download Score, BCC and Page Rank of each paper are computed by using (6.2), (6.4) and (6.3) as shown in Table 6.6 In this example, the maximum number of downloads is assumed to be 10. Finally, the static rank is computed by adding all these three parameter as shown in Table 6.6.

Now, Assume user fires a query as:

Q : “*Various Page Ranking Algorithms*”

The query keyword extractor extracts the keywords from the user's query which are listed below,

Query Keywords: various, page, rank, algorithm.

Now, the system extracts the most relevant cluster from the database against the user query by comparing the query keywords with the keywords of cluster. The similarity score between the query and the cluster keywords is computed by using (4.1) is also shown in Table 6.6.

Clearly, it can be seen that the cluster *C1* is the suitable cluster for forming the result set of the query fired. The papers in the *C1* will be re-ordered according to the dynamic rank

Table 6.8 Final Result Set against the User's Query

S.No	Paper Title
F	Analysis of Various Web Page Ranking Algorithms in Web Structure Mining
A	Page Ranking Algorithms for Web Mining
G	Web Mining Research: A Survey
D	Empirical study of Ranking Algorithms for Web Mining

(as shown in Table 6.6) and will be displayed to the user as search result set. The final ordered result set provided to the user is shown in Table 6.8.

6.7 COMPARISON STUDY

A critical look at the available literature concluded that each algorithm has some relative strengths and limitations. The proposed approach ranks the results in order to organize them in an efficient and easily accessible manner as compared to Citation Count (CC), PageRank (PR) and Content based Citation Count (C3) algorithms. Proposed approach considers combination of all three mining i.e. Web Content, Web Structure and Web Usage mining for ranking the more relevant results at the top of search result list as compared to PR, CC and C3. The comparison of the proposed ranking mechanism with three ranking algorithms CC, PR and C3 based on different parameters is shown in Table 6.9.

Table 6.9 Comparison between CC, PR, C3 and Proposed Approach

Algorithms →	Citation Count (CC) [103]	PageRank (PR) [10, 36, 106]	Content Based Citation Count (C3) [110]	Proposed Ranking (Clustering and Ranking Method)
Measures↓				
Main Technique Used	Web Structure Mining	Web Structure Mining	Web Structure Mining, Web Content Mining	Web Structure Mining, Web Content Mining, Web Usage Mining, Clustering
Description	Results are ranked by considering the number of incoming citations.	Computes scores at indexing time. Results are sorted by taking into account the importance of citing papers.	Rely on links as well as content of the paper.	Results are ranked by taking into account the link structure as well as content similarity among the papers. It also involves clustering of papers for enhancing the results.
I/P parameters	Backlinks	Backlinks	Backlinks and Summary of the publication	Bookmarks, query's content, paper posted time, number of downloads

Relevancy	Less	Less(more than CC, Time dependent Citation Count)	Medium	High
Quality of Results	Less	Medium	High	High
Importance	Simplicity of computation.	It statistically analyses whole citation graph at once. It captures not just quantity, but also quality of citing papers.	The rank of the paper is calculated on the basis of citations to the paper and content of the paper.	User will get the results in the form of sorted order of papers within the cluster.
Limitations	It considers all the citations equally.	Results come at the time of indexing and not at the query time.	More space and time complexity is required because of computing rank.	More complexity in terms of time and space.

6.8 SUMMARY

Proposed ranking technique is summarized in Table 6.10 from where it can be observed that ranking technique is targeted towards presenting relevant results to the user.

Table 6.10 Summary of Proposed Ranking Technique

<i>Parameters</i>	<i>Clustering Ranking Technique</i>
<i>Module Optimization</i>	Query Processor
<i>Metric</i>	Presenting the search results in the form of clusters with ranked pages within.
<i>Mined Web Resource</i>	Web Graph, Document Contents and User logs
<i>Type of Mining</i>	Web Content Mining, Web Usage Mining and Web Structure Mining
<i>Advantages</i>	User search space will be reduced as he can direct his search to a fraction of documents in a particular cluster of his interest.

The proposed crawling, indexing and ranking techniques were implemented and test run was carried on some sample citation graphs and user logs. The implementation details and the results obtained thereof are discussed in the next chapter.

Chapter VII

IMPLEMENTATION RESULTS AND ANALYSIS

7.1 GENERAL

A Unified Digital Library Search System has been developed in this work that overcomes the problem of relevant document retrieval by mechanizing the process of crawling, indexing, ranking and query processing of digital library search engines.

The proposed techniques have been implemented and their result analysis has been carried out. The following sections provide in detail the data set and the experiments conducted for evaluation of different techniques.

7.2 PERFORMANCE METRICS

There are three performance metrics that have been used for performance analysis of proposed approaches, namely Precision, Recall and F-measure. These metrics, are defined below.

- a) **Precision (P)**: It is defined as a fraction of retrieved documents/ publications that are relevant to the query.

Mathematically, Precision is given by:

$$P = \frac{RD}{(RD + WRD)} \quad (7.1)$$

where RD is the number of relevant documents and WRD is the number of irrelevant i.e. $RD+WRD$ represents the total number of retrieved documents.

- b) **Recall (R)**: It is defined as a fraction of relevant documents or publications that are successfully retrieved by the digital library search system.

Mathematically, Recall is given by:

$$R = \frac{RD}{(RD + NRD)} \quad (7.2)$$

where RD is the number of relevant documents and NRD is the number of relevant documents which are not retrieved i.e. $RD+NRD$ represents the total number of relevant documents presents in WWW.

- c) **F-measure (F)**: Mathematically, it combines both precision and recall.

F-measure is given by:

$$F = \frac{2PR}{(P + R)} \quad (7.3)$$

where an equal weight is assigned to both P and R .

7.3 EXPERIMENTAL EVALUATION OF PROPOSED CRAWLER

For the implementation of the proposed crawling technique, Java JDK 6.0, mySql 5.6, Apache PDFBox 1.8.9 and WordNet Version 3.0 is used. Experiments are performed on Dual-Core Intel Pentium IV or higher Processor with 2.60GHz frequency and 4.00 GB RAM. NetBeans IDE is used for the implementation of the proposed system.

A detailed discussion on the implementation and evaluation of proposed techniques is given in this section. Testing of the proposed crawler system was conducted and the home screen is shown in Fig. 7.1.

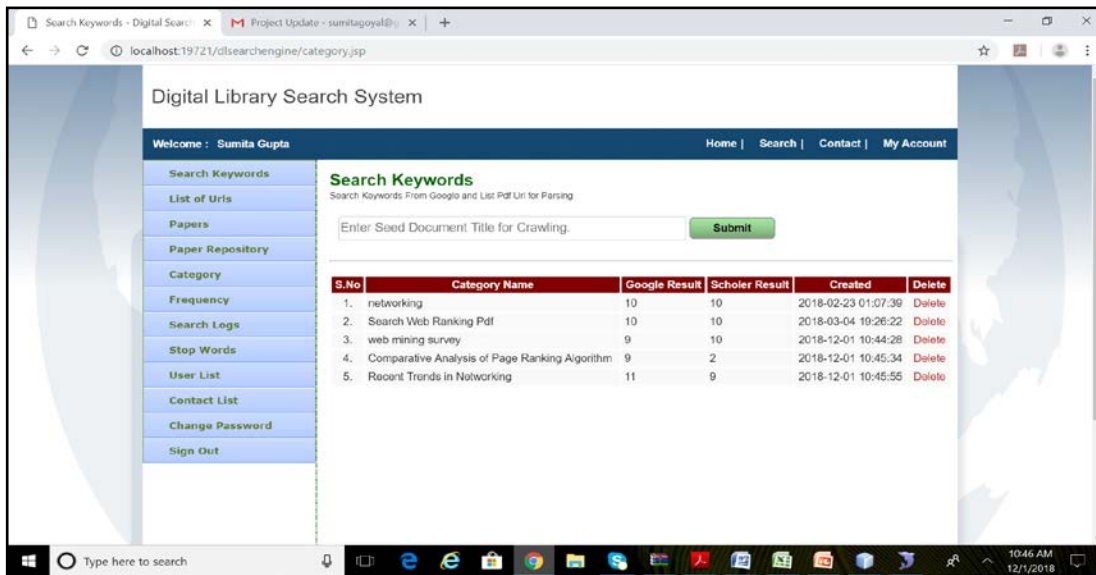


Fig 7.1 Home Page of Crawling System

The home page comprises of an input box for giving the seed URLs or document titles and a search button. On clicking the button, the result(s) are displayed on the Crawler interface. For analysis, the administrator can feed 10-15 seed document titles from the computer science field as shown in Fig 7.2. Here, a database of approximately 100 documents is crawled by the crawler.

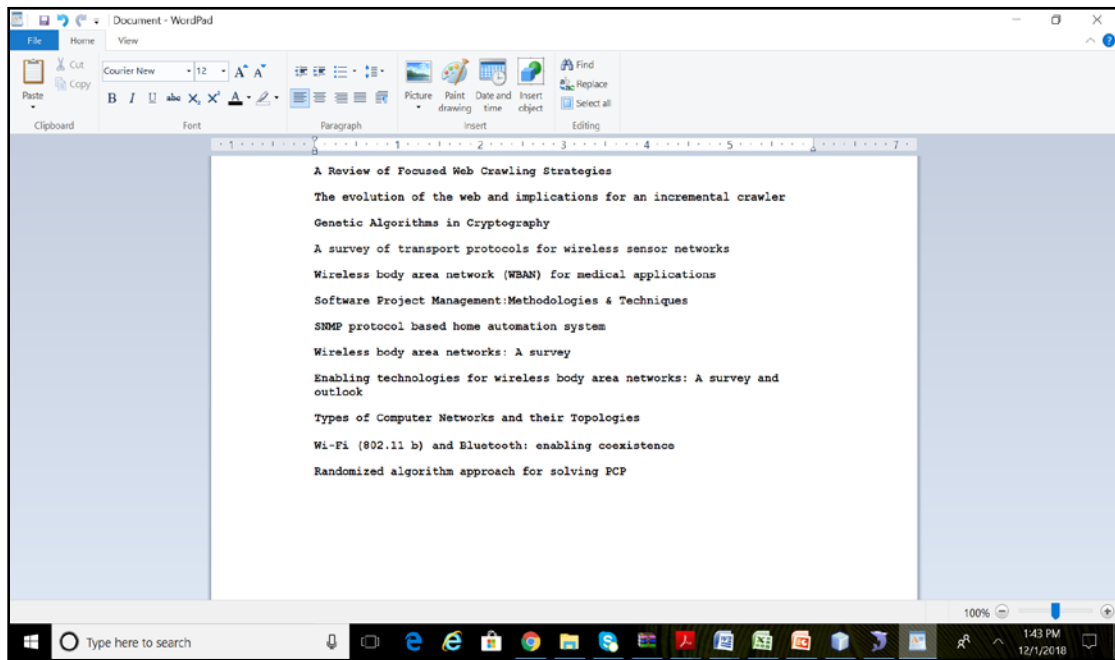


Fig 7.2 List of Seed Document Titles

The system first downloads, parses and then finds the category of the downloaded documents as shown in Fig. 7.3.

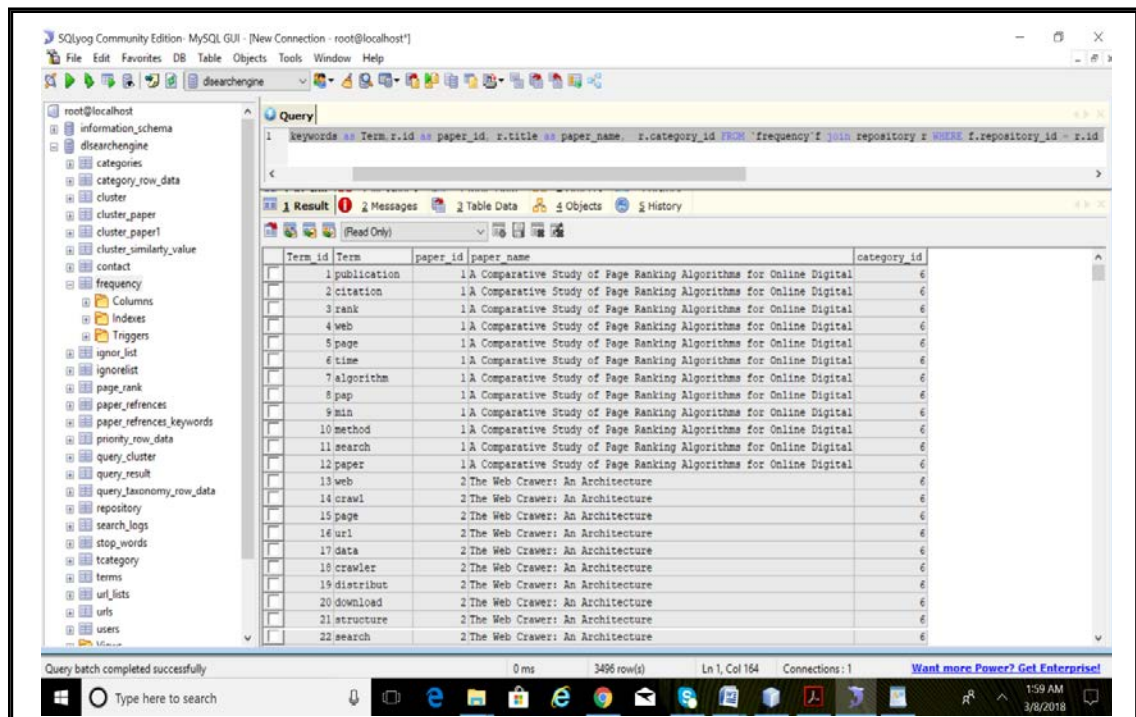


Fig 7.3 Paper Repository with Category Information

After this, the system extracts all the references of the downloaded document and finds the similarity score between extracted references and downloaded document (as shown in Fig 7.4).

id	paper_id	reference_id	cosine_similarity_value
1	1	2	0.62
2	1	4	0.40
3	1	6	0.35
4	1	8	0.62
5	1	10	0.53
6	3	13	0.00
7	3	15	0.67
8	3	17	0.83
9	3	19	0.83
10	3	21	0.83
11	4	23	0.77
12	4	25	0.48
13	4	27	0.83
14	4	29	0.53
15	5	31	0.85
16	6	33	0.46
17	7	35	0.85
18	8	37	0.36
19	8	39	0.50
20	8	41	0.56
21	9	43	0.85
22	9	45	0.49
23	10	47	0.34
24	10	49	0.50
25	10	51	0.27

Fig 7.4 Similarity Values Computed by Link Priority Analyzer

Depending upon these similarity values, the link priority analyzer assigns the priority to unvisited URLs and forwards them to priority queue for further crawling. In this way, crawler crawls the WWW for gathering the documents only in the case if pdf format of document is available. If pdf format of document is not available or pdf downloader is not able to download (in case of missing information or access authorities), then URL is processed by *Missing Document Finder Module*.

In *Missing Document Finder Module*, system first extracts the meta-data of references (as shown in Fig 7.5 and Fig 7.6) for framing multiple queries to be sent to different search engines such as Google and Google Scholar.

Fig 7.5 shows quoted and unquoted title types queries formed by extracting meta-data of missing documents and Fig 7.6 shows the author name’s queries formed by extracting meta-data of missing documents.

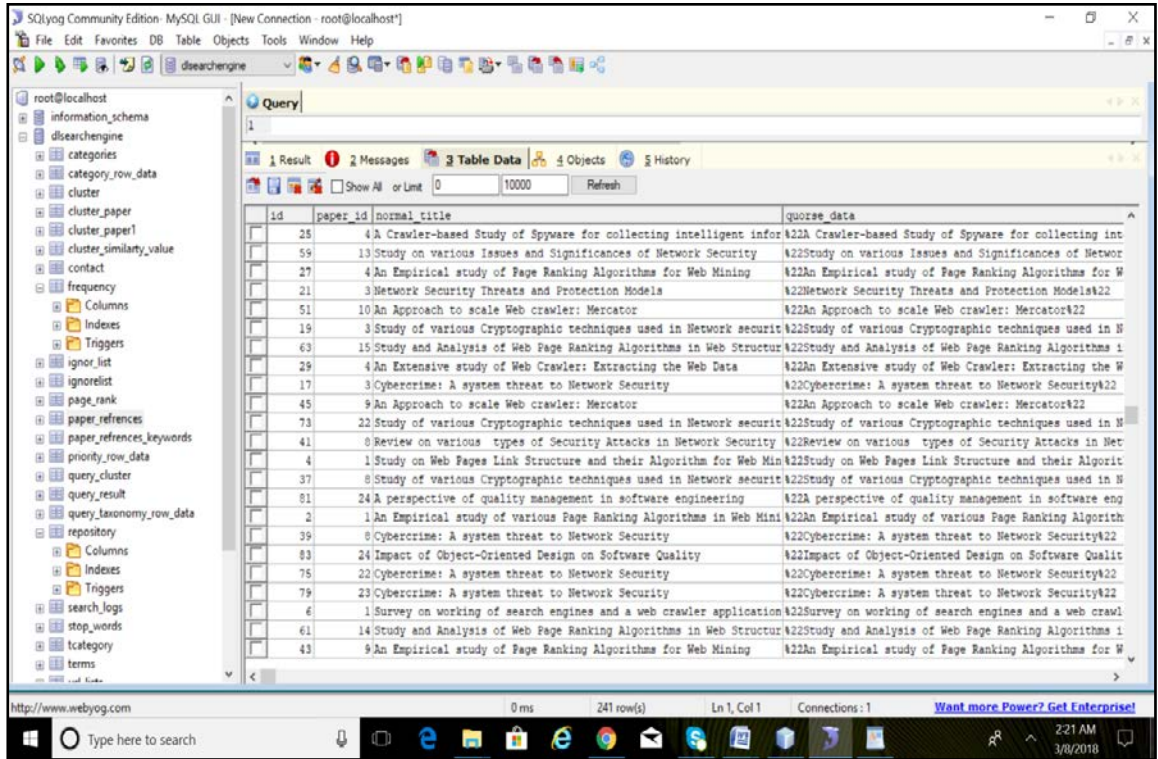


Fig 7.5 Extracting meta-data information (i.e. Title) of references for finding missing document

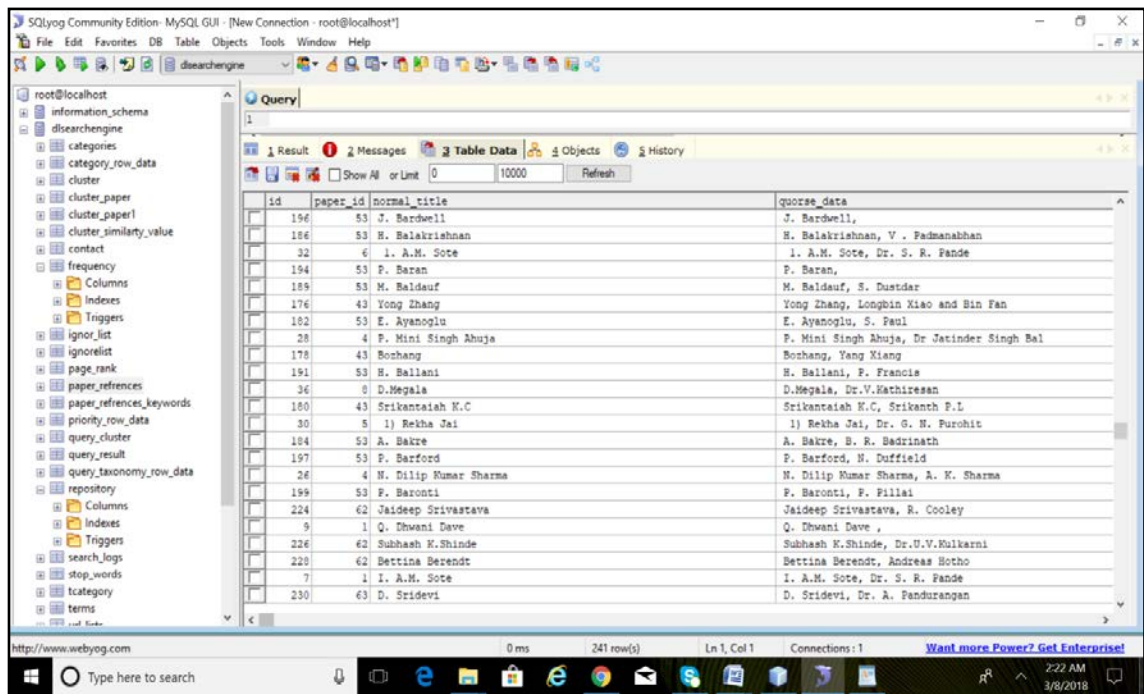


Fig 7.6 Extracting Author's Information of References for finding Missing Document

Fig 7.7 shows the result list obtained after hitting the unquoted title type query on search engine Google and Google Scholar.

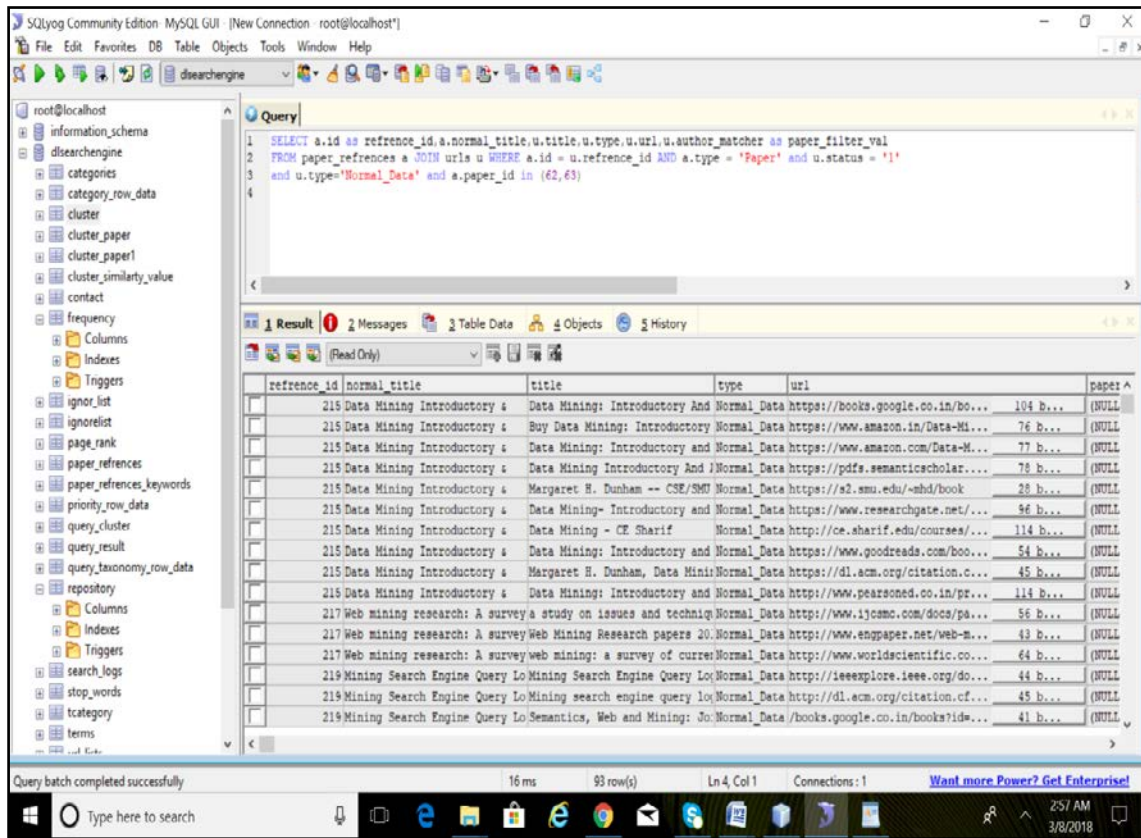


Fig 7.7 Results after Browsing Paper Title Query (i.e. without quotes) from Different Search Engines

The results of the proposed crawling approach are compared and analyzed based on user query with CiteSeer^x and Google Scholar.

For instance, if a researcher, when browses for research paper related to query “*Survey of Recent Web Prefetching Techniques*”, then the results screen after browsing in the proposed system is shown in Fig.7.8. The snapshot of the returned list of result papers after submitting the same query on CiteSeer^x’s interface and Google Scholar’s interface is shown in Fig 7.9 and Fig. 7.10.

As shown in Fig 7.8, Fig 7.9 and Fig. 7.10, proposed crawler system and Google Scholar finds the documents corresponding to user query whereas CiteSeer^x does not find the results corresponding to the query.

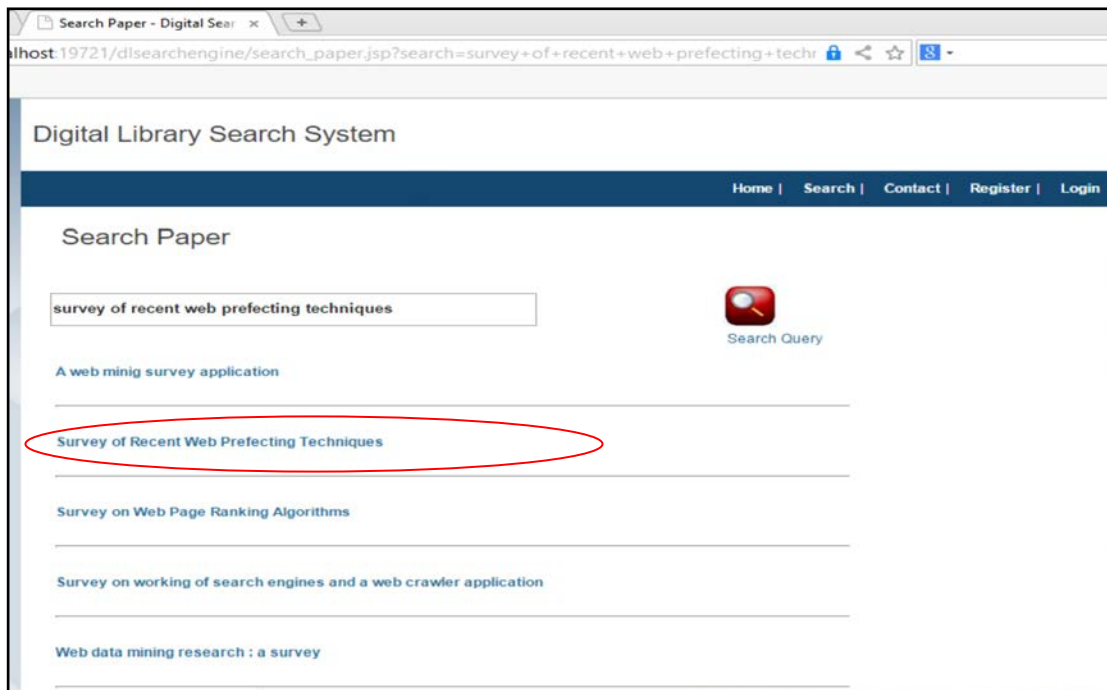


Fig 7.8 Result Screen of Proposed Crawler for query “Survey of Recent Web Prefetching Techniques”

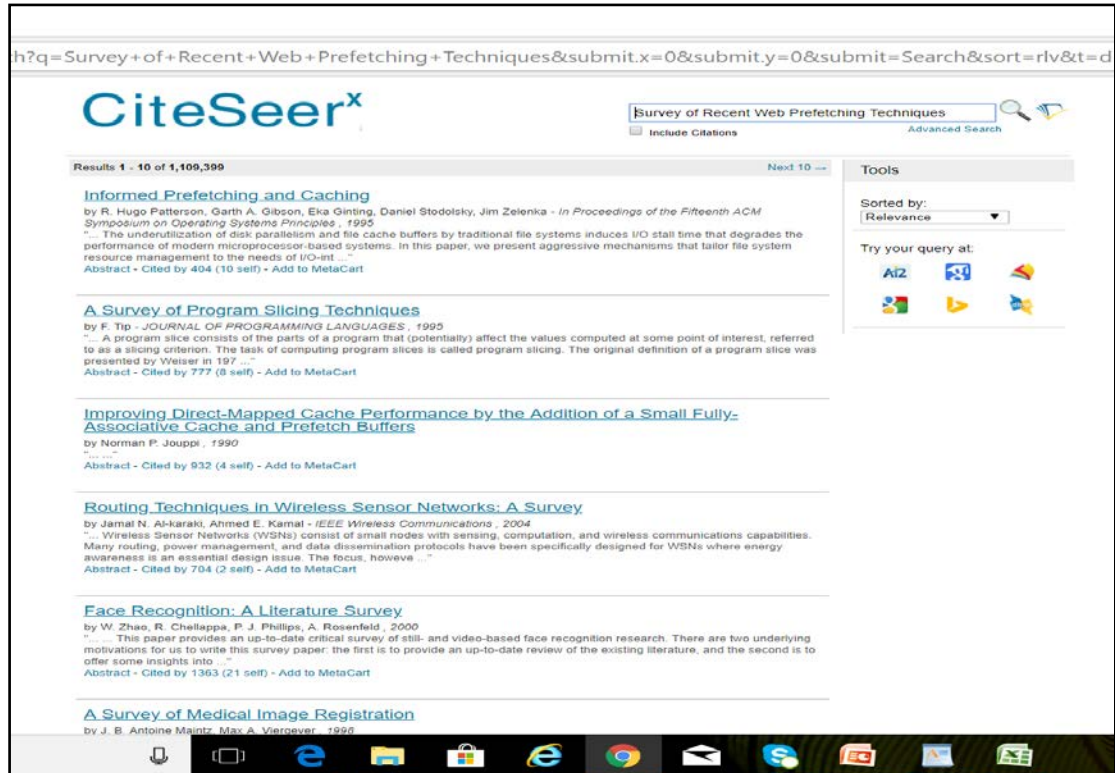


Fig 7.9 Result Screen of CiteSeerx for query “Survey of Recent Web Prefetching Techniques”

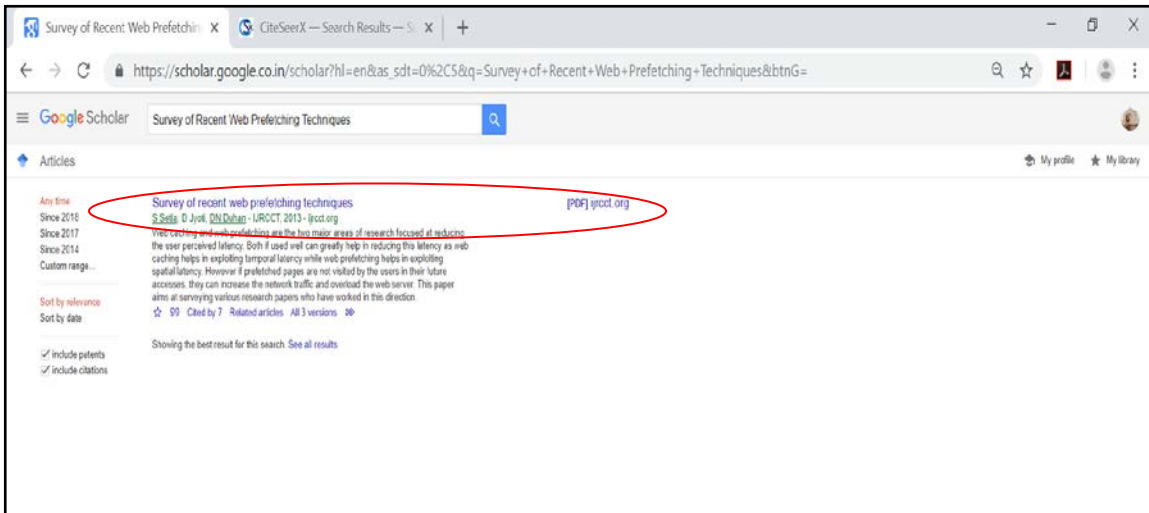


Fig 7.10 Result Screen of Google Scholar for query “Survey of Recent Web Prefetching Techniques”

Thus, it is observed that the proposed unified framework for Digital Library search System gives more precise results than existing approaches for the example scenario. By using multiple query formation techniques on different search engines, the proposed system harvest publisher’s site, author homepages and WWW efficiently. For the large size of database, the precision of proposed approach is even more than the existing systems.

For experimental analysis of the proposed focused crawler, list of seed document titles are given as:

- *Data Mining and their Applications*
- *Information Retrieval Techniques*
- *Network topologies and Ethernet*
- *Issues in Networking Protocol*

Now, the crawler starts the crawling process to download the documents from the list.

The runs of the proposed focused crawler and the process of experimental evaluation are given below.

On the first run of the focused crawler, it collected about 10 references corresponding to each of the assigned URLs, thus collectively a sample of 40 reference URLs is collected.

Out of 40 references that have been crawled, 31 references are the documents (in pdf format found) and 9 are the URLs or missing documents. There are 11 references that not crawled by our proposed crawler. So, using the terms defined above:

Thus, RD=31, WRD=9, NRD=11.

By using eq. 7.1, 7.2 and 7.3, values of precision, recall and F-measure are as:

$$P=31/(31+9)=77.5\%$$

$$R=31/(31+11)=75.6\%$$

$$\text{and } F=2*77.5*75.6/(77.5+75.6)=76.53\%.$$

Similarly, Focused crawler is run more time for analysis and the values of precision, recall and F-measure comes out to be shown in Table 7.1 and graphically shown in Fig 7.11.

Table 7.1 P, R and F values of Proposed Crawler

Run #	P(in %)	R(in %)	F(in %)
1	77.5	75.6	76.53
2	75	73.1	74.03
3	76	74.2	75

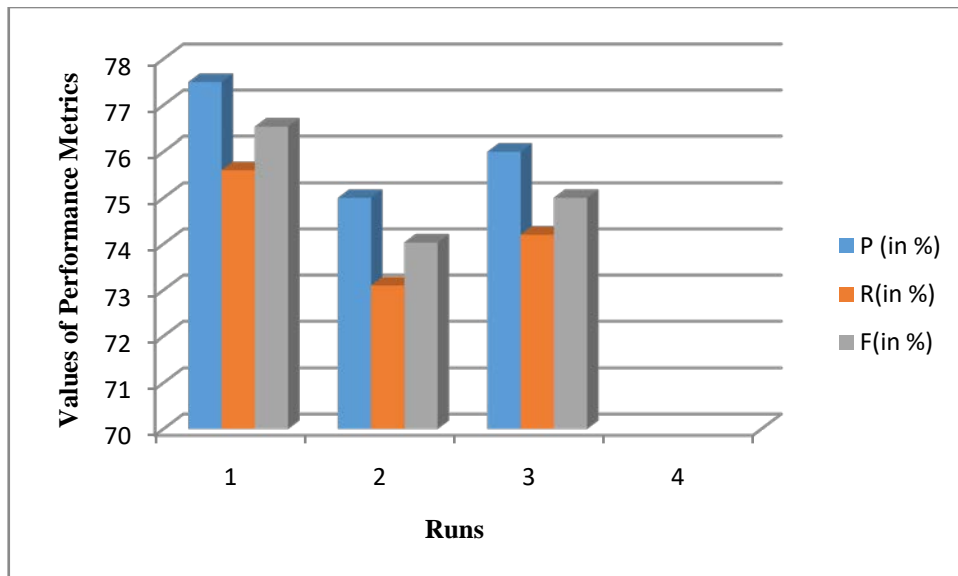


Fig. 7.11 P, R and F Values for each Runs of Proposed Crawler

7.4 IMPLEMENTATION RESULTS OF PROPOSED DIGITAL LIBRARY SYSTEM

Proposed digital library (DL) Search system has been implemented on Java JDK 6.0 and tested on the citation graph given in Appendix B. Home screen of the proposed system is shown in Fig 7.12. On clicking the search button, search interface of the proposed digital library system is displayed for the submission of a topical query by the user as shown in Fig 7.13.



Fig 7.12 Home Screen of Proposed Digital Library Search System

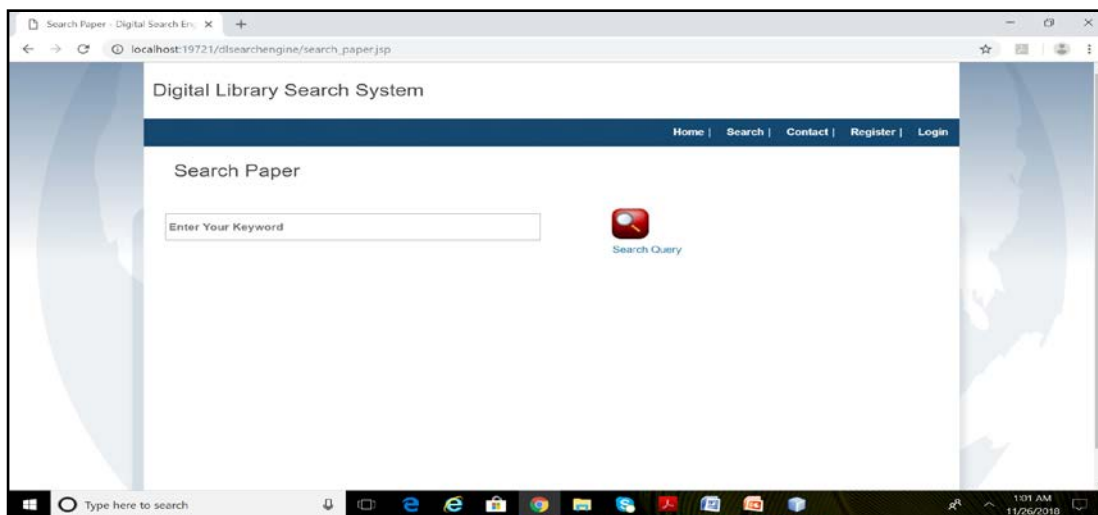


Fig 7.13 Search Interface of Proposed DL Search System

At the back end, for retrieving the relevant results as per user query, the system first generates the clusters based on the similarity values computed by the similarity analyzer as shown in Fig 7.14. Fig 7.15 shows the static rank values of documents computed at the backend which involves the parameters PR, Number of Downloads and BCC (described in Section 6.4). A log called search log (refer Section 6.4.1) is used to record the number of downloads of documents.

id	paper_id	similarity_paper_id	paper_name	cosine_similarity_value
1	1	1	1	1
2	1	1	2 The Web Crawler: An Architecture	0.46203210
3	1	1	3 Study on Network Security: History, Importance, and Future	0.24125579
4	1	1	4 An Empirical study of various Page Ranking Algorithms in Web Mini	0.58683190
5	1	1	5 Study on Web Pages Link Structure and Algorithms for Web Mining	0.63683105
6	1	1	6 Survey on working of search engines and a web crawler application	0.52075946
7	1	1	7 Network Security: it's a challenging to secure your network	0.24125579
8	1	1	8 Network Security Threats and Protection Models	0.34118724
9	1	1	9 Comparative study on various Application of Page Ranking Algorithm	0.44620530
10	1	1	10 Weighted Page Rank Algorithm Based on link structure of Web Page	0.61785020
11	1	1	11 A Crawler-based Study of Spymare for collecting intelligent infor	0.32952010
12	1	1	12 Study of various Cryptographic techniques used in Network securit	0.00000000
13	1	1	13 Cybercrime: A system threat to Network Security	0.24125579
14	1	1	14 An Empirical study of Page Ranking Algorithms for Web Mining	0.67983680
15	1	1	15 An Approach to scale Web crawler: Mercator	0.40359593
16	1	1	16 An Extensive study of Web Crawler: Extracting the Web Data	0.57367426
17	1	1	17 Study and Analysis of Web Page Ranking Algorithms in Web Structur	0.44620530
18	1	1	18 An Approach to design a High-Performance Distributed Web Crawler	0.8058426
19	1	1	19 Review on various types of Security Attacks in Network Security	0.00000000
20	1	1	20 Study on various Issues and Significances of Network Security	0.00000000
21	1	1	21 Review based on Application of Genetic Algorithms in Machine lear	0.34118724
22	1	1	22 CRYPTOGRAPHY AND GENETIC ALGORITHMS	0.34118724
23	1	1	23 An Approach to solve PCP by using Randomized algorithm	0.34118724
24	1	1	24 Study of representing a classical view of object-oriented cohesio	0.24125579
25	1	1	25 A perspective of quality management in software engineering	0.24125579

Fig 7.14 Cosine Similarity Values calculated by Similarity Analyzer

This log is updated in JavaScript, which records user's download events on the downloaded document as shown in Fig 7.15. The log is periodically accessed by static ranking module to find out ranks of various documents. When a query is submitted, the most recent calculated rank values are returned depending on the static and dynamic rank values.

On the front end, when a user fires a query then the query analyzer finds the category of query by comparing the query keywords with the keywords of categories as shown in Fig. 7.16. Then based on category, relevant cluster (s) is retrieved and the documents within the cluster are ranked as per the dynamic rank computed by query processing engine and results are displayed to the user.

id	category_id	title	file_path	no_of_download	page_rank	ppt	static_weight	incoming_link	outgoing_link	year_of_publication
1	4	A Comparative repository/75		0.15	0.00	0.15	0	4		2014
2	4	The Web Crawl repository/16		0.15	0.00	0.15	0	4		2014
3	7	Study on Netw repository/110		0.15	0.00	1.15	0	7		2014
4	4	An Empirical repository/13		0.20	1.	2.07	2	5		2011
5	4	Study on Web repository/87		0.17	1.00	1.84	1	5		2012
6	4	Survey on wor repository/13		0.17	2.00	2.17	2	3		2014
7	7	Network Secur repository/12		0.17	1.00	1.17	1	4		2012
8	4	Network Secur repository/11		0.20	2.00	2.20	2	3		2012
9	4	Comparative # repository/19		0.28	4.00	4.28	4	2		2014
10	4	Weighted Page repository/110		0.30	5.00	5.30	2	5		2005
11	4	A Crawler-bas repository/10		0.28	3.00	3.28	3	3		2014
12	4	Study of Vari repository/14		0.40	3.00	3.73	4	2		2002
13	8	Cybercrime: A repository/91		0.51	5.72	6.56	4	4		2000
14	4	An Empirical repository/16		0.46	5.72	6.18	4	1		2001
15	4	An Approach t repository/13		0.44	5.72	6.36	2	2		1994
16	4	An Extensive repository/19		0.21	2.00	2.21	6	0		1990
17	4	Study and Ana repository/12		1.44	13.07	14.31	1	2		1994
18	4	An Approach t repository/11		0.26	3.72	3.90	5	0		1994
19	4	Review on var repository/19		1.01	8.44	9.45	3	2		2000
20	4	Study on Vari repository/110		0.83	6.44	6.97	2	0		2003
21	4	Review based repository/11		0.20	2.00	2.20	1	2		2014
22	4	CRYPTOGRAPHY repository/11		0.17	1.00	1.17	0	2		2012
23	9	An Approach t repository/16		0.15	0.00	0.15	0	1		2012
24	8	Study of repr repository/13		0.15	0.00	0.15	2	0		2007

Fig 7.15 Fragment of Number of Downloads Saved in Log

id	query_keyword	category_id	category_name	cosine_similarity_value
178	crawl	2	Computer Science and Game Theory	0.0000
180	crawl	4	Cryptography and Security	0.0000
181	crawl	5	Distributed, Parallel, and Cluster Computing	0.0000
182	crawl	6	Information Retrieval	0.1804
183	crawl	7	Networking and Internet Architecture	0.0000
184	crawl	8	Software Engineering	0.0000
185	crawl	9	Soft Computing	0.0000
186	crawl	10	Formal Languages and Automata Theory	0.0000
187	crawl	11	Analysis & Design of Algorithms	0.0000
179	crawl	3	Computer Vision and Pattern Recognition	0.0000
177	crawl	1	Artificial Intelligence	0.0000
541	cryptography network security threat	2	Computer Science and Game Theory	0.0000
540	cryptography network security threat	1	Artificial Intelligence	0.2920
543	cryptography network security threat	4	Cryptography and Security	0.5679
530	cryptography network security threat	11	Analysis & Design of Algorithms	0.0000
549	cryptography network security threat	10	Formal Languages and Automata Theory	0.0000
548	cryptography network security threat	9	Soft Computing	0.0000
547	cryptography network security threat	8	Software Engineering	0.0000
546	cryptography network security threat	7	Networking and Internet Architecture	0.3325
542	cryptography network security threat	3	Computer Vision and Pattern Recognition	0.0000
544	cryptography network security threat	5	Distributed, Parallel, and Cluster Computing	0.0000
545	cryptography network security threat	6	Information Retrieval	0.0000
155	Cryptography technique	1	Artificial Intelligence	0.0000
158	Cryptography technique	4	Cryptography and Security	0.4856
159	Cryptography technique	5	Distributed, Parallel, and Cluster Computing	0.0000

Fig 7.16 Computation of Query Category by Query Analyzer

It can be observed from the screen shots that the proposed system crawls the WWW efficiently and researcher gets the desired results as per his query.

Fig. 7.17 shows the result screen after submitting a query “Survey of Web Page Ranking Algorithm” to the search interface.

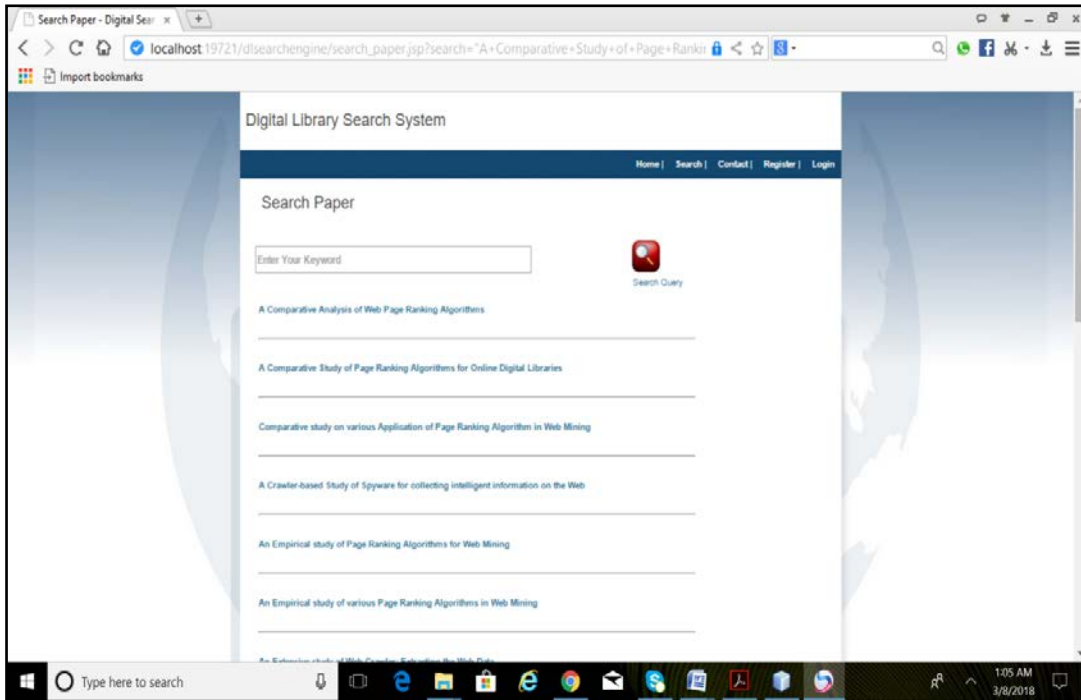


Fig. 7.17 Result Screen of proposed System for Query “Survey of Web Page Ranking Algorithm”

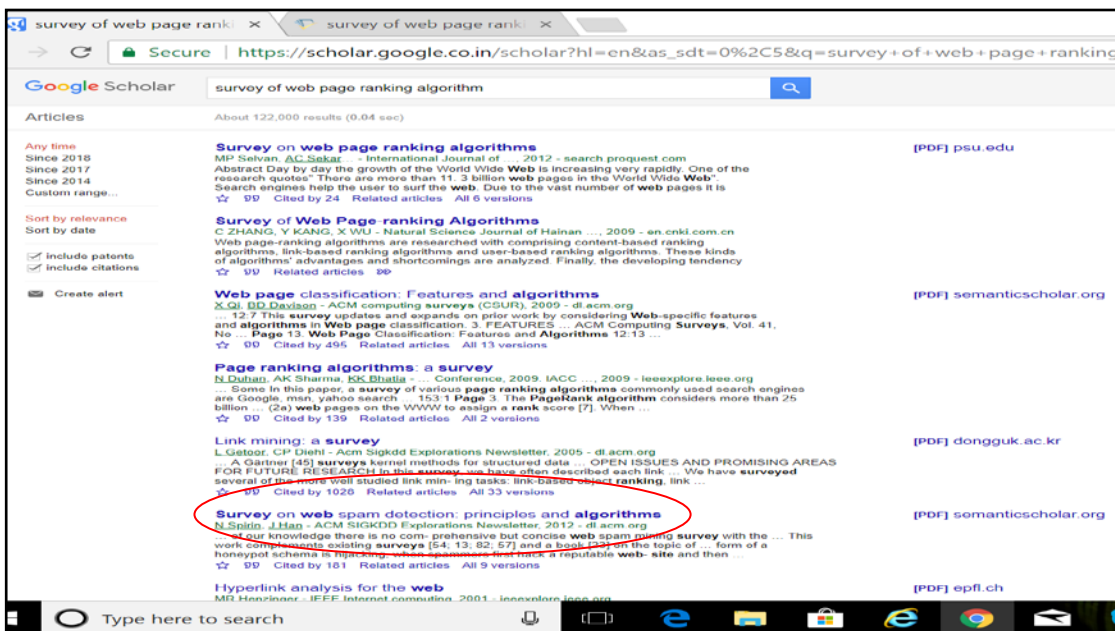


Fig. 7.18 Google Scholar’s Results for Query “Survey of Web Page Ranking Algorithm”

For comparing this scenario with Google Scholar and CiteSeer^X, the same query i.e. “Survey of Web Page Ranking Algorithm” was submitted on Google Scholar’s interface and CiteSeer^X. The snapshots of result list after submitting the query are shown in Fig. 7.18 and Fig 7.19.

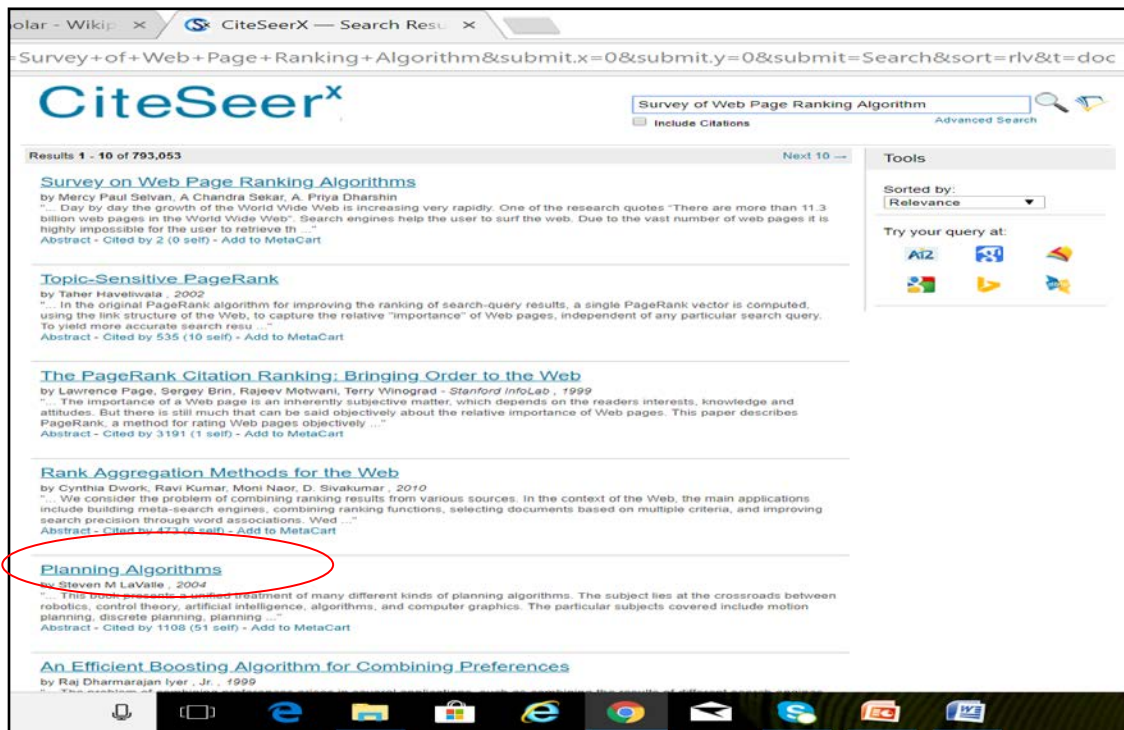


Fig. 7.19 CiteSeer^X’s Results for Query “Survey of Web Page Ranking Algorithm”

It can be seen that result list contains few irrelevant documents in Google Scholar and CiteSeer^X results list. To be noted, the encircled document “Web page Classification: Features and Algorithms”, which was appearing at order 3 in the Google Scholar results (as shown in Fig. 7.18) and encircled document “Planning Algorithm”, which was appearing at order 5 in CiteSeer^X results (as shown in Fig 7.19) are irrelevant as compared to proposed system results against user query. Therefore, search space has been reduced to large extent by using proposed DL system.

The result analysis of proposed DL system, Google Scholar and CiteSeer^X for query “Survey of Web Page Ranking Algorithm” is given in Fig. 7.20. The comparison is shown with respect to precision values.

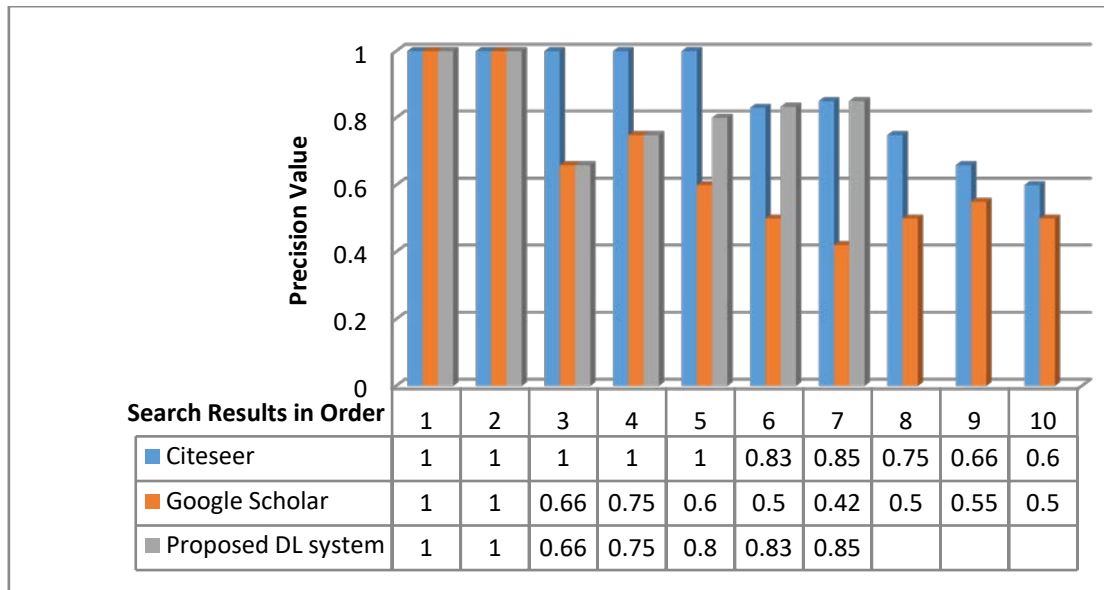


Fig 7.20 Comparison of Precision Values between CiteSeerx, Google Scholar and Proposed DL System

A group of 25 users from computer science domain were asked to search on proposed system and other keyword based search engines like Google Scholar, CiteSeer^x etc. The net performance of proposed system in terms of quality of search results and reduced navigation time is found to be higher than traditional digital library search system.

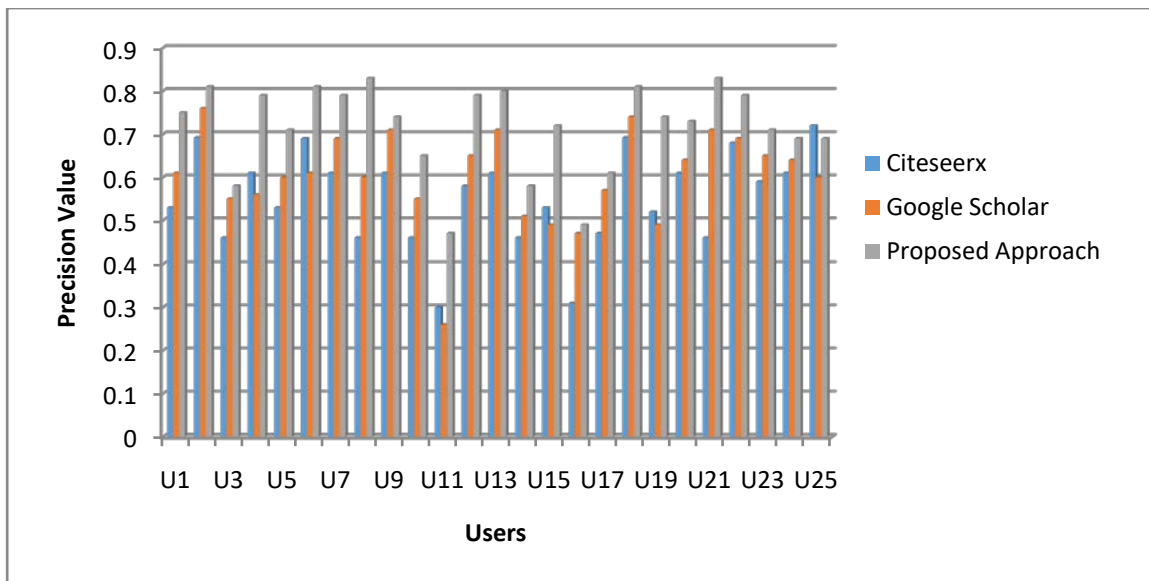


Fig. 7.21 Comparison of Precision Values between the Existing Approach and Proposed Approach as per User’s Perceptive

The system is also analyzed based on the performance measure precision by taking different queries from different categories. Two query sets were taken from different categories and each query set included 10 queries fired by different users on search interface as shown in Table 7.2.

Table 7.2 Query Sets given Different Users

SNo.	Query Set 1(Information Retrieval)	Query 2 (Networking)
1	Survey of Ranking Algorithms	Issues in Networking Protocols
2	Web Mining Algorithms	OSI Layer Architecture
3	Comparative Analysis of Page Ranking Algorithm	Data Link Layer Protocol
4	Web Crawler for Digital Documents	Difference between Wired and Wireless Communication
5	Information Retrieval Techniques	Recent Trends in Networking
6	Data Mining and their Applications	Network Topologies and Ethernet
7	Social Network Analysis in Information Retrieval	Difference between Switches, Routers and Modem
8	Recent Trends in Natural language Programming	Study of Media Access Protocol
9	Issues in Web Pre Fetching Techniques	Various Types of Topologies in Networking
10	Mining Techniques in Information Retrieval	Internet Protocol v4 and v6

The graph plotted between average precision, recall and F-measure values for *Query Set 1* and *Query Set 2* is shown in Fig 7.22.

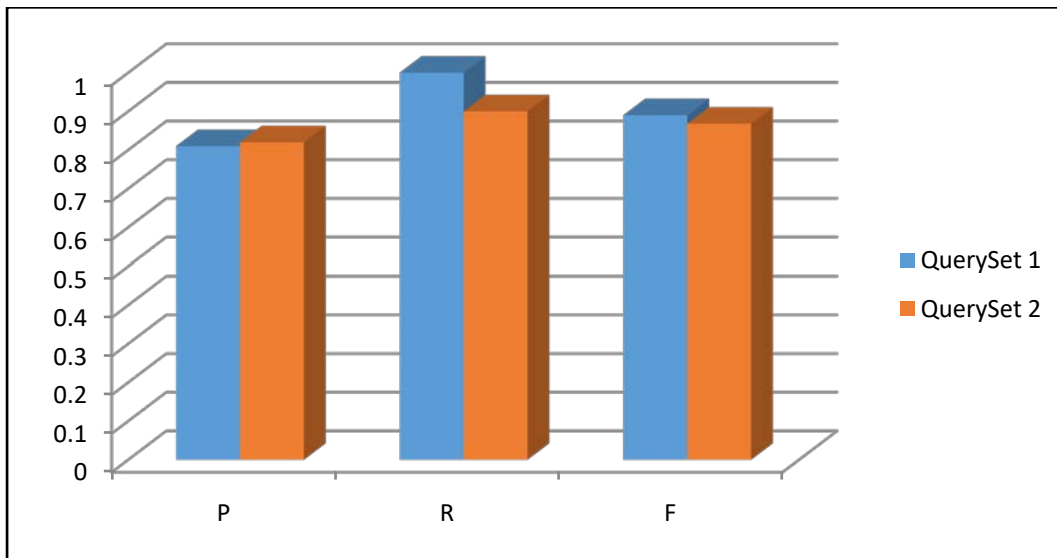


Fig 7.22 A graph showing Precision (P), Recall (R) and F-measure (F) values for Query Set 1 and Query Set 2

The next section describes the implementation details of proposed document categorization approach, which is based on multi level hierarchical structure to categorize the documents as per their topic.

7.5 DOCUMENT CATEGORIZATION

The proposed document categorization mechanism is implemented using Java JDK 6.0, NetBeans IDE , WordNet Version 3.0, PDFBox, MS-Access. Experiments are performed on Dual-Core Intel Pentium IV or higher Processor with 2.60GHz frequency and 4.00 GB RAM. The prebuilt topic taxonomy database is used as described in Appendix A. Categorization system was separately implemented to check its accuracy and has later embedded with the main search system.

Fig 7.23 displays the home page of the proposed categorization system for digital libraries. This page provides two options:-

1. Upload the new research paper
2. Search papers by submitting queries



Fig. 7.23 Home Screen of the Proposed Document Categorization System

In the upload section, a new research paper is uploaded in the database from the repository. The upload section is protected through password and can be accessed by authentic users or administrators. Depending upon the success or failure of the uploading

action, different outcomes are returned to the user. In the search section, upon submitting the user query, if the research papers related to the query exist in the database then their links are returned to the user otherwise an error page is displayed to the user.

Upload Section: After selecting the paper, system checks whether this selected paper exist in the database or not. If the paper does not exist in the database then it goes through various processing modules. First of all, information about the research paper i.e. authors, titles, references etc is extracted and the bookmarks of the paper are extracted. Then, the important keywords from the bookmarks are selected by the system and stored in the database.

After this, comparison is done between the keywords of paper and keywords of the categories. First the comparison is done with the main categories. After deciding main category, comparison is done to select the sub-category. Suppose the main category of the uploaded paper is *Networking*, then the comparison is done with the keywords of sub-categories as shown in Fig. 7.24.

categoryID	n/w_protoca	rout_protoca	type_n/w
1	SNMP	routing	wireless
1	ATM	token ring	wired
1	OSI	message	WSN
1	TCP/IP	adhoc	client
1	UDP	dynamic	network
1	layer	static	optical
1	ethernet	node	topology
1	protocol	mobility	server
1	signaling	link	internet
1	remote	OSPF	switch

Fig. 7.24 Keywords of the Networking Category

On the basis of similarity of these keywords, the category of the paper is decided and all this information gets stored in the database. This information is stored in the database as shown in Fig. 7.25.

paperID	title	author1	author2	author3	author4	category	path	categoryID
1	Network (In)Se	D. Brent Chapn				network protocol	F:\study\thesis\main doc\IP.pdf	1
2	Research and II	Rong Jin	Weiming Wang			network protocol	F:\study\thesis\main doc\research.pdf	1
3	A Genetic Algo	Darrell Whitley				neural network	F:\study\thesis\main doc\algo.pdf	2
4	Application of	Harsh Bhasin	Surbhi Bhatia			genetic algorithm	F:\study\thesis\main doc\app.pdf	2
5	Modified Gene	Harsh Bhasin	Neha Singla			genetic algorithm	F:\study\thesis\main doc\genetic.pdf	2
6	exact algorithm	Gerhard J. Woi				NP Problems	F:\study\thesis\main doc\exact.pdf	3
7	CHARACTER RE	Fakhraddin Ma	Jamal Fathi Abi			neural network	F:\study\thesis\main doc\neural.pdf	2
8	Security Proble	S.M. Bellovin				network protocol	F:\study\thesis\main doc\security.pdf	1

Fig. 7.25 Paper information in the Categorized Document Database

After this processing, the paper is successfully uploaded in the database and following outcome is displayed to the user by the system as shown in Fig. 7.26.

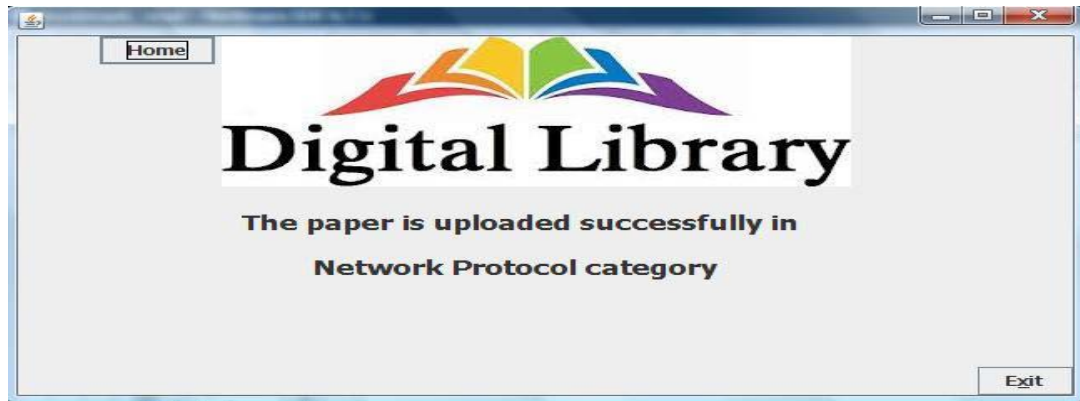


Fig. 7.26 Successful Uploading of the Paper

Along with the option of uploading, the proposed system provides the facility to search the database. If a user wants to search the documents, then he selects the option of searching by just clicking on the search option. The search interface shown in Fig. 7.27 is displayed to the user to search the database.

Search Interface: To search in the digital library, user has to submit a query either by entering the Title of the paper or any keyword based query as shown in Fig 7.27. After entering the query, the user proceeds by clicking on the submit button. System processes the submitted query. First of all, query is tokenized and the keywords of the paper are matched with the keywords of category on the basis of which category of the query is

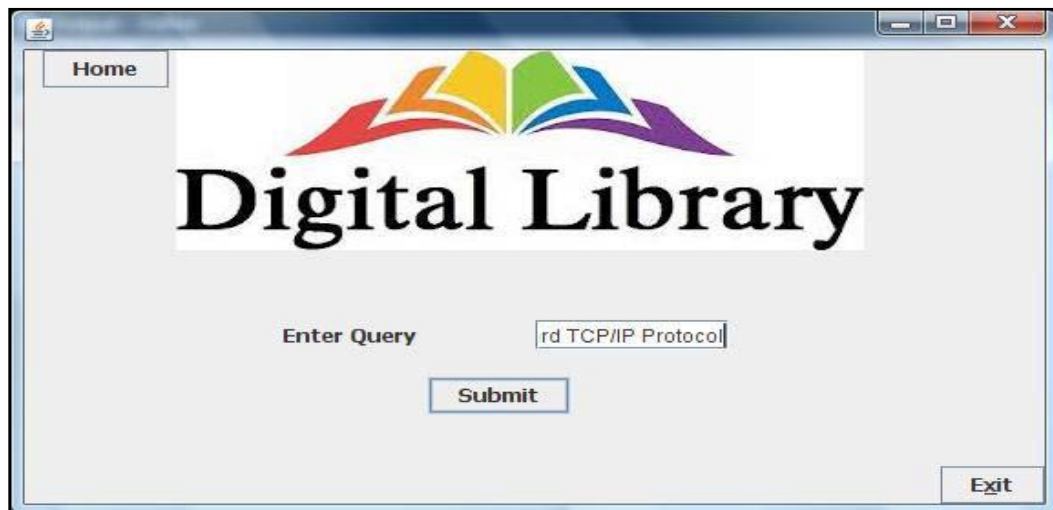


Fig. 7.27 Interface for searching the database

decided.

The category having highest cosine similarity value with the query terms is the most relevant category. Thus using cosine similarity measure, the leaf node category is decided in the domain tree. The links of the papers within the decided category are returned back to the user and following list of resultant papers is displayed as shown in Fig. 7.28.

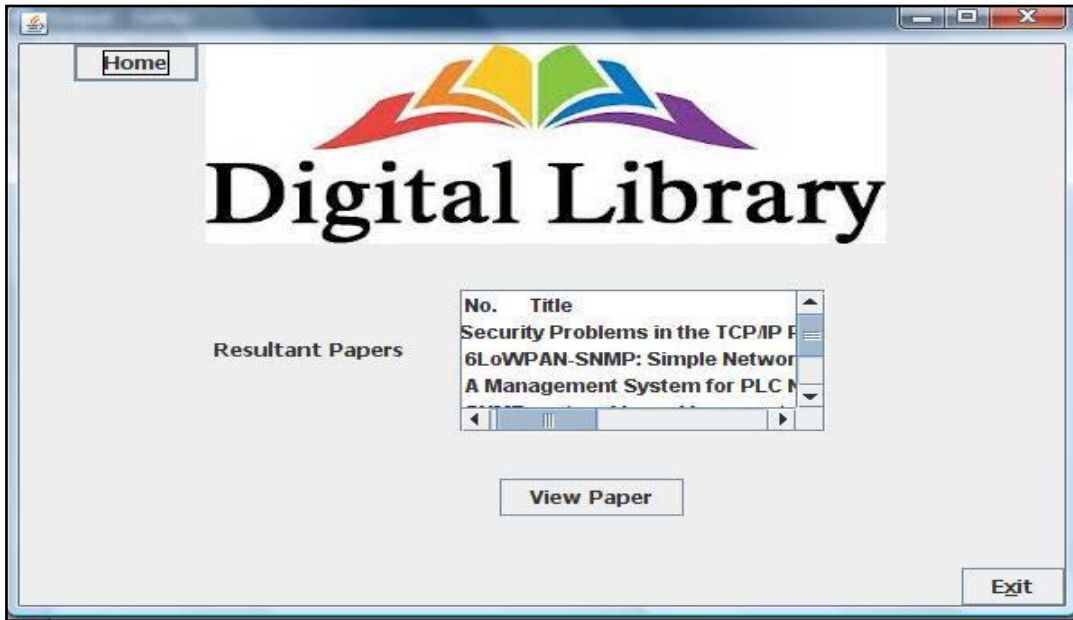


Fig. 7.28 Resultant List of Papers

On this output page, user has the option to view the paper of his choice. User can select the paper from the list according to his requirement and the click on View Paper button to view the selected paper.

The results of the proposed multi level document categorization approach are compared with the single level approach. Consider a sample fragment of two queries for which comparison is done. The comparison is done on the basis of precision value. The results obtained after submitting the query Q1: “SNMP is a standard TCP/IP Protocol” and Q2: “Working of artificial digital library” are shown in Table 7.2 along with those obtained from single level approach.

Analysis of these two approaches is done by plotting a graph between their precision, recall and F-measure as shown in Fig. 7.29.

Table 7.3 Resultant Papers for both the Approaches

Query	Results of Single Level Approach	Results of Multi Level Approach
1. SNMP is a standard TCP/IP Protocol Query	Topics in network and service management	Research and Implementation of SNMP in For CES Framework.
	A Management System for PLC Networks Using SNMP Protocol	6LoWPAN-SNMP: Simple Network Management Protocol for 6LoWPAN
	OS1 Reference Model-The ISO Model of Architecture for Open Systems Interconnection	A Management System for PLC Networks Using SNMP Protocol
	A Brief Tour of the Simple Network Management Protocol – CERT	OS1 Reference Model-The ISO Model of Architecture for Open Systems Interconnection
	Dynamic Routing Protocols II OSPF	SNMP protocol based home automation system
	SNMP protocol based home automation system	
	6LoWPAN-SNMP: Simple Network Management Protocol for 6LoWPAN	
	Towards Autonomic Network Management: an Analysis of Current and Future Research Directions	
	Research and Implementation of SNMP in For CES Framework	
	Types of Computer Networks and their Topologies	
2. Working of artificial digital library	Neural network approach to quantum-chemistry data: Accurate prediction of density functional theory energies	Neural network approach to quantum-chemistry data: Accurate prediction of density functional theory energies.
	Character Recognition Using Neural Networks	Artificial Neural Network Modelling for the Study of pH on the Fungal Treatment of Red mud
	Face Recognition using Principle Component Analysis, Eigenface and Neural Network	Artificial Neural Network to Predict Skeletal Metastasis in Patients with Prostate Cancer
	Optimization and Evaluation of a Neural-Network Classifier for PET Scans of Memory-Disorder Subjects.	A Hierarchical Self-organizing Associative Memory for Machine Learning
	Artificial Neural Network Modelling for the Study of pH on the Fungal Treatment of Red mud	Face Recognition using Principle Component Analysis, Eigenface and Neural Network
	A Hierarchical Self-organizing Associative Memory for Machine Learning	
	Artificial Neural Network to Predict Skeletal Metastasis in Patients with Prostate Cancer	
	Neural Network with Memory and Cognitive Functions	
	Optimized Approximation Algorithm in Neural Networks Without Over fitting.	

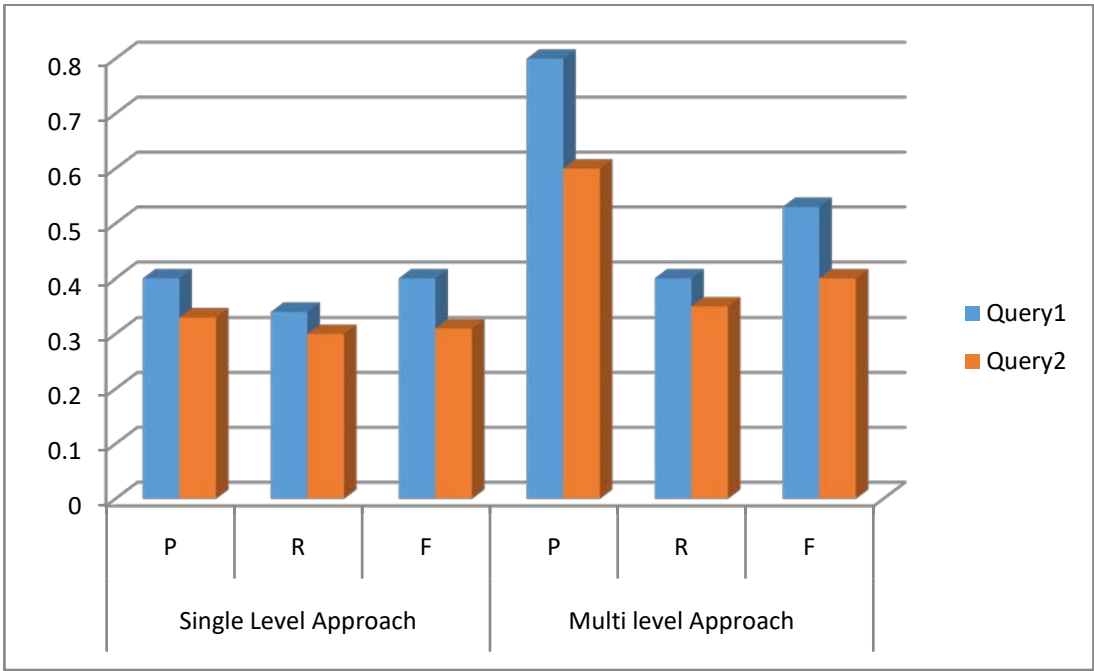


Fig 7.29 Comparison between Single level and Multi level Approach

The next section describes the implementation details of proposed ranking algorithm BCC, which is based on Web Content and Structure Mining.

7.6 BOOKMARK BASED CITATION COUNT

The Bookmark based Citation Count (BCC) method has been proposed for the Clustering and Ranking technique described in Section 6.3. It has been implemented and its comparison is carried out with Citation Count (CC), Time Dependant Citation Count (TDCC) and PageRank (PR). Result analysis depicts that BCC produces relevant i.e. best possibly matched documents in the top of the results. The BCC has been implemented in .NET technology. For the present experimentation, citation graph shown in Appendix B is considered by BCC.

The result analysis of ranking algorithm BCC has been carried out by comparing the results with CC, PR and TDCC approach using graphical analysis. Fig 7.30 shows the CC, TDCC, PR and BCC values for documents of the citation graph given in Appendix B. Analysis of these approaches is done by plotting a graph between their rank score as shown in Fig. 7.31.

id	title	CC	page_rank	BCC	TDCC	incoming link	outgoing link	year of publication	abstract_data
1	A Comparative Study of Pa	0.0000	0.15	0.00	0.00	0	6	2014	With the increasing
2	The Web Crawler: An Archite	0.0000	0.15	0.00	0.00	0	6	2016	A web crawler is a p
3	Study on Network Security: 0	0.0000	0.15	0.00	0.00	0	7	2015	Network security has
4	An Empirical study of vari	2.0000	1.09	1.32	1.62	2	5	2011	As the web is growin
5	Study on Web Pages Link St	1.0000	0.30	0.65	0.86	1	5	2012	Web mining is the no
6	Survey on working of sear	2.0000	0.89	1.10	1.23	2	3	2010	The main purpose of
7	Network Security: it's a d	1.0000	0.53	0.60	0.75	1	4	2013	Network security has
8	Network Security Threats a	2.0000	0.99	1.23	0.89	2	3	2012	In a brave new age o
9	Comparative study on vari	4.0000	2.21	3.60	2.31	4	2	2019	The World Wide Web i
10	Weighted Page Rank Algorit	2.0000	1.09	1.46	0.70	2	5	2005	The World Wide Web o
11	A Crawler-based Study of	3.0000	0.93	1.45	2.12	3	3	2014	Malicious spyware po
12	Study of various Cryptogr	4.0000	1.67	2.45	3.70	4	2	2002	Cryptography is the
13	Cybercrime: A system three	4.0000	2.90	3.50	0.82	4	4	2000	This research paper
14	An Empirical study of Page	4.0000	3.10	3.20	2.56	4	1	2001	Web is expending day
15	An Approach to scale Web	2.0000	1.20	1.89	1.21	2	2	1998	This paper describes
16	An Extensive study of Web	6.0000	2.32	3.10	4.90	6	0	1990	Internet usage has i
17	Study and Analysis of Web	1.0000	0.49	0.81	0.21	1	2	1996	The World Wide Web
18	An Approach to design a P	5.0000	2.01	2.21	3.21	5	0	1998	Broad web search eng
19	Review on various types	3.0000	1.21	2.10	1.20	3	2	2000	Cryptography is emer
20	Study on various Issues an	2.0000	1.33	1.75	1.10	2	0	2003	Cyber Security plays
21	Review based on Applicatio	1.0000	0.30	0.63	0.31	1	2	2010	This Genetic Algorith
22	CRYPTOGRAPHY AND GENETIC	0.0000	0.15	0.00	0.00	0	2	2013	The genetic algorith
23	An Approach to solve PCP	0.0000	0.15	0.00	0.00	0	1	2012	Post Correspondence

Fig 7.30 Variation of CC, PR, TDCC and BCC Values

It is observed that the proposed Bookmark based Citation Count gives more precise results than existing approaches for the example scenario. BCC method considers the content similarity of citations with the cited publication for ranking the publication whereas PR method computes the ranking by considering only number of backlinks; CC considers the incoming links and TDCC method consider the incoming links with the age of the publication.

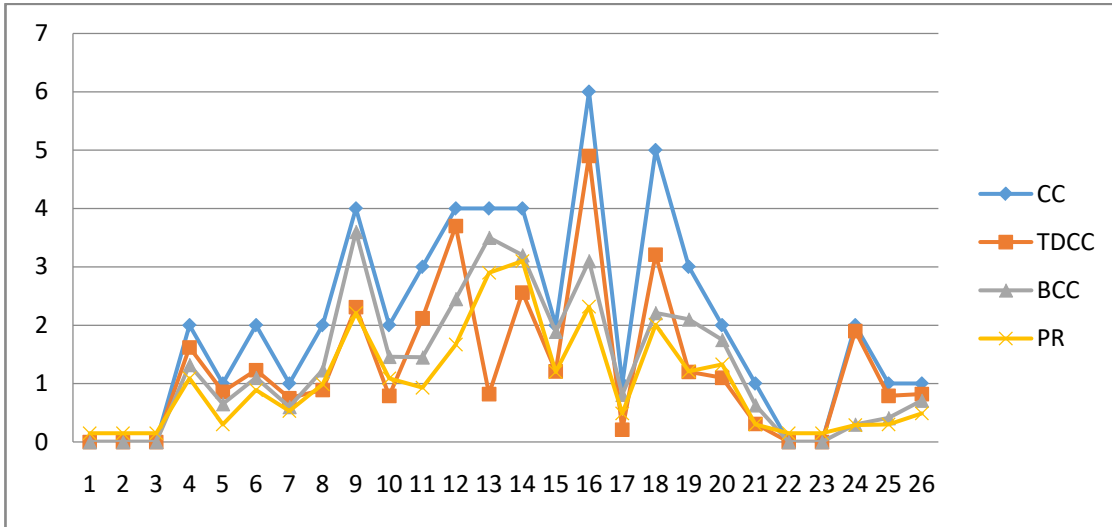


Fig 7.31 Comparison of CC, TDCC, PR and BCC Values

The next chapter concludes the work accomplished in this thesis. The future research directions are also enumerated in this regard.

Chapter VIII

CONCLUSION AND FUTURE SCOPE

8.1 CONCLUSION

In this thesis, shortcomings of existing systems have been resolved in order to get efficient and quality documents as per a researcher query. A unified search system for digital libraries which improves the relevancy of search results and reduces the search space is designed which effectively achieves the following objectives:

- ***Unified System to Crawl Maximum Documents:*** A unified framework for online digital library search engine is proposed in order to cover the existing web as maximum as possible. The experimental results have shown a highly increase in coverage by incorporating proposed framework for online digital library search engines.
- ***Approach to Categorize Documents:*** In the proposed categorization system, the approach of grouping together the related research papers within a category is developed. By this way, it is often easier to scan a few coherent groups than many individual papers thereby reducing the search space.
- ***Efficient Document Organization Scheme:*** With the help of indexing module, documents are organized in a multi-level hierarchal structure. It maintains the primary index for keyword based searching and secondary index consists of multi level index structure, which provides the retrieval of documents based on category and clustering. This scheme works in such a way that, it provides fast and more efficient retrieval of relevant documents as per user's query.
- ***More Relevant Results:*** In this work, documents are analyzed in order to achieve more relevant results based on content, link and User browsing behavior. The experiment results have shown that with the help of considering clustering and ranking based approach, the results corresponding to users' query provide more relevant results as per user's interest.

Summarizing, a design of a novel digital library search engine for efficiently crawling, index, order and represent the documents has been proposed that not only addresses the

problems prevailing in the existing digital library systems but also uses the missing document finder module and document categorization as the major source of the topical information retrieval system.

After the analyzing the experimental results, following observations regarding the performance of proposed system have been drawn:

- **High Precision:** The results were analyzed using the performance metrics: *Precision, Recall and F-measure*. High values of performance metrics for various tests conducted on the system indicate that it accurately retrieves the documents that the user desires.
- **Extensibility:** The classification of the proposed work is done in such a way that a modular architecture is developed with the expectation that new functionalities can easily be added by third parties according to their requirements.
- **Robustness:** The system is robust in the sense that it is able to find the missing documents by employing missing document finder module. So, the system, after getting the URLs with dangling information during the crawl process is unlikely to break or fall.

8.2 FUTURE SCOPE

Some of the possible extensions and issues that could be further explored in the near future are as follows:

- **Automatic Creation of Taxonomy:** In the proposed system, categories of considered taxonomy are pre-defined. There is a scope of dynamic creation of categories and sub-categories under them.
- **Crawler Freshness:** The crawler downloads the documents which get stored in database. After downloading, there is need to revisit them again to get up-to-date copy or missing information of documents. So, an automated mechanism may be developed that will revisit documents for updated documents.
- **Compatibility with Semantic web:** The Proposed framework may be made compatible with the semantic web in order to get more refined and relevant results.

REFERENCES

- [1] D. Kelly, "Methods for Evaluating Interactive Information Retrieval Systems with Users," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 1–2, pp. 1-224, 2009.
- [2] X. Dong and T. Su. Louise, "Search Engines on the World Wide Web and Information Retrieval from the Internet: A Review and Evaluation," *Online and CD-Rom Review*, vol. 21, no. 2, pp. 67-82, 1997.
- [3] R. B. Yates and B. R. Neto, *Modern information retrieval*, vol. 463, New York: ACM press, v1999.
- [4] C. Manning, P. Raghavan and S. Hinrich, "Introduction to Information Retrieval," *Natural Language Engineering*, vol. 16, no. 1, pp. 100-103, 2010.
- [5] Google Search Engine. <http://www.google.com>
- [6] B. R. Schatz, "Information Retrieval in Digital Libraries: Bringing Search to the Net," *Science*, vol. 275, no. 5298, pp. 327-334, 1997.
- [7] M. Krishnamurthy, "Open Access, Open Source and Digital Libraries: A current Trend in University Libraries around the World," *Program: electronic library and information systems*, vol. 42, no. 1, pp. 48-55, 2008.
- [8] R. Pandey, "Digital Library Architecture," *DRTC Workshop on Digital Libraries: Theory and Practice*, Bangalore, pp. 1-16, 2003.
- [9] H. Suleman, E. A. Fox, and D. P. Madalli, "Design and Implementation of Networked Digital Libraries: Best Practices," *DRTC Workshop on Digital Libraries: Theory and Practice*, pp. 79-87, 2003.
- [10] S. Brin and P. Lawrence, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Computer networks and ISDN systems*, vol. 30, no. 1-7, pp. 107-117, 1998.
- [11] A. Spink, D. Wolfram, M. B. J. Jansen and T. Saracevic, "Searching the Web: The Public and their Queries," *Journal of the American society for information science and technology*, vol. 52, no. 3, pp. 226-234, 2001.
- [12] B. E. Brewington and G. Cybenko, "How Dynamic is the Web? 1," *Computer Networks*, vol. 33, no. 1-6, pp. 257-276, 2000.
- [13] N. Duhan, A. K. Sharma, and K. K. Bhatia, "Page Ranking Algorithms: A Survey," In *Proc. IEEE International Advance Computing Conference*, 2009, pp. 1530-1537.

- [14] W. B. Croft, D. Metzler and T. Strohman, *Search engines: Information retrieval in practice*, vol. 520, Reading: Addison-Wesley, 2010.
- [15] K. D. Bollacker, L. Steve and C. L. Giles, "CiteSeer: An Autonomous Web Agent for Automatic Retrieval and Identification of Interesting Publications," In Proc. second international conference on Autonomous agents, ACM, 1998, pp. 116-123.
- [16] S. Das and M. Krishnamurthy, "Architectural Components of Digital Library: A Practical Example Using DSpace," In Proc. Trends in Management of Academic Libraries in Digital Environment (TMALDEN), Jain University, Bangalore, 2014, pp. 183-194.
- [17] D. Gourley, "An Architecture for the Evolving Digital Library," *EDUCAUSE Information Resources Library*, vol. 26, 2001.
- [18] R. Kahn and R. Wilensky, "A Framework for Distributed Digital Object Services," *International Journal on Digital Libraries*, vol. 6, no. 2, pp. 115-123, 2006.
- [19] R. E. Kahn and D. K. Ely, "System for Uniquely and Persistently Identifying, Managing, and Tracking Digital Objects," U.S. Patent 6,135,646, 24 October, 2000.
- [20] G. Marchionini, "Exploratory Search: from Finding to Understanding," *Communications of the ACM*, vol. 49, no. 4, pp. 41-46, 2006.
- [21] W. Hong, J. Y. L Thong, W. M. Wong and K.Y. Tam, "Determinants of User Acceptance of Digital Libraries: An Empirical Examination of Individual Differences and System Characteristics," *Journal of Management Information Systems*, vol. 18, no. 3, pp. 97-124, 2002.
- [22] B. J. Jansen and A. Spink, "How are We Searching the World Wide Web? A Comparison of Nine Search Engine Transaction Logs," *Information processing & management*, vol. 42, no. 1, pp. 248-263, 2006.
- [23] M. Kobayashi and K. Takeda, "Information Retrieval on the Web," *ACM Computing Surveys (CSUR)*, vol. 32, no. 2, pp. 144-173, 2000.
- [24] L. Braden-Harder, S. H. Corston, W. B. Dolan and L. H. Vanderwende, "Apparatus and Methods for An Information Retrieval System that Employs Natural Language Processing of Search Results to Improve Overall Precision," U.S. Patent 5,933,822, 3 Aug., 1999.

- [25] J. Bar-Ilan and M. Levene, "A Method to Assess Search Engine Results," *Online Information Review*, vol. 35, no. 6, pp. 854-868, 2011.
- [26] AOL Search Engine. <https://search.aol.com/>
- [27] Web Crawler Search Engine. <https://www.webcrawler.com/>
- [28] Yahoo Search Engine. <https://in.search.yahoo.com/>
- [29] Good Search Engine. <https://www.goodsearch.org/>
- [30] DuckDuckGo Search Engine. <https://duckduckgo.com/>
- [31] Bing search Engine. <https://www.bing.com/>
- [32] Yacy Search Engine. <https://yacy.net/>
- [33] Cliqz Search Engine. <https://cliqz.com/en/whycliqz/search-engine>
- [34] C. Castillo, "Effective Web Crawling," *ACM sigir forum*, vol. 39, no. 1, pp. 55-56, 2005.
- [35] J. A. Curtis and G. F. Scherer, "Search Engine using Indexing Method for Storing and Retrieving Data," U.S. Patent 6,278,992, 21 Aug., 2001.
- [36] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," Tech. Report, Stanford InfoLab, 1999.
- [37] R. Blumberg and S. Atre, "The Problem with Unstructured Data," *Dm Review*, vol. 13, no. 42-49, pp. 62, 2003.
- [38] O. Hoerber and T. Khazaei, "Evaluating Citation Visualization and Exploration Methods for Supporting Academic Search Tasks," *Online Information Review*, vol. 39, no. 2, pp. 229-254, 2015.
- [39] CiteSeer Digital Library. citeseerx.ist.psu.edu/
- [40] Google Scholar Search Engine. <https://scholar.google.co.in/>
- [41] IEEE Digital Library. <https://ieeexplore.ieee.org/Xplore/home.jsp>
- [42] Science Direct Digital Library. <https://www.sciencedirect.com/>
- [43] Open Access Journals Search Engine. www.oajse.com/
- [44] Academic Publications eJournal. apejournal.weebly.com/
- [45] Academic Search Digital Library. <https://www.academic-search.com/>
- [46] Springer Digital Library Search Engine. <https://link.springer.com/>
- [47] J. Sun and B. Z. Yuan, "Development and Characteristic of Digital Library as a Library Branch," *IERI Procedia*, vol. 2, pp.12-17, 2012.

- [48] B. Nahak and P. S. Patra, "Planning, Designing and Developing of Digital Libraries and Digital Preservation," INFLIBNET Centre, Gandhinagar, 2014.
- [49] A. Shiri, "Digital Library Research: Current Developments and Trends," *Library Review*, vol. 52, no. 5, pp. 198-202, 2003.
- [50] D. Bhatt, D. A. Vyas and S. Pandya, "Focused Web Crawler," *Advances in Computer Science and Information Technology*, vol. 2, no. 11, pp. 1-6, 2015.
- [51] B. Ganguly and R. Sheikh, "A Review of Focused Web Crawling Strategies," *International Journal of Advanced Computer Research*, vol. 2, no. 4, pp. 261, 2012.
- [52] D. Bergmark, C. Lagoze, and A. Sbityakov, "Focused Crawls, Tunneling, and Digital Libraries," In Proc. International Conference on Theory and Practice of Digital Libraries, Springer, Berlin, Heidelberg, 2002, pp. 91-106.
- [53] S. Chakrabarti, M. V. D. Berg, and B. Dom, "Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery," *Computer networks*, vol. 31, no. 11-16, pp. 1623-1640, 1999.
- [54] W. Wang, X. Chen, Y. Zou, H. Wang and Z. Dai, "A Focused Crawler based on Naive Bayes Classifier," In 2010 Third International Symposium on Intelligent Information Technology and Security Informatics (IITSI), 2010, pp. 517-521.
- [55] H. L. Yu, L. Bingwu and Y. Fang, "Similarity Computation of Web Pages of Focused Crawler," In Proc. IEEE 2010 International Forum on Information Technology and Applications (IFITA), pp. 70-72.
- [56] P. Boldi, B. Codenotti, M. Santini and S. Vigna, "Ubicrawler: A Scalable Fully Distributed Web Crawler," *Software: Practice and Experience*, vol. 34, no. 8, pp. 711-726, 2004.
- [57] J. Cho and H. G. Molina, "Parallel crawlers," In Proc. ACM 11th International Conference on World Wide Web, 2002, pp. 124-135.
- [58] S. Sharma, A. K. Sharma and J. P. Gupta, "A Novel Architecture of a Parallel Web Crawler," *International Journal of Computer Applications*, vol. 14, no. 4, pp.38-42, 2011.
- [59] T. V. Udupure, R. D. Kale and R. C. Dharmik, "Study of Web Crawler and its Different Types," *IOSR Journal of Computer Engineering*, vol. 16, no. 1, pp. 1-5, 2014.

- [60] N. Singhal, A. Dixit and A. K. Sharma, "Design of a Priority based Frequency Regulated Incremental Crawler," *International Journal of Computer Application*, vol. 1, no. 1, pp. 47-52, 2010.
- [61] J. Cho and H. G. Molina, "The Evolution of the Web and Implications for An Incremental Crawler," Tech. Report, Stanford, 1999.
- [62] A. K. Sharma and A. Dixit, "Self Adjusting Refresh Time based Architecture for Incremental Web Crawler," *International Journal of Computer Science and Network Security*, vol. 8, no. 12, pp. 349-354, 2008.
- [63] B. W. On and D. Lee, "PaSE: Locating Online Copy of Scientific Documents Effectively," In Proc. Springer International Conference on Asian Digital Libraries, Berlin, Heidelberg, 2004, pp. 408-418.
- [64] G. Hoff and M. Mundhenk, "Finding Scientific Papers with HomepageSearch and MOPS," In Proc. ACM 19th annual international conference on Computer documentation, 2001, pp. 201-207.
- [65] D. Carmel, E. Y. Tov and H. Roitman, "Enhancing Digital Libraries using Missing Content Analysis," In Proc. ACM 8th ACM/IEEE-CS joint conference on Digital libraries, 2008, pp. 1-10.
- [66] J. Qin, Y. Zhou and M. Chau, "Building Domain-Specific Web Collections for Scientific Digital Libraries: A Meta-Search Enhanced Focused Crawling Method," In Proc. IEEE 2004 Joint ACM/IEEE Conference on, 2004, pp. 135-141.
- [67] J. Wu, P. Teregowda, J. P. F. Ramírez, P. Mitra, S. Zheng, and C. L. Giles, "The Evolution of a Crawling Strategy for an Academic Document Search Engine: Whitelists and Blacklists," In Proc. ACM the 4th Annual ACM Web Science Conference, 2012, pp. 340-343.
- [68] K. Premlatha and T. Geetha, "Focused Crawling for Educational Materials from the Web," *International Journal of Computer Science & Informatics*, vol. 1, no. 2, pp. 26-29, 2011.
- [69] S. D. Gollapalli, C. L. Giles, P. Mitra, and C. Caragea, "On Identifying Academic Homepages for Digital Libraries," In Proc. ACM the 11th annual international ACM/IEEE joint conference on Digital libraries, 2011, pp. 123-132.

- [70] S. D. Gollapalli, K. Patel, and C. Caragea, "A Search/Crawl Framework for Automatically Acquiring Scientific Documents," *arXiv preprint arXiv:1604.05005*, 2016.
- [71] S. Das, P. Mitra and C. Giles, "Learning to Rank Homepages for Researcher Name Queries," The International Workshop on Entity-Oriented Search, Citeseer, pp. 53-58. 2011.
- [72] Z. Zhuang, R. Wagle and C. L. Giles, "What's there and What's not?: Focused Crawling for Missing Documents in digital libraries," In Proc. ACM the 5th ACM/IEEE-CS joint conference on Digital libraries, 2005, pp. 301-310.
- [73] J. E. Conover and D. M. C. Anthony, "System and Method for Cataloguing Digital Information for Searching and Retrieval," U.S. Patent 6,701,314, 2 March, 2004.
- [74] S. Ceri, A. Bozzon, M. Brambilla, E. D. Valle, P. Fraternali, and S. Quarteroni, "Meta-Search and Multi-Domain Search," *Web information retrieval, Data-centric systems and applications*, Springer, pp 161-179, 2013.
- [75] P. Sojka and M. Liska, "Indexing and Searching Mathematics in Digital Libraries," In Proc. Springer International Conference on Intelligent Computer Mathematics, Berlin, Heidelberg, 2011, pp. 228-243.
- [76] RongJin, "Text Processing", <http://slideplayer.com/slide/7097844/>
- [77] J. Zobel, A. Moffat and K. Ramamohanarao, "Inverted files versus Signature Files for Text Indexing," *ACM Transactions on Database Systems (TODS)* , vol. 23, no. 4, pp. 453-490, 1998.
- [78] Y. Chen, "Signature Files and Signature Trees," *Information Processing Letters*, vol. 82, no. 4, pp.213-221, 2002.
- [79] J. T. Y. Ching and K. R. Chennupati, "Collection Evaluation through Citation Analysis Techniques: a Case Study of the Ministry of Education, Singapore," *Library Review*, vol. 51, no. 8, pp. 398-405, 2002.
- [80] A. W. K. Harzing and R. V. Wal, "Google Scholar as a New Source for Citation Analysis," *Ethics in science and environmental politics*, vol. 8, no. 1, pp. 61-73, 2008.
- [81] S. Lawrence, C. L. Giles and K. Bollacker, "Digital Libraries and Autonomous Citation Indexing," *Computer*, vol. 32, no. 6, pp. 67-71, 1999.

- [82] T. D. Nguyen and M. Y. Kan, "Keyphrase Extraction in Scientific Publications," In Proc. Springer International conference on Asian digital libraries, Berlin, Heidelberg, 2007, pp. 317-326.
- [83] C. Gutwin, G. Paynter, I. Witten, C. N. Manning and E. Frank, "Improving Browsing in Digital Libraries with Keyphrase Indexes," *Decision Support Systems*, vol. 27, no. 1-2, pp. 81-104, 1999.
- [84] P. Kanerva, Pentti, J. Kristoferson and A. Holst, "Random Indexing of Text Samples for Latent Semantic Analysis," *In Proceedings. of the Annual Meeting of the Cognitive Science Society*, vol. 22, no. 22, 2000.
- [85] E. Garcia, "Singular Value Decomposition (svd) a Fast Track Tutorial," *Using the Singular Value Decomposition*, 2006.
- [86] ALTAVISTA Digital Library Search Engine. <https://digital.com/about/altavista/>
- [87] Academic Search Engine. <https://academic.microsoft.com/>
- [88] INFOSEEK Digital Library Search Engine. <https://web.archive.org/web/20090520050825/http://www.clubi.ie:80/webserch/engines/infoseek/index.htm>
- [89] S. Malik, "A Comparative Study of two major Search Engines: Google and Yahoo," *Oriental Journal of Computer Science & Technology*, vol. 1, no. 7, pp. 29-37, 2014.
- [90] EXCITE Digital Library Search Engine. www.excite.com/
- [91] D. Gupta, K. K. Bhatia and A. K. Sharma, "A Novel Indexing Technique for Web Documents using Hierarchical Clustering," *International Journal of Computer Science and Network Security*, vol. 9, no. 9, pp. 168, 2009.
- [92] S. H. Cha, "Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions," *City*, vol. 1, no. 2, pp. 1, 2007.
- [93] P. Gupta and A. K. Sharma, "Context based Indexing in Search Engines using Ontology," *International Journal of Computer Applications*, vol. 1, no. 14, pp. 49-52, 2010.
- [94] P. Mudgil, A. K. Sharma, and P. Gupta, "An Improved Indexing Mechanism to Index Web Documents," In Proc. IEEE 2013 5th International Conference on Computational Intelligence and Communication Networks (CICN), 2013, pp. 460-464.

- [95] S. Nešić, F. Crestani, M. Jazayeri and D. Gašević, "Concept-based Semantic Annotation, Indexing and Retrieval of Office-like Document Units," *Adaptivity, Personalization and Fusion of Heterogeneous Information*, LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, pp. 134-135, 2010.
- [96] M. A. Angrosh, S. Cranefield, and N. Stanger, "Contextual Information Retrieval in Research Articles: Semantic Publishing Tools for the Research Community," *Semantic Web*, vol. 5, no. 4, pp. 261-293, 2014.
- [97] S. Bashir and A. Rauber, "On the Relationship between Query Characteristics and IR Functions Retrieval Bias," *Journal of the American Society for Information Science and Technology*, vol. 62, no. 8, pp. 1515-1532, 2011.
- [98] M. Coates, "Search Engine Queries used to Locate Electronic Thesis and Dissertations: Differences for local and non-local users," *Library Hi Tech*, vol. 32, no. 4, pp. 667-686, 2014.
- [99] H. Khatri, "Query Processing over Incomplete Autonomous Web Databases," PhD diss., Arizona State University, 2006.
- [100] N. Alemayehu and P. Willett, "The Effectiveness of Stemming for Information Retrieval in Amharic," *Program*, vol. 37, no. 4, pp. 254-259, 2003.
- [101] H. Cui, J. R. Wen, J. Y. Nie and W. Y. Ma, "Probabilistic Query Expansion using Query Logs," In Proc. ACM 11th international conference on World Wide Web, 2002, pp. 325-332.
- [102] Z. Zhang and O. Nasraoui, "Mining Search Engine Query Logs for Query Recommendation," In Proc. ACM 15th international conference on World Wide Web, 2006, pp. 1039-1040.
- [103] J. Beel and B. Gipp, "Google Scholar's Ranking Algorithm: the Impact of Citation Counts (an Empirical Study)," In Proc. IEEE Third International Conference on Research Challenges in Information Science, 2009, pp. 439-446.
- [104] L. Marian, "Ranking Scientific Publications based on their Citation Graph," Tech Report, .No. CERN-THESIS-2009-029. 2009.

- [105] L. Marian, J. Y. Le Meur, M. Rajman and M. Vesely, "Citation Graph based Ranking in Invenio," In Proc. Springer International Conference on Theory and Practice of Digital Libraries, Berlin, Heidelberg, 2010, pp. 236-247.
- [106] A. Sidiropoulos and Y. Manolopoulos, "Generalized Comparison of Graph-based Ranking Algorithms for Publications and Authors," *Journal of Systems and Software*, vol. 79, no. 12, pp. 1679-1700, 2006.
- [107] Y. Sun and C. L. Giles, "Popularity Weighted Ranking for Academic Digital Libraries," In Proc. Springer European Conference on Information Retrieval, Berlin, Heidelberg, 2007, pp. 605-612.
- [108] J. M. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," *Journal of the ACM (JACM)*, vol. 46, no. 5, pp. 604-632, 1999.
- [109] Math Explorer's Club, "The Mathematics of Web Search", <http://www.math.cornell.edu/~mec/Winter2009/RalucaRemus/Lecture4/lecture4.html>
- [110] S. Singla, N. Duhan and U. Kalkal, "A Novel Approach for Document Ranking in Digital Libraries using Extractive Summarization," *International Journal of Computer Applications*, vol. 74, no. 18, pp. 25-31, 2013.
- [111] L. Li, Y. Shang and W. Zhang, "Improvement of HITS-based Algorithms on Web Document," In Proc. ACM 11th international conference on World Wide Web, 2002, pp. 527-535.
- [112] Z. Guangqian and L. Xin, "Study on the Method of Ranking Scientific Papers," In Proc. IEEE 2010 International Conference on E-Business and E-Government (ICEE), 2010, pp. 3062-3066.
- [113] Y. Ding, E. Yan, A. Frazho and J. Caverlee, "PageRank for Ranking Authors in Co-citation Networks," *Journal of the American Society for Information Science and Technology*, vol. 60, no. 11, pp. 2229-2243, 2009.
- [114] P. Thwe, "Proposed Approach for Web Page Access Prediction using Popularity and Similarity based Page Rank Algorithm," *International Journal of Science and Technology Research*, vol. 2, no. 3, pp. 240-246, 2013.
- [115] M. Deshpande and G. Karypis, "Selective Markov Models for Predicting Web Page Accesses," *ACM Transactions on Internet Technology (TOIT)*, vol. 4, no. 2, pp. 163-184, 2004.

- [116] S. Qiao, T. Li, H. Li, Y. Zhu, J. Peng and J. Qiu, "SimRank: A Page Rank Approach based on Similarity Measure," In Proc. IEEE 2010 International Conference on Intelligent Systems and Knowledge Engineering (ISKE), 2010, pp. 390-395.
- [117] V. T. Nguyen, "Using Social Annotation and Web Log to Enhance Search Engine," *International Journal of Computer Science Issues, IJCSI*, vol. 6, no. 2, pp. 1-6, 2009.
- [118] M. P. S. Bhatia and D. Khurana, "Experimental Study of Data Clustering using k-Means and Modified Algorithms," *International Journal of Data Mining & Knowledge Management Process*, vol. 3, no. 3, pp. 17-30, 2013.
- [119] K. Wen, R. Li, J. Xia, and X. Gu, "Optimizing Ranking Method using Social Annotations based on Language Model," *Artificial Intelligence Review*, vol. 41, no. 1, pp. 81-96, 2014.
- [120] J. R. Bellegarda, "Statistical Language Model Adaptation: Review and Perspectives," *Speech communication*, vol. 42, no. 1, pp. 93-108, 2004.
- [121] A. Z. Broder, S. C. Glassman, M. S. Manasse and G. Zweig, "Syntactic Clustering of the Web," *Computer Networks and ISDN Systems*, vol. 29, no. 8, pp. 1157-1166, 1997.
- [122] C. Carpineto, S. Osiński, G. Romano and D. Weiss, "A Survey of Web Clustering Engines," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, pp. 17, 2009.
- [123] D. J. Lawrie and W. B. Croft, "Generating Hierarchical Summaries for Web Searches," In Proc. ACM 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, 2003, pp. 457-458.
- [124] H. Toda and R. Kataoka, "A Search Result Clustering Method using Informatively Named Entities," In Proc. ACM 7th annual ACM international workshop on Web information and data management, 2005, pp. 81-86.
- [125] D. Beeferman and A. Berger, "Agglomerative Clustering of a Search Engine Query Log," In Proc. ACM the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, 2000, pp. 407-416.
- [126] N. Duhan and A. K. Sharma, "DBCCOM: Density Based Clustering with Constraints and Obstacle Modeling," In Proc. Springer International Conference on Contemporary Computing, Berlin, Heidelberg, 2011, pp. 212-228.

- [127] A. Jain, A. Jain, N. Chauhan, V. Singh and N. Thakur, "Information Retrieval using Cosine and Jaccard Similarity Measures in Vector Space Model," *International Journal of Computer Applications*, vol. 164, no. 6, pp. 28-30, 2017.
- [128] A. Huang, "Similarity Measures for Text Document Clustering," In Proc. sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand, 2008, pp. 49-56.
- [129] K. Maher, M. S. Joshi, "Effectiveness of Different Similarity Measures for Text Classification and Clustering," *International Journal of Computer Science and Information Technologies*, vol. 7, no. 4, pp. 1715-1720, 2016.
- [130] V. Thada and V. Jaglan, "Comparison of Jaccard, Dice, Cosine Similarity Coefficient to Find Best Fitness Value for Web Retrieved Documents using Genetic Algorithm," *International Journal of Innovations in Engineering and Technology*, vol. 2, no. 4, pp. 202-205, 2013.
- [131] P. Achananuparp, X. Hu and X. Shen, "The Evaluation of Sentence Similarity Measures," In Proc. Springer International Conference on data warehousing and knowledge discovery, Berlin, Heidelberg, 2008, pp. 305-316.
- [132] D. Boley, M. Gini, R. Gross, E. H. S. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore, "Partitioning-based Clustering for Web Document Categorization," *Decision Support Systems*, vol. 27, no. 3, pp. 329-341, 1999.
- [133] J. Kaur and V. Gupta, "Effective Approaches for Extraction of Keywords" *International Journal of Computer Science Issues (IJCSI)*, vol. 7, no. 6, pp. 144, 2010.
- [134] D. B. Bracewell, F. Ren and S. Kuriowa, "Multilingual Single Document Keyword Extraction for Information Retrieval," In Proc. IEEE 2005 International Conference on Natural Language Processing and Knowledge Engineering, 2005, pp. 517-522.
- [135] C. Zhang, "Automatic Keyword Extraction from Documents using Conditional Random Fields," *Journal of Computational Information Systems*, vol. 4, no. 3, pp. 1169-1180, 2008.
- [136] Y. Matsuo and M. Ishizuka, "Keyword Extraction from a Single Document using Word Co-occurrence Statistical Information," *International Journal on Artificial Intelligence Tools*, vol. 13, no. 01, pp. 157-169, 2004.

- [137] J. Ramos, "Using Tf-idf to Determine Word Relevance in Document Queries," *Proceedings of the first instructional conference on machine learning*, vol. 242, pp. 133-142, 2003.
- [138] R. M. Alguliev and R. M. Aliguliyev, "Effective Summarization Method of Text Documents," In Proc. IEEE The 2005 IEEE/WIC/ACM International Conference on Web Intelligence, 2005, pp. 264-271.
- [139] W. B. Frakes and R. B. Yates, "Introduction to Information Storage and Retrieval Systems." In *Information retrieval: Data structures and algorithms*, vol. 331. Englewood Cliffs, NJ: prentice Hall, 1992.
- [140] F. Sebastiani, "Machine Learning in Automated Text Categorization," *ACM computing surveys (CSUR)*, vol. 34, no. 1, pp. 1-47, 2002.
- [141] K. Zhang, H. Xu, J. Tang and J. Li, "Keyword Extraction using Support Vector Machine," In Proc. Springer International Conference on Web-Age Information Management, Berlin, Heidelberg, 2006, pp. 85-96.
- [142] S. Fisher and B. Roark, "Query-Focused Summarization by Supervised Sentence Ranking and Skewed Word Distributions," In Proc. Document Understanding Conference, DUC-2006, New York, USA. 2006.
- [143] M. Maharasi and N. A. Sophia, "A Survey of Text Categorization and its Various Approaches," *International Journal of Computer Science and Information Technologies*, vol. 6, no 3, pp. 2663-2666, 2015.
- [144] P. Bolaj and S. Govilkar, "A Survey on Text Categorization Techniques for India Regional Languages," *International Journal of computer science and Information Technologies*, vol. 7, no. 2, pp. 480-483, 2016.
- [145] V. C. Gandhi and J. A. Prajapati, "Review on Comparison between Text Classification Algorithms," *International Journal of Emerging Trends & Technology in Computer Science*, vol. 1, no. 3, pp. 75-78, 2012.
- [146] H. Brücher, G. Knolmayer, and M. A. Mittermayer, "Document Classification Methods for Organizing Explicit Knowledge," *Institute of Information Systems*, pp. 1-25, 2002.
- [147] B. V. Dasarathy, "Nearest Neighbor ({NN}) Norms:{NN} Pattern Classification Techniques," *IEEE Computer Society Tutorial*, 1991.

- [148] C. C. Aggarwal and C. X. Zhai, “A Survey of Text Classification Algorithms,” *Mining text data*, Springer, Boston, MA, 2012, pp. 163-222.
- [149] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos and C. D. Spyropoulos, “An Experimental Comparison of Naive Bayesian and Keyword-based Anti-Spam Filtering with Personal E-mail Messages,” In Proc. ACM 23rd annual international ACM SIGIR conference on Research and development in information retrieval, 2000, pp. 160-167.
- [150] L. Waltman and N. J. V. Eck, “A New Methodology for Constructing a Publication-level Classification System of Science,” *Journal of the American Society for Information Science and Technology*, vol. 63, no. 12, pp. 2378-2392, 2012.
- [151] J. C. R. Miranda, J. Y. A. Llana, J. G. G. Serna and N. G. Franco, “Automatic Classification of Scientific Papers in PDF for Populating Ontologies,” In Proc. IEEE 2014 International Conference on Computational Science and Computational Intelligence (CSCI), vol. 2, 2014, pp. 319-320.
- [152] J. Protasiewicz, M. Mirończuk and S. Dadas, “Categorization of Multilingual Scientific Documents by a Compound Classification System,” In Proc. Springer International conference on artificial intelligence and soft computing, Cham, 2017, pp. 563-573.
- [153] A. Markov, M. Last and A. Kandel, “Fast Categorization of Web Documents Represented by Graphs,” In Proc. Springer International Workshop on Knowledge Discovery on the Web, Berlin, Heidelberg, 2006, pp. 56-71.
- [154] J. Bhogal, A. MacFarlane and P. Smith, “A Review of Ontology based Query Expansion,” *Information processing & management*, vol. 43, no. 4, pp. 866-886, 2007.
- [155] C. Claudio and G. Romano, “A Survey of Automatic Query Expansion in Information Retrieval,” *ACM Computing Surveys (CSUR)*, vol. 44, no. 1, pp. 1, 2012.
- [156] W. Kraaij and R. Pohlmann, “Porter’s Stemming Algorithm for Dutch,” *Informatiewetenschap*, pp. 167-180, 1994.
- [157] M. F. Porter, “Snowball: A Language for Stemming Algorithms,” 2001.
- [158] A. G. Jivani, “A Comparative Study of Stemming Algorithms,” *Int. J. Comp. Tech. App.*, vol. 2, no. 6, pp. 1930-1938, 2011.
- [159] C. Moral, A. de Antonio, R. Imbert and J. Ramírez, “A Survey of Stemming Algorithms in Information Retrieval,” *Information Research*, vol. 19, no. 1, pp. n1, 2014.

- [160] A. Callery and D. T. Proulx, "Yahoo! Cataloging the Web," *Journal of Internet Cataloging*, vol. 1, no. 1, pp. 57-64, 1997.
- [161] Blooms Taxonomy Virtual Library, <https://www.virtuallibrary.info/blooms-taxonomy.html>
- [162] B. J. Jansen, "Search Log Analysis: What It is, What's been Done, How to Do It," *Library & information science research*, vol. 28, no. 3, pp. 407-432, 2006.
- [163] S. Singla, N. Duhan and U. Kalkal, "A Novel Approach for Document Ranking in Digital Libraries using Extractive Summarization," *International Journal of Computer Applications*, vol. 74, no. 18, pp. 25-31, 2013.
- [164] S. Gupta, N. Duhan and P. Bansal, "A Comparative Study of Page Ranking Algorithms for Online Digital Libraries", *International Journal of Scientific & Engineering Research*, vol. 4, no 4, pp.1225-1233, 2013.
- [165] S. Gupta, N. Duhan and P. Bansal, "Categorization of Documents in Digital Libraries in Multi Level Taxonomy By Using Bookmark", *International Journal of Computer Engineering and Applications*, vol. XI, no. X, pp. 1-10, 2017.
- [166] S. Gupta, N. Duhan and P. Bansal, "Efficient Method of Retrieving Digital Library Search Results using Clustering and Time Based Ranking," *International Journal of Applied Engineering Research (IJAER)*, vol. 12, no 22, pp. 12305-12314, 2017.
- [167] S. Gupta, N. Duhan, P. Bansal and J. Sidhu, "Page Ranking Algorithms in Online Digital Libraries: A Survey", In Proc. 2014 3rd International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), 2014, pp. 1-6.
- [168] N. Duhan, P. Bansal and S. Gupta, "An Approach for Ranking Search Results in Digital Library using Bookmark," In Proc. 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, 2016, pp. 3424-3429.
- [169] S. Gupta, N. Duhan and P. Bansal, "Indexing Online Academic Documents In Digital Libraries: A Survey", In Proc. International Conference on Electrical, Electronics, Computers, Communication, Mechanical and Computing (EECCMC), 2018, pp. 1-6.

Appendix A

DATA SET FOR TOPIC TAXONOMY

Taxonomy is a classification system. Normally, the aim of taxonomy is to group things according to similarities in some respect such as similarities in structure, role, behavior, etc. As the Greek root "taxis" implies, it is about putting things in order. In the proposed Document Categorization system (described in Section 4.4.5) a set of multi-level topic taxonomies are used to categorize the documents. For topic taxonomy, instead of using the existing canonical taxonomies, the system considers the digital document libraries or archives of some universities for extracting the different types of categories. The archives considered for constructing topic taxonomy by proposed system are:

- *Cornell University Library arxiv.org*
- *ACM Computing Classification System-2012*

By considering the some digital document archives, the proposed system considers the categories as shown in Fig A.1.

Now, the retrieved archives are parsed, stop words such as “the” and “is” are eliminated, words are stemmed using the porter stemming algorithm and the term frequency (tf) of each word is calculated. The words are ordered by their weight after which a certain number of words are extracted with high weight value as shown in Table A.1 to Table A.9.

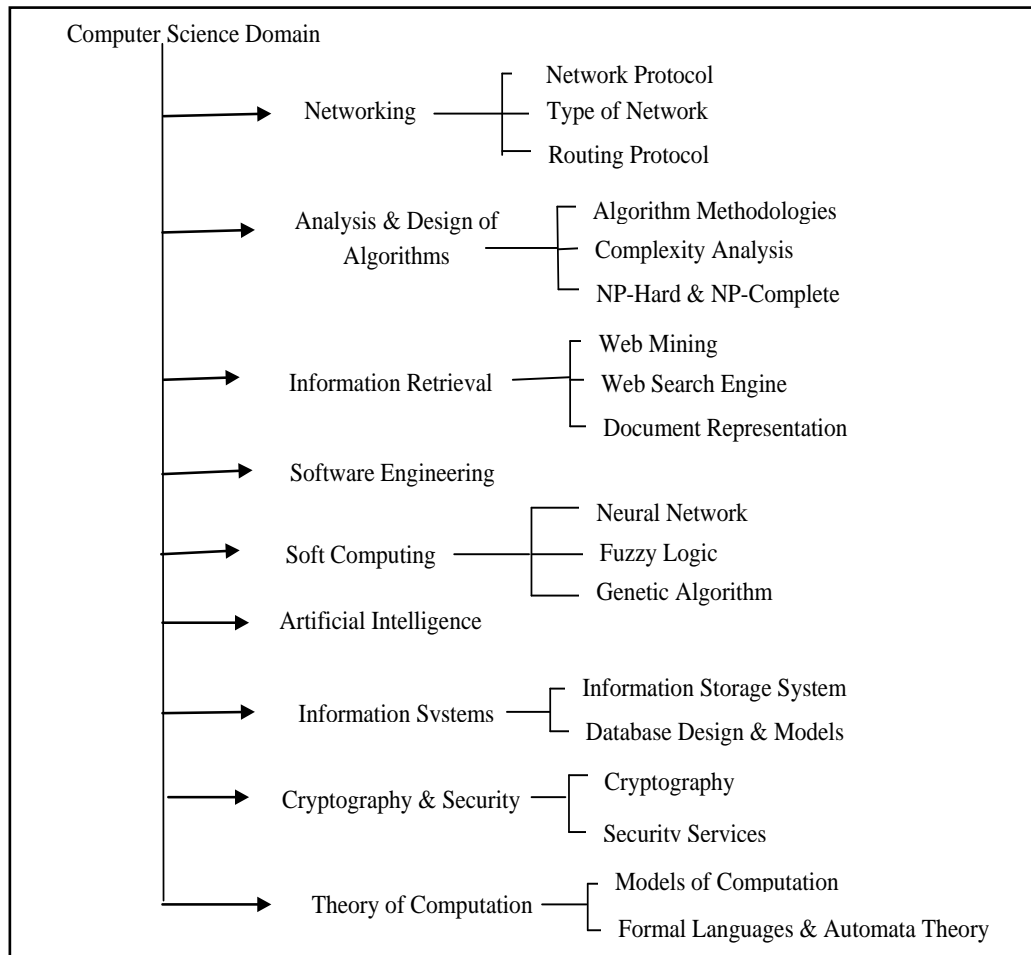


Fig A.1 Structure of Topic Taxonomy

Table A.1 Keywords of Categories

Software Engineering		Soft Computing		Formal Languages and Automata Theory	
Tool	10	Neural	13	Automata	20
Metric	11	Neuron	9	Theory	24
Debug	9	Genetic	18	Formal	17
Coupl	7	Crossover	13	Language	19
Software	23	Defuzzification	15	Grammar	15
Test	21	Expert	12	Computation	10
Cases	6	Mutation	3	Complexity	17
Program	10	Learn	2	Class	9
Cohesion	8	Chromosome	1	Machine	6
Quality	9	Habituation	1	Finite	8

Table A.2 Keywords of Categories

Cryptography and Security		Information Retrieval		Networking	
Sender	9	Web	8	Switch	15
Authentication	23	Mining	9	SNMP	10
Public	16	Rank	10	Wired	14
Key	20	Index	9	UDP	6
Cryptography	26	Crawl	7	Ethernet	5
Code	15	Information	20	ATM	10
Security	17	Search	21	Wireless	8
Information	7	Content	19	OSI	15
Cipher	13	Usage	10	Layer	21
Decode	11	Data	23	Signal	7

Table A.3 Keywords of Categories

Network Protocol		Type of Network		Routing Protocol	
SNMP	10	Wireless	11	Route	7
ATM	12	Wired	13	Token	13
OSI	15	WSN	8	Message	17
TCP/IP	8	Client	9	Adhoc	14
UDP	6	Network	11	Dynamic	7
Layer	21	Optical	7	Static	8
Ethernet	5	Topology	9	Node	12
Protocol	14	Server	15	Transfer	5
Signal	7	Manage	12	Protocol	10
Remote	6	Switch	15	OSPF	4

Table A.4 Keywords of Categories

Fuzzy Logic		Genetic Algorithm		Algorithm Methodologies	
Fuzzy	23	Genetic	18	Graph	8
Inference	13	Mutation	13	Shortest	6
Defuzzification	15	Crossover	10	Path	10
Uncertainty	8	Chromosome	8	Dynamic	14
Expert	5	Fitness	12	Backtrack	6
Logic	18	GA	8	Branch	3
Membership	13	Selection	8	Bound	2
Controller	8	Reproduction	7	Divide	9
Implication	5	Population	9	Conque	8
Linguistic	7	Evolutionary	6	Preconditioning	2

Table A.5 Keywords of Categories

Web Mining		Web search engine		Information Systems	
Web	8	Crawl	15	Data	17
Log	5	Index	13	Management	9
Analysis	10	Spam	10	Database	12
Extract	8	Detection	6	Structure	5
Integrat	4	Reformation	7	Information	13
Site	3	Suggestion	4	Storage	8
Wrap	5	Query	11	System	9
Rank	12	Log	9	Model	3
Structure	10	Architecture	11	Design	2
Usage	9	Search	17	Integration	6

Table A.6 Keywords of Categories

Artificial Intelligence		Information Storage System		Computer Vision and Pattern Recognition	
Robotics	25	Magnetic	10	Pattern	25
Machine	20	Disk	12	Machine	21
Learning	19	Tape	9	Learn	19
Neural	26	Optical	4	Recognize	11
Network	17	Flash	3	Analysis	17
Classifier	21	Memory	10	Image	21
Probability	16	Array	5	Geometry	16
Expert	11	Record	9	Structure	6
System	10	Block	5	Match	7
Reason	11	Hash	2	Training	19

Table A.7 Keywords of Categories

Database Design Models		Theory of computation		Models of computation	
Relational	9	Model	10	Computability	15
Design	8	Computation	16	Interactive	11
Entity	6	Languages	12	Probabilistic	14
Graph	12	Logic	10	Quantum	10
Hierarchal	9	Programming	11	Stream	5
Network	15	Reasoning	9	Concurrency	6
Physical	7	Data	16	Model	12
Temporal	3	Structure	12	Distribute	9
Inconsistent	5	Theory	9	Recursive	8
Model	8	Algorithm	10	Turing	2

Table A.8 Keywords of Categories

Security Services		Cryptography		Document Representation	
Authentication	11	Key	15	Document	21
Biometric	13	Management	7	Structure	9
Password	10	Public	11	Content	15
Access	9	Digital	10	Analysis	12
Control	12	Signature	8	Encoding	6
Digital	16	Symmetric	5	Feature	9
Authorization	8	Hash	3	Ontologies	8
Privacy	6	Block	10	Dictionaries	7
Pseudonymity	3	Cipher	13	Thesauri	5
Untraceability	5	Code	14	Canonicalization	3

Table A.9 Keywords of Categories

Distributed, parallel and Cluster Computing		Neural network		Complexity Analysis	
Fault	10	Neural	13	Problem	12
Tolerance	8	Learning	14	Complexity	15
Algorithm	15	Backpropagation	12	Algebraic	9
Processor	10	ADALINE	7	Theory	7
Cluster	10	Activation	5	Logic	10
Computing	15	Neuron	9	Reduction	7
Parallel	11	SVM	7	Completeness	5
Distributed	13	Habituation	4	Quantum	2
System	19	Network	16	Proof	3
Concurrency	23	Fuzzy	9	Class	8

Appendix B

DATA SETS FOR IMPLEMENTATION

This appendix gives input data sources considered for the result analysis of various indexing, ranking and query processing techniques. As it is impossible to conduct the experiments over complete WWW, the web sources at a small scale are designed. Section B.1 describes the structure and content utilized for the analysis in terms of citation graph.

B.1 THE CITATION GRAPH

A citation graph consisting of 26 interlinked publications or documents from the computer science domain has been designed which is shown in Fig. B.1. The citation graph provides information about the documents from computer science domain and their citation linkage in between. This citation graph consists of sufficient information about the link structure of documents. Each document in turn has been designed to have adequate content being used for the analysis of indexing, ranking and query processing techniques. For simplifying the calculations, a nomenclature of documents is assumed, which is given in Table B.1. Here document having title “*Web Mining Research: A Survey*” is named as ‘A’ and so on.

B.2 SEARCH LOG

For Web searching, a search log is an electronic record of interactions that have occurred during a searching episode between a Web search engine and users searching for information on that Web search engine. A Web search engine may be a general-purpose search engine, a niche search engine, or a searching application on a single Web site. The users may be humans or computer programs acting on behalf of humans. Interactions are the communication exchanges that occur between users and the system. Either users or the system may initiate elements of these exchanges.

A sample fragment of search log for analysis by proposed digital library search system is shown in Fig B.2.

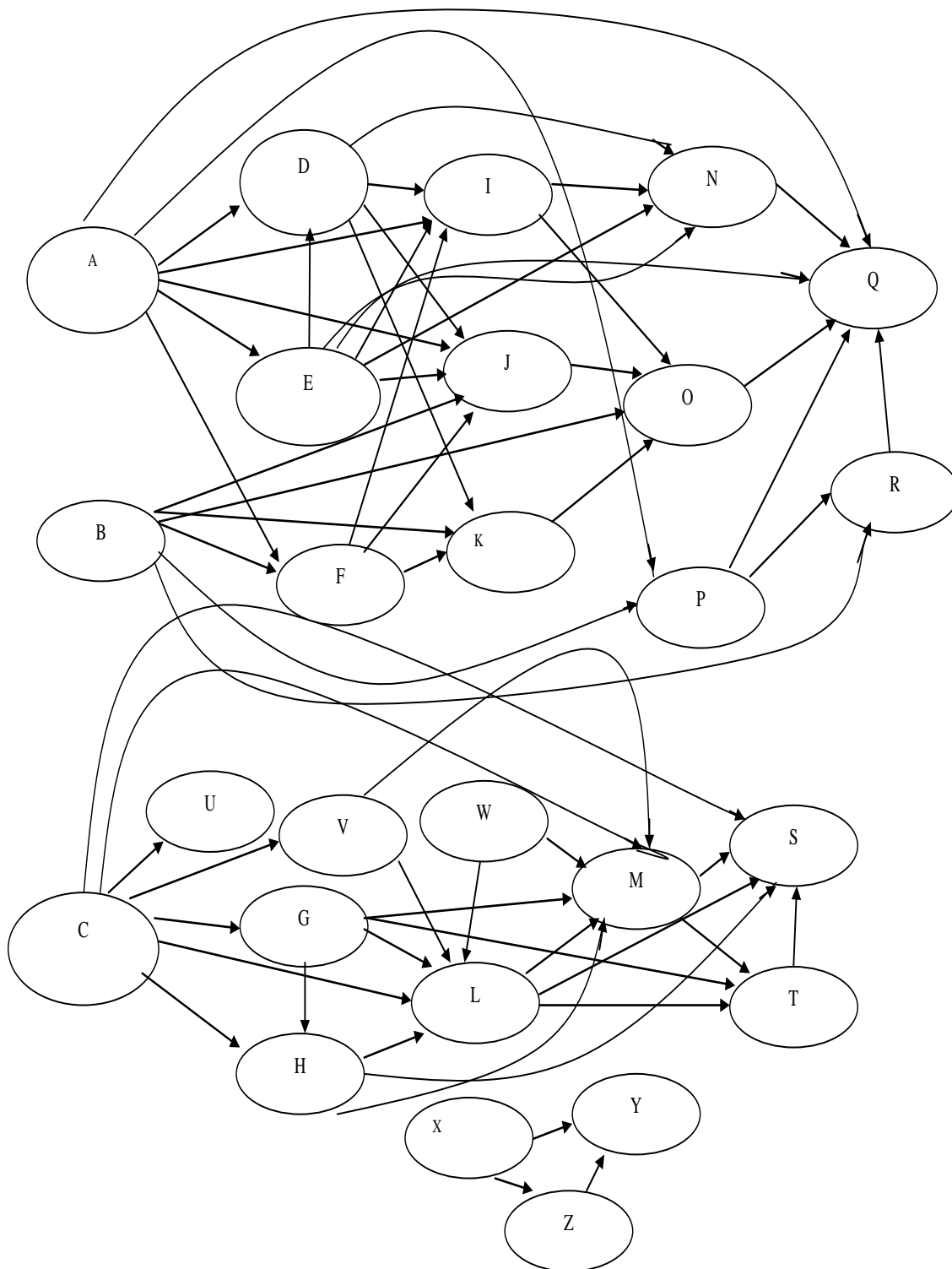


Fig. B.1 The Citation Graph for Analysis

Table B.1 The Nomenclature Scheme of Documents

Nomenclature	Document	Year of Publication
A	Web Mining Research: A Survey	2014
B	Web Crawler Architecture	2016
C	Network Security: History, Importance, and Future	2015
D	Page Ranking Algorithms for Web Mining	2011
E	A Survey- Link Algorithm for Web Mining	2012
F	How search engines work and a web crawler application	2010
G	Network Security: it's time to take it seriously	2013
H	Network Security Attacks Solution and Analysis	2012
I	Application of Page Ranking Algorithm in Web Mining	2009
J	Weighted Page Rank Algorithm Based on Number of Visits of Links of Web Page	2005
K	A Crawler-based Study of Spyware on the Web	2003
L	Network Security Using Cryptographic Techniques	2014
M	Cybercrime: A threat to Network Security	2003
N	Comparative study of Page Ranking Algorithms for Web Mining	2000
O	Mercator: A scalable, extensible Web crawler	2001
P	Web Crawler: Extracting the Web Data	1998
Q	Analysis of Various Web Page Ranking Algorithms in Web Structure Mining	1990
R	Design and Implementation of a High-Performance Distributed Web Crawler	1996
S	A Review of types of Security Attacks and Malicious Software in Network Security	1998
T	Significances and Issues of Network Security	2000
U	Application of Genetic Algorithms in Machine learning	2003
V	Genetic Algorithms in Cryptography	2010
W	Randomized algorithm approach for solving PCP	2013
X	A classical view of object-oriented cohesion and coupling	2012
Y	Software engineering: a quality management perspective	2007
Z	Software engineering: a quality management perspective	2011

SQLyog Community Edition - MySQL GUI - [New Connection - root@localhost]

File Edit Favorites DB Table Objects Tools Window Help

dsearchengine

Query

1 Result 2 Messages 3 Table Data 4 Objects 5 History

Show All or Limit 0 10000 Refresh

id	search	search_result	ip_addr	user_agent	created
43	survey of web page rank	2,6,16,11,15,	0:0:0:0:0:0:1	Mozilla/5.0 (Windows;2018-03-13 22:51:52	
44	web page rank	2,6,16,11,15,	0:0:0:0:0:0:1	Mozilla/5.0 (Windows;2018-03-13 22:52:08	
45	web crawl	2,6,16,11,15,	0:0:0:0:0:0:1	Mozilla/5.0 (Windows;2018-03-13 22:52:18	
46	cryptography techniques	2,22,6,16,11,12,15,	0:0:0:0:0:0:1	Mozilla/5.0 (Windows;2018-03-13 22:52:31	
47	software quality manage	2,22,26,6,16,25,11,12,15,	0:0:0:0:0:0:1	Mozilla/5.0 (Windows;2018-03-13 22:56:16	
48	software quality manage	2,22,26,6,16,25,11,12,15,	0:0:0:0:0:0:1	Mozilla/5.0 (Windows;2018-03-13 22:56:19	
49	web page rank	2,22,26,6,16,25,11,12,15,	0:0:0:0:0:0:1	Mozilla/5.0 (Windows;2018-03-13 22:57:14	
50	web page rank	2,22,26,6,16,25,11,12,15,	0:0:0:0:0:0:1	Mozilla/5.0 (Windows;2018-03-13 22:57:16	
51	Network security issues	22,12,	192.168.43.77	Mozilla/5.0 (Linux; 2018-03-13 23:54:33	
52	Software quality issues	26,25,	192.168.43.77	Mozilla/5.0 (Linux; 2018-03-13 23:55:41	
53	Cryptography techniques	22,26,25,12,	192.168.43.77	Mozilla/5.0 (Linux; 2018-03-13 23:56:24	
54	Cryptography techniques	22,26,25,12,	192.168.43.77	Mozilla/5.0 (Linux; 2018-03-13 23:56:35	
55	crawl	2,22,26,6,16,25,11,12,15,	192.168.43.98	Mozilla/5.0 (Windows;2018-03-13 23:58:55	
56	Survey of Web Page Rank	2,6,16,11,15,	0:0:0:0:0:0:1	Mozilla/5.0 (Windows;2018-03-18 02:12:55	
57	Survey of Web Page Rank	2,6,16,11,15,	0:0:0:0:0:0:1	Mozilla/5.0 (Windows;2018-03-18 02:14:39	
58	Survey of Web Page Rank	2,6,16,11,15,	0:0:0:0:0:0:1	Mozilla/5.0 (Windows;2018-03-18 02:14:41	
59	Web crawl	2,6,16,11,15,	0:0:0:0:0:0:1	Mozilla/5.0 (Windows;2018-03-18 02:14:53	
60	Web crawl	2,6,16,11,15,	0:0:0:0:0:0:1	Mozilla/5.0 (Windows;2018-03-18 02:14:55	
61	network security	2,22,6,16,11,12,15,	0:0:0:0:0:0:1	Mozilla/5.0 (Windows;2018-03-18 02:15:03	
62	web crawl	2,22,6,16,11,12,15,	0:0:0:0:0:0:1	Mozilla/5.0 (Windows;2018-03-18 02:15:23	
63	web page rank	2,6,16,11,15,	0:0:0:0:0:0:1	Mozilla/5.0 (Windows;2018-03-18 20:54:39	
64	web crawl	2,6,16,11,15,	0:0:0:0:0:0:1	Mozilla/5.0 (Windows;2018-03-18 20:54:55	
65	web crawl	2,6,16,11,15,	0:0:0:0:0:0:1	Mozilla/5.0 (Windows;2018-03-18 20:55:02	
66	network security	22,12,	0:0:0:0:0:0:1	Mozilla/5.0 (Windows;2018-03-18 20:55:15	

http://www.webyog.com 0 ms 85 row(s) Connections: 1 Want more Power? Get Enterprise!

Type here to search 9:54 AM 12/7/2018

Fig B.2 Sample Fragment of Search Log