

ENHANCEMENT AND INTEGRATION for CINDI SYSTEM

Krishma Dutta

A THESIS
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE & SOFTWARE ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS
FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE
CONCORDIA UNIVERSITY
MONTREAL, QUEBEC, CANADA

AUGUST 2007
© KRISHMA DUTTA



Library and
Archives Canada

Bibliothèque et
Archives Canada

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 978-0-494-34689-1

Our file Notre référence

ISBN: 978-0-494-34689-1

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.


Canada

Abstract

ENHANCEMENT AND INTEGRATION for CINDI SYSTEM

By
Krishma Dutta

The CINDI system is an assembly of inter-related subsystems, working together as a digital library for academic documents in the field of computer science. These subsystems include the CINDI Robot, which downloads scientific documents including theses, technical reports, FAQ's, academic papers and discussion groups, the CINDI Conference system and the CINDI Registration and Upload subsystem, where authors upload academic documents. In addition, there is the Gleaning subsystem that converts the non-PDF documents to PDF format and filters out the documents that are more appropriate, the Automatic Semantic Header Generator which locates information about the author, title, keywords, subject and abstract from the documents, and the CINDI Search subsystem which enables users to search for resources in the CINDI repository.

This thesis is based on the techniques that were used for the integration of subsystems, which includes porting of the Document Converter from the Windows platform to Linux. Enhancements were made to the Registration and Upload subsystem to allow multiple file uploads and improvements were made to the Graphical User Interface. The CINDI Search subsystem was redesigned to improve functionality and its interface was made more user-friendly. We have also developed an Annotation subsystem allowing users to make comments on documents in the CINDI repository.

Acknowledgements

I take this opportunity to express my gratitude towards all people who directly or indirectly contributed in this work.

I would like to take this opportunity to special thanks Dr B.C Desai for his timely guidance, support and encouragement in completion of this thesis. He is always motivated and full of ideas to share with his students for solving their problems; I am privileged to be one of them.

I am also thankful to my fellow colleague Tao Wang for inspirational and helpfulness. I am thankful to Min Huang for helping me to integrate conference system part to CINDI System. I would also like to thank Cong Zhou and Rui for helping me out with working of CINDI Robot.

Lastly, I would like to thank my parents and friends for their appreciation, motivation and believing in me.

Contents

List of Figures.....	viii
List of Tables.....	xi
List of Acronyms.....	xii
1. Introduction.....	1
1.1 Overview	1
1.2 Statement of Problem.....	2
1.3 Purposed Solution.....	3
1.4 Organization of Thesis.....	4
2. Background.....	5
2.1 Existing Academic Search Systems.....	5
2.2 Existing Annotations Systems.....	8
2.3 An overview to CINDI Subsystems.....	10
2.4 Conclusion.....	17
3. File Conversion Subsystem	19
3.1 Existing Subsystem	19
3.2 Statement of Problem.....	21
3.3 Purposed Solution	21
3.4 Modified Subsystem	23
4. CINDI Registration and Upload Subsystem	32
4.1 Overview	32
4.2 Problems.....	34

4.3	Revision of Registration and Upload subsystem Interface	35
5.	CINDI Search subsystem	41
5.1.	Overview of Search in CINDI.....	41
5.2.	Enhancement of Search Subsystem	41
5.3.	Architecture of Search Subsystem.....	42
5.4.	Automatic Semantic Header Generator Storage.....	42
5.5.	Index Creation - Full-text Index.....	43
5.6.	Graphical User Interface.....	44
5.7.	Implementing Basic Search	51
5.8.	Advanced Search Module.....	54
5.9.	Similar Documents.....	58
6.	Annotation	60
6.1	Overview of Annotation.....	60
6.2	Architecture of CINDI Annotation Subsystem.....	62
6.3	User Interface of Annotation Subsystem.....	64
6.4	Annotation Search Module.....	66
7.	Administration of CINDI Registration and Annotation Subsystem.....	68
7.1	Administrative Panel Interface.....	68
8.	Integration	74
8.1.	Overview of Integration	74
8.2.	Data Integration of CINDI.....	76
8.3.	Integrated Architecture of CINDI System.....	82

9. Experiments and Results	84
9.1 File Conversion Subsystem.....	84
9.2 Registration and Upload subsystem.....	86
9.3 Search Subsystem.....	87
10. Conclusion and Future Work	92
10.1 Conclusion.....	92
10.2 Contribution of this thesis.....	93
10.3 Future Work and Suggestions.....	94
11. References.....	95
Appendix A Stop Word List.....	101
Appendix B Impolite Words List.....	103

List of Figures

Figure 2.1 CINDI subsystems and repositories.....	11
Figure 2.2 Architecture of CINDI Robot –CndRobot.....	12
Figure 2.3 An Example of Semantic header generated by CINDI.....	16
Figure 3.1 Architecture of Document Conversion Subsystem.....	22
Figure 3.2 Ways to convert Tex into PDF.....	24
Figure 3.3 Module tree listing of Macro DOCToPDF.....	27
Figure 4.1: Architecture CINDI Registration Subsystem.....	33
Figure 4.2: Showing registration form for user, no post back required for checking fields availability.....	37
Figure 4.3 showing login screen CINDI registration system.....	38
Figure 4.4: Showing multi file upload Contributor Interface.....	39
Figure 5.1 Architecture of CINDI Search Module.....	42
Figure 5.2 Show database model of the search module.....	44
Figure 5.3 showing old search module.....	45
Figure 5.4 Shows advanced search in old CINDI system.....	45
Figure 5.5 Logical sections of modified graphical layout of CINDI search.....	47
Figure 5.6 Interface of results displayed by CINDI search module	47
Figure 5.7 shows the information displayed to user when user clicks document.....	49
Figure 5.8 CINDI Advanced search Interface.....	50

Figure 5.9 CINDI advanced search showing search ‘database’ title as mandatory parameter.....	50
Figure 5.10 Show result of query with images but not databases in document’s abstract.....	57
Figure 5.11 List of similar documents generated by CINDI.....	59
Figure 6.1 Architecture of the CINDI Annotation Subsystem.....	63
Figure 6.2 Displaying current document has two annotations and a link to add new annotations.....	64
Figure 6.3 Interface for making Annotations to CINDI system.....	65
Figure 6.4 CINDI annotation flow of information.....	66
Figure 6.5 Annotation Search results displaying title of document and annotations.....	67
Figure 7.1 CINDI Administrative Panel and its features.....	68
Figure 7.2 Manage user modules Interface	69
Figure 7.3 Interface to create new mailing lists	70
Figure 7.4 Administrative Interface for send email.....	70
Figure 7.5 Administrative Interface for manage resources.....	71
Figure 7.6 Interface to view and delete annotations.....	72
Figure 7.7 Search interface to search users by name, email and other information.....	73
Figure 7.8 Search interface when administrator asks for more information.....	73
Figure 8.1 Table schema showing fields from DOWNLOAD_STATUS.....	74
Figure 8.2 Schema of information stored about PDF converted document.....	76

Figure 8.3 Integration of Document Filter and File Conversion module.....	77
Figure 8.4 Integration of two subsystems of CINDI.....	78
Figure 8.5 Integration of Document Filtering Subsystem and ASHG.....	79
Figure 8.6 ASHG inputs information into table Document in MYSQL database.....	80
Figure 8.7 Integration of ASHG and Search Subsystem.....	81
Figure 8.8 Integrated CINDI Architecture.....	83

List of Tables

Table 5.1 Comparison between CINDI search with Google Scholar and Citeseer.....	54
Table 9.1 Time overhead saved by porting FCS from Windows to Linux Platform.....	85
Table 9.2 Statistics of number of files uploaded, time taken to file upload and the total size of files.....	86
Table 9.3 Statistics of query term searched, time take to process and number of results fetched.....	87
Table 9.4 List of first 30 results of the query term “database” searched within title of documents.....	89
Table 9.5 Search of author “desai” by using CINDI advance search within Author column in database.....	90
Table 9.6 Similarity matching of document title and abstract generated by search subsystem compared with human similarity value.....	91

Acronyms

ASHG	Automatic Semantic Header Generator
BMP	Bitmap format
CJK	Chinese, Japanese, and Korean
CINDI	Concordia Indexing and Discovery system
DFS	Document Filtering Subsystem
DOC	Microsoft Word documents
DSN	Data Source Name
DTD	Data Type Definition
FAQ	Frequently Asked Question
FCS	File Converting Subsystem
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol (World Wide Web protocol)
JPG	Joint Photographic Experts Group
PDF	Portable Document Format
PPT	MS PowerPoint documents
PS	PostScript documents
RTF	Rich Text Format documents
SHDB	Semantic Header Database
SQL	Structured Query Language
TEX	LaTex format
TXT	Text format
VQAS	Virtual Query and Answering Subsystem for CINDI
WWW	World Wide Web
XML	Extensible Markup Language
OOO	Open Office Format

Chapter 1

Introduction

1.1 Overview

Concordia Indexing and Discovery System (CINDI) is a digital library that consists of subsystems which includes the CINDI Robot (CndRobot), the Conference subsystem (ConfSys), the Gleaning subsystem consisting of File Conversion subsystem (FCS) and Document Filtering subsystem (DFS), Automatic Semantic Header Generator (ASHG), CINDI Registration and Upload subsystem, the Search subsystem and the Annotation subsystem. CINDI robot uses “pull” technique to download the research papers from the websites of universities, professors and uses the search engines such as Google and Alta Vista to initiate its search. The Conference subsystem and CINDI Registration and Upload subsystem make use of the “push” technique allowing the authors to submit their paper to the CINDI repository. The non-PDF documents in the CINDI repository are converted into PDF format by a File conversion subsystem, after conversion these documents are transferred to a Document Filtering subsystem. Document Filtering subsystem extracts the textual information from these documents and checks whether they belong to CINDI accepted list or not, if the documents are listed in DFS accepted DTD (Document Type Definition) they are transferred to accepted documents repository else they are thrown into CINDI trash. The ASHG subsystem downloads the accepted documents from DFS and extracts information regarding their title, author, abstract, keywords and subject. This information is now indexed by the search module and is used for information retrieval by CINDI Search subsystem [1, 2, 3, 4 and 5].

1.2 Statement of the Problem

Developing an effective digital library poses some serious challenges. In CINDI Digital Library Project the subsystems CINDI Robot (CndRobot), the Conference System (ConfSys), the Gleaning subsystem, Automatic Semantic header generator (ASHG), CINDI registration and Upload subsystem and the Search subsystem were required to be integrated. These components were created by previous students as a part of their graduate work and were developed in different programming languages on different platforms.

While testing the subsystems for performance and enhancement we found that all of the subsystems were developed on the Linux platform except File Conversion subsystem. FCS uses OmniFormat and PDF995 third party software and runs on Windows platform to convert different documents (DOC, HTML, PS, LaTeX, RTF, TXT and XML) into PDF. Here we found that there was lot of overhead when we transfer files from Linux platform to Windows and back to Linux.

The CINDI Registration and Upload subsystem provided limited functionality by allowing authors to upload one file at a time and restricted file size to 4 MB. This was serious drawback when author had more than one file to upload, the method becomes tedious. Hence we were required to modify the subsystem to allow increased size upload of file and allowing system to upload multiple files at one time. The search subsystem of CINDI needed modifications because the previous user interface was quite complex to be used by user. •

1.3 Purposed Solution

All of the CINDI subsystems were developed separately and they needed to be integrated. Integrating subsystems into an operational system is a complex job. The small systems can be assembled and tested in one phase, but this could not be done for the integration of the CINDI as there are numerous subsystems which are depended on each other. CINDI System required many integration phases, beginning with assembling subsystems into a bigger subsystem like Document Filtering subsystem and the new Linux platform based Document Converting subsystem were integrated into The CINDI Gleaning Subsystem, and then this subsystem was integrated with CINDI Robot. For Integrating CINDI System we have used bottom up data integration strategy.

We ported the File Conversion subsystem to Linux platform by using some of Linux internal commands and OpenOffice Basic macro language. The CINDI Registration and Upload subsystem has been modified; the new interface provides authors to upload multiple files at a time and with an increased upload limit of file. The annotation subsystem has also been added to the CINDI Registration subsystem, so users can add annotations to the documents they download, this will increase the rank of the documents [Details in Chapter 5].

The Search subsystem is redesigned and so its graphical user interface is simpler and easy to use. We have made search available to non registered users and also we have added advanced search techniques including simpler Boolean and phrase searches.

1.4 Organization of thesis

This thesis is organized as follows. In Chapter 2 the background of the CINDI digital library is discussed which includes an insight of the subsystems and their working. Chapter 3, illustrates the approach to port the File Conversion subsystem from Windows to Linux platform and an overview of the Document Filtering subsystem. Chapter 4 presents the enhancements that were made to the CINDI Registration and Upload subsystem, Chapter 5 illustrates a new search subsystem and its improved Graphical User Interface. In Chapter 6 the Annotation subsystem and its working is discussed. Chapter 7 illustrates the design and the working of Administrative Panel for CINDI Registration and Annotation subsystem. In Chapter 8 illustrate the methods and the techniques used to integrate the CINDI system. The experiments and evaluations are performed in Chapter 9 and Chapter 10 presents the conclusion, contribution of this thesis, and suggestions for future research.

Chapter2

Background

2.1 Existing Search Systems

2.1.1 Alta Vista Search [24]

The AltaVista search environment consists of several components that are important for analyzing queries: the engine itself, which is accessible from the web page www.altavista.com, and the query logs, which store information about what queries are made to the engine. AltaVista is based on weighted boolean search. The keyword or basic querying consists of a collection of words, which are ORed together. The information about the query is stored in query logs. A query log contains a timestamp, exact query terms, a cookie that tells if two queries come from same user, query type basic or advanced and submitter's ip address. When a searcher submits a query, then views a document, and returns to the search engine, the Alta Vista server logs this second visit with the identical user identification and query, but with a new time (i.e., the time of the second visit). This is beneficial information in determining how many of the retrieved results pages the searcher visited from the search engine. The query is stemmed and is checked for spelling. The results are listed by the exact match followed by the stemmed words. The query can have maximum of 393 terms which are ORed in basic search. The query log is maintained for query that share similar query terms and a new query log is created when user submits a new query.

2.1.2 Citeseer Search [24, 25]

Citeseer is an autonomous citation indexing system which indexes academic literature in electronic format. Citeseer identifies citations from the body of documents and rank paper, author, journal, etc based on the number of citations. The CiteSeer supports keyword search queries only. CiteSeer's keyword search returns a list of citations matching the query or a list of indexed articles. The articles can then be browsed by following the links between the articles made by citations. The result also shows a set of BibTeX entries that matches the query.

CiteSeer offers an interface schema *ICiteSeer* that allows keyword retrieval only: *ICiteSeer* = (*keyword*). The result schema *RCiteSeer* is more sophisticated, it is source of information based on *keyword* \rightarrow (*title, author, year, link, citations, and similar documents*). There is no spell check available for query correction and strict stemming is not implemented as results are listed differently for walk, walking, walked, etc. CiteSeer also does not use any "stop" words (such as common words like "the," which indexing typically excludes), so it is possible to search for phrases containing initials. When searching the full text of indexed articles, CiteSeer returns the header for matching documents along with the context of the articles where the keywords occur. Users can order documents according to the number of citations to them, their citations of important articles, or by date. CiteSeer can display details of particular documents, including the abstract, full text, list of citations, and an active bibliography of related documents.

2.1.3 Google Search [21, 22]

Google is the search tool of the world's largest and most powerful search engine. Google is an incredible tool allowing users to locate a wide array of literature on the Web, including web pages, tutorials, journals, abstracts, articles, theses, dissertations, books, PowerPoint presentations and technical reports from websites, forums, groups, universities, academic institutions, professional societies, research groups, and preprint repositories around the world.

The heart of Google's search technology is PigeonRank, a system for ranking web pages developed by Google founders Larry Page and Sergey Brin at Stanford University. The whole process works as follows: the documents are first parsed into words, then the words are represented by their stems, for example 'walk', 'walking' and 'walks' would be represented by the stem 'walk'. A stop list is used to reject very common words, such as 'the' and 'an', which occur in most documents and are therefore not discriminating for a particular document. The remaining words are then assigned a unique identifier, and each document is represented by a vector with components given by the frequency of occurrence of the words the document contains. In addition the components are weighted depending on the number of web pages linking to that particular page. All of the above steps are carried out in advance of actual retrieval, and the set of vectors representing all the documents in a corpus are organized as an inverted file to facilitate efficient retrieval.

The subsystem of Google that works for academic documents is Google Scholar. Google Scholar was developed by Anurag Acharya, an Indian-born computer scientist. It is tool allowing researchers to locate a wide array of scholarly literature on the Web, including scholarly journals, abstracts, peer reviewed articles, theses, dissertations, books,

PowerPoint presentations and technical reports from universities, academic institutions, professional societies, research groups, and preprint repositories around the world. What makes Google Scholar most useful is its citation index feature. Google Scholar consists of articles, with a list under each article of the subsequently published resources that cite the article. In Google Scholar, “papers with many citations are generally ranked highest, and they get a further boost if they are referenced by highly cited articles”.

Google Scholar ranks search results by how relevant they are to a query, considering the title and the full text of each article as well as the publication in which the article appeared and how often it has been cited in other scholarly literature. Further Google Scholar lists the latest articles first i.e. it list results from present to past. The advance search of Google Scholar allows users to search for documents based on authors, publication, date and subject, anywhere in document and in title of document.

2.2 Existing Annotation Systems

2.2.1 ComMentor

ComMentor [30] is a Web-based annotation system developed by the Stanford Integrated Digital Library Project and written in Perl. This annotation system is implemented as a Web browser.

When a reader views an annotated document, the annotated text is highlighted and an icon is displayed after it. The icon serves as a link to the text of the annotation. If the reader clicks on the icon a new window displays the annotation. The reader can also view

the text of the annotation by passing the mouse over the icon. To add a new annotation, the reader can highlight text in an HTML document and choose annotate.

2.2.2 AnnotatorTM

AnnotatorTM [31] is an annotation system based on the ideas of Annotation Technology developed in Java. AnnotatorTM requires the user to start at a particular URL and log into the annotation database. When the user chooses to add a new annotation, she must highlight the text she wants to annotate. The annotations are normally inline except when annotations are longer than a few lines or text, they are not added to the document as in-line notes, since this disrupts the flow of the original document.

2.2.3 Amaya

Amaya [32] is developed in Visual C++; it allows annotations as comments, notes, explanations, questions, references, examples, advice, correction or any other type of external remark that can be attached to a Web document or a selected part of the document. As they are external, it is possible to annotate any Web document independently, without needing to edit that document. When the user is making the annotation, he is asked to give details about the annotation, which constitutes the attributes of the annotation i.e. title, source document, author of document, rating and annotation text. The user can also search for annotations belonging to a particular subject or annotations made by a particular user. They can go in for a simple search where they select the subject category and enter the keywords.

2.3 An overview to CINDI Subsystems

The CINDI (Concordia Indexing and DIscovering System) system was conceived in 1994 by Dr. B.C Desai [38]. The purpose of developing such a system is to allow users easy search for and access to resources available on the Internet. It provides fast, efficient and easy access to Web documents by using a standard indexing structure and building an expert system-based bibliographic system using standardized control definitions and terms [3]. This chapter provides an overview to the CINDI subsystems, their working and an overview of the existing systems which works in similar context to that of CINDI subsystems. The figure 2.1 shows the subsystems and repositories of CINDI system.

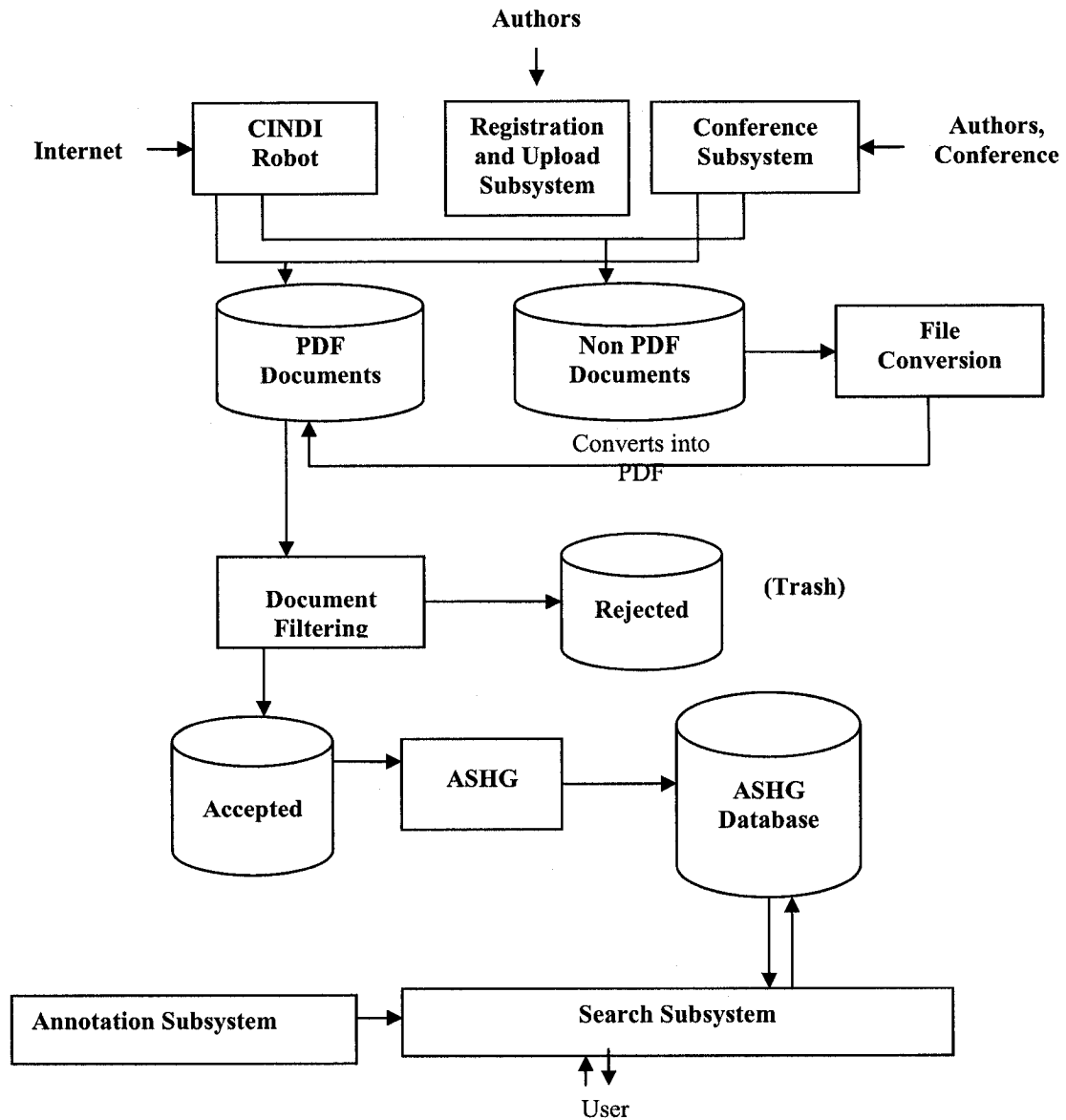


Figure 2.1 CINDI subsystems and repositories

2.3.1 CINDI Robot System [5]

CINDI web robot is a focused crawler, which starts with a set of seed URLs generated by the Seed Finder which uses a number of manually generated initial web sites from which other trusted web sites are generated. CINDI Robot extracts and follows the hyper links from the Web pages, filters unwanted documents, downloads and indexes relevant

documents in different file formats to local repository. Figure 2.2 shows the architecture of the CINDI Robot.

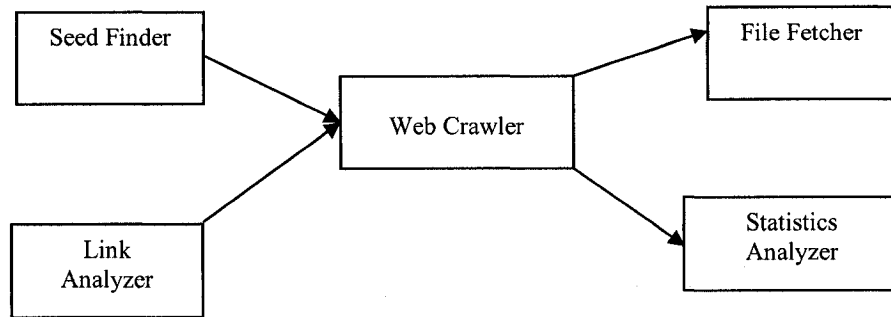


Figure 2.2 Architecture of CINDI Robot -CndRobot

- Seed Finder: The main task of seed finder is to search for research documents. CINDI robot uses Google and Alta Vista search engine to locate a set of well selected seed URL for crawling the web.
- Link Analyzer: The task of link analyzer is to identify possible irrelevant directory names based on previous crawling experiences and the filtering results for downloaded documents.
- File Fetcher: File fetcher eliminates duplicated files, filters files with undesired filenames and sizes and checks file for digital signatures.
- Statistics Analyzer: Analyzes the information gathered from previous crawling and feedback provided by CINDI filtering subsystem to acquire knowledge on crawling history and documents downloaded.

2.3.2 Conference System [36]

The Conference system (ConfSys) is an electronic paper collecting system that allows authors of the conference to submit papers. The ConfSys system includes six types of

users: administrators, authors, reviewers, program committee members, program chair, general chair and conference participants.

The administrator's main duties are: initializing ConfSys for a new conference and to create a user for general chair. When papers are called for conference, author signs up to the ConfSys and get author ID. The people who submit paper in conference are termed as authors in ConfSys. Reviewers are program committee members are selected by the general chair. Reviews give their reviews on the papers submitted by the author in ConfSys. A general chair is in charge of general management tasks during the conference preparation. The main duties of general chair are setting up conference topics, nominate reviewers, setup conference registration parameters and arrange conference. The program chair is nominated by the general chair. The main tasks of program chair are to allocate papers automatically, oversee that reviewers finish their tasks on time and make final decision of all papers. The last group of users in ConfSys is participant. Participant is anyone who wants to be part of the conference.

All the documents that are submitted by the authors are stored in the ConfSys database. These documents are in PDF format and have an associated XML file that stores the information about the title of the document, keywords, author, abstract and other information provided by author at the time of submission of paper. While integrating ConfSys system into CINDI system we have used these XML files and associated documents.

2.3.3 Gleaning System [4]

The subsystem consists of two modules first Document Conversion subsystem to convert all the documents received by robot into one unified format PDF and second one is Document Filtering subsystem that filters out documents that are not academic documents.

In previous version the File Conversion subsystem was developed on Windows platform using third party software PDF995 and Omniformat. During the enhancement of CINDI system we have ported this subsystem to the Linux system. The current file conversion module converts files of different format doc, txt, text, latex, html, ps, xml and rtf into a unified PDF format.

The second part of this system is Document Filtering Subsystem. The main function of Document Filtering subsystem is to filter out irrelevant documents. The subsystem makes use of Document type definition (DTD) for thesis, forum, academic papers, technical reports and FAQ (frequently asked questions). The subsystem extracts text out of the PDF documents converted by Document Conversion subsystem and verifies if the document is desired or not.

2.3.4 ASHG Subsystem [3]

Automatic Semantic header Generation (ASHG) module is heart of CINDI system. Since the majority of searches begin with a title, name of one of the authors, subject and abstract, CINDI made the entry of these elements mandatory in the semantic header. The main tasks of ASHG are to extract Author, Title, Keywords, Abstract and Subject

Headings. The semantic header is extracted for documents that get passed by the Document Filtering system.

ASHG tries to extract the title, the abstract, the keywords, subject and author. In order to locate the title we first look for the pattern title and what follows if found will be assumed to be the title. However, if that pattern is not found, we then select the first sentence of the document. For locating the abstract we look for the pattern abstract and if found we select everything till we meet a tab, otherwise, we select the first paragraph. The keywords are located within document by selecting the first few sentences of all the paragraphs. These sentences are then used as input to the word extractor that extracts the keywords. To locate the author(s) we look for the patterns edited by or written by in the first couple of sentences or the last couple of sentences. The e-mail is extracted by the looking for the @ symbol. The information that ASHG tries to extract from a documents is as follows:

1. **Title:** Title is the required field. It is a name given to the resource by its creator.
2. **Authors:** For the author and other responsible agents (editor, compiler), the information includes fields such as name, telephone number, fax number, and email address. At least the name or the organization and address is required.
3. **Keywords:** This field contains a list of keywords used in the resource.
4. **Abstract:** The abstract of the documents is either provided by the author or generated by ASHG.
5. **Subject:** Subject field contains a list of possible subject classifications of the resource.

Figure 2.3 shows the Semantic header generated for the document; if the desired fields are found they are added to structure else the structure is left empty. An example of the syntax used in ASHG to govern the extraction is shown below [3]:

```
<semhdr>

    <title>

        Semantic Lexicon Construction: Learning from Unlabeled Data

    <title>

    <author>

        Rie Kubota Ando

    <author>

    <abstract>

        This paper considers the task of automatically collecting words with their entity
        class labels, starting from a small number of labeled examples ('seed' words).
        We show that spectral analysis is useful for compensating for the paucity of
        labeled examples by learning from unlabeled data. The proposed method
        significantly outperforms a number of methods that employ techniques such as
        EM and co-training.

    <abstract>

    <keywords>

        Baseline algorithms, Bootstrapping, Similarity

    <keywords>

</semhdr>
```

Figure 2.3 An Example of Semantic header generated by CINDI

2.3.5 CINDI Search Subsystem [37]

This is a user search interface to perform search on documents in CINDI repository. Search can be basic search where user entered query is broken down into keywords and is searched. User can also make use of advanced search module where user can search only

within title, abstract, author or other fields. The advanced search technique includes boolean operators, phrases, truncation and wildcard operators. This is further explained in Chapter 5 in this thesis.

2.3.6 CINDI Registration and Upload Subsystem [37]

This subsystem allows users of CINDI system to register with CINDI. Registered user can select either of one of two roles annotator or contributor. Annotator is allowed to make annotations on documents in CINDI repository and contributors are allowed to upload their documents to the CINDI registration and upload subsystem. These documents are then transferred to Document Conversion subsystem for conversion into PDF if their current format is non PDF. After that they are passed through Document Filtering subsystem and then by ASHG. Documents that do not meet the intension of CINDI are discarded.

2.3.7 CINDI Annotation Subsystem [37]

Annotation subsystem works hand in hand with CINDI registration and Upload subsystem. When a user is registered with CINDI system they are given permission to make annotation on the documents they have downloaded. These annotations can later, are searched with annotations search module and advance annotation search module.

2.4 Conclusion

Reviewing the literature we derived following conclusions:

Search Subsystem: We studied that the Google, Altavista and CiteSeer have similar search systems. These systems have their own ranking techniques to rank the documents

i.e. there is no involvement of the users. They take number of citations to the documents to increase or decrease the page rank of the document. While developing CINDI search subsystem we took a different approach by allowing the user search to increase the rank of the documents. We have also provided users with advanced functionality allowing them to restrict their search only on title, author, abstract, keywords and subject.

Annotation Subsystem: We reviewed ComMentor, Annotator and Amaya annotation system to develop our CINDI annotation subsystem. These annotation subsystems allowed adding annotations within the document, attached to the document, and more. The documents in the CINDI repository consists of copyrighted work of authors hence we cannot allow them to add annotations within the document. Hence we have added annotation as separate notes that are linked to the parent document.

Integration: We studied all the existing subsystems of CINDI, and developed a bottom up integration strategy to create a unified system.

Chapter 3

File Conversion Subsystem

3.1 Overview of existing system

File Conversion subsystem (FCS) [4] is a part of CINDI Gleaning Subsystem [4]. The Gleaning subsystem consists of File Conversion subsystem and Document Filtering subsystem (DFS). CINDI system accepts documents in HTML, TXT, LaTeX, RTF, PS, DOC, XML, TEX and PDF format. These files format are converted into a unified PDF format by File Conversion subsystem and later is filtered by Document Filtering subsystem.

3.1.1 File Conversion subsystem

The purpose of File Conversion subsystem is to provide a single document format to facilitate document processing in the subsequent subsystems of CINDI. Since PDF is emerging as the current favorite format for electronic documents, it was chosen as the single format for CINDI. FCS was developed by Tong Zhang, in 2004 as a part of her graduate thesis. FCS checks for the CINDI robot database on Linux platform every 30 minutes. When CINDI robot adds non-PDF documents in its database, a daemon transfers the newly added documents securely and automatically from the non-PDF repository on Linux to repository on Windows platform for conversion. FCS employed third party software PDF995 and OmniFormat to convert downloaded documents in PDF files. After conversion, FCS sends the PDF version of documents back to the temporary repository on the Linux platform for filtering.

3.1.2 Document Filtering Subsystem [4]

DFS is the other half of the Gleaning subsystems that filter out irrelevant documents from CINDI repository. DFS matches the document one by one with the DTD of theses, technical reports, academic papers, and FAQs. If the document matches one of them, DFS treats this file as an acceptable document and saves it in the permanent repository. If this document fails all of these DTDs, it is sent to the CINDI trash.

The documents downloaded from the Web by the CINDI Robot and CINDI registration and upload system are processed by DFS into two subsets: *accepted* and *rejected* as described in the following formulas; the former set contains documents which are accepted and the latter set contains files which are rejected. The academic documents like thesis, reports, paper and FAQ belongs to our accepted list.

$$downloads = accepted \cup rejected$$

$$accepted = \{f : |f \wedge downloads \wedge (f \in thesis \vee f \in report \vee f \in paper \vee f \in faq) \}$$

$$rejected = \{f : |f \wedge downloads \wedge (f \notin thesis \wedge f \notin report \wedge f \notin paper \wedge f \notin faq) \}$$

If DFS finds a document $d \in accepted$, then d will be accepted; in contrast, if $d \notin accepted$, then DFS will reject this document and put it into the CINDI trash.

When DFS starts processing a document it assumes that it could be one of the acceptable types of documents and tries to match it to its Document type definition (DTD) that consists of front matter, body and back matter. The front matter has a title, abstract, table of contents and more, the body consists of paragraphs in particular format and the back matter is bibliography, references and appendix.

3.2 Statement of Problem

In order for CINDI system to grow and prosper, we need to broaden its usability and speed. While testing the File Conversion subsystem we discovered some drawbacks. These drawbacks are listed below.

3.2.1. Time Overhead: The File Conversion subsystem was wasting a lot of time in downloading files from Linux platform to Windows platform and uploading the converted file to Linux server. The process was tedious because on average CINDI robot downloads thousands of documents every day and these documents need to be converted. In order to download these documents and upload back we have some serious overhead.

3.2.2. Linux Format files conversion problem: As the previous Document Converter uses Windows PDF converter i.e. PDDF995, FCS used another program to convert LaTeX files into PDF.

In order to fix these problems we have ported the DFS to Linux machine. The next section discusses about how this system was ported to Linux and the challenges that we faced while developing DFS on Linux platform.

3.3 Purposed Solution

The CINDI Robot is a subsystem used for collecting significant documents related to the topics of interest for a particular CINDI file from the Web. It retrieves information from remote sites using standard HTTP protocols through graph-traversal algorithm. It first downloads an initial set of URLs and then parses the associated documents to extract additional URLs pointing to other documents. The documents

are categorized into two storage PDF and non- PDF. Non-PDF documents have one of these extensions TXT, PS, RTF, HTML, XML, DOC and LaTeX. We will begin step by step how we converted non-PDF files into PDF. Figure 3.1 shows the architecture of DFS.

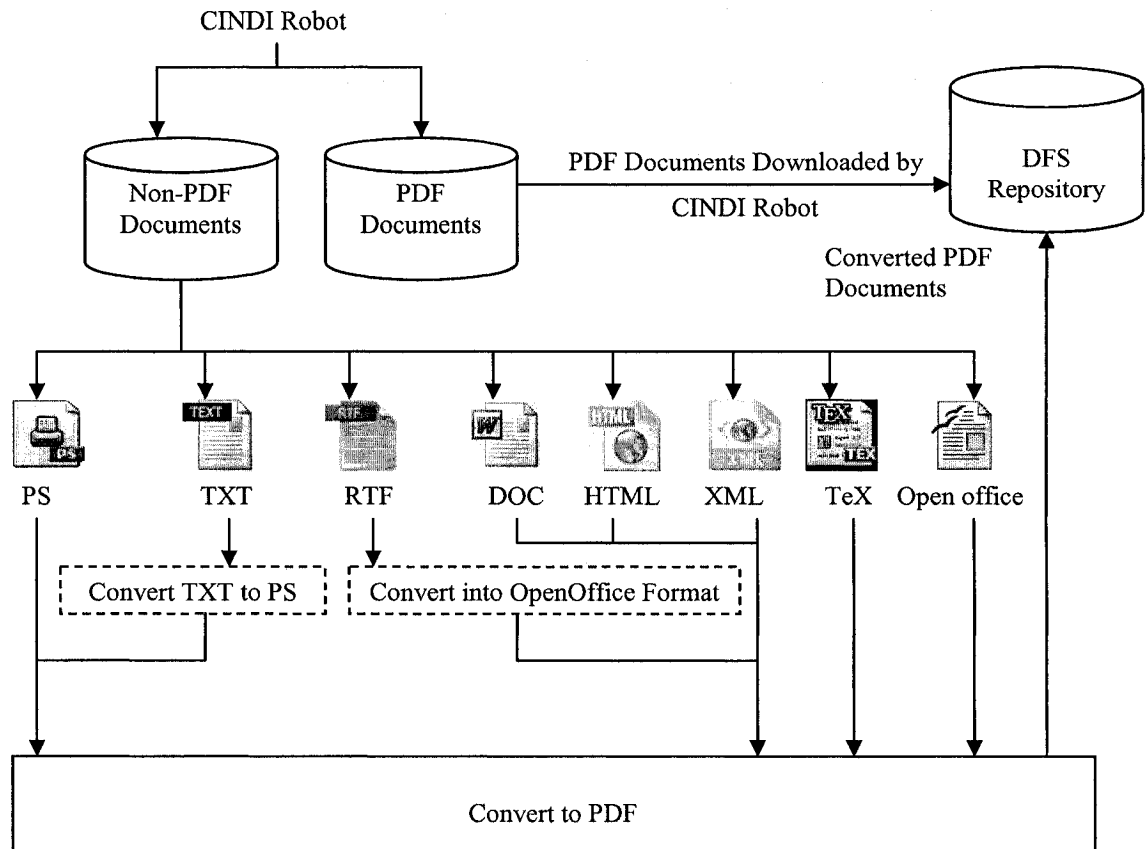


Figure 3.1 Architecture of Document Conversion Subsystem

3.4 Modified Subsystem

3.4.1 Converting TXT documents into PDF

The .txt is a filename extension for files consisting of text with very little formatting. To convert TXT file into PDF we have first converted TXT document into PS and then converted PS into PDF.

Step1: Convert TXT document into PS document, we make use of GNU a2ps [6] filter. GNU a2ps is a filter which generates PostScript from various formats, with strong support for many alphabets, and customizable layout. Calling a2ps is fairly simple and syntax is shown below:

```
Shell> a2ps [OPTION]... [FILE]...
```

```
Shell> a2ps doc.txt doc.ps
```

Now we have file doc.ps from our text file doc.txt. The second step is to convert this PS file into PDF. We have achieved this by making use of ps2pdf filter.

Step2: The ps2pdf filter converts PostScript to PDF using ghostscript [7]. The syntax of using ps2pdf is given below:

```
Shell> ps2pdf [options...] {input.[e]ps|-} [output.pdf|-]
```

```
Shell> ps2pdf [options...] {doc.ps|-} [doc.pdf|-]
```

The resultant document is now PDF named doc.pdf in our case.

3.4.2 Converting PS documents into PDF

To convert PS to PDF we have made use of ps2pdf filter. The procedure is same as described while converting text document step2.

3.4.3 Converting LaTeX documents into PDF

There are two ways to convert Tex documents into PDF. These two ways are shown in figure 3.2.

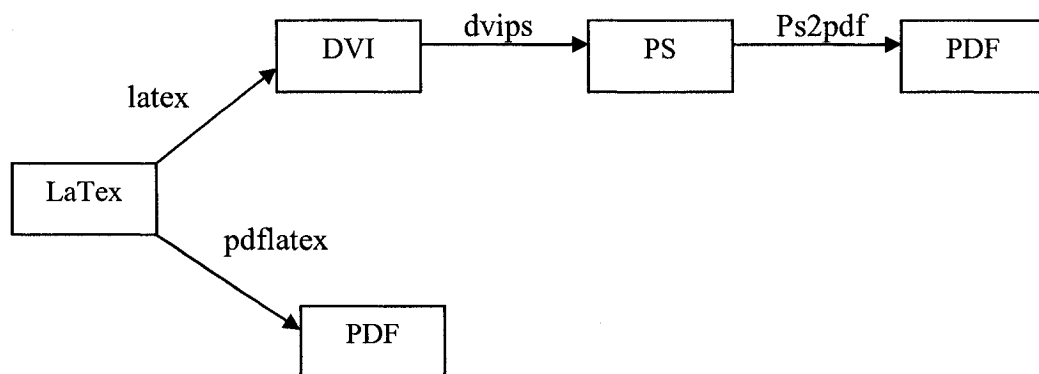


Figure 3.2 Ways to convert Tex into PDF

In figure 3.2 we see that one way to convert LaTeX documents into PDF is to convert .tex file into .dvi document using latex [8] filter and then convert .dvi file into .ps file using dvips filter, now convert ps document to PDF using ps2pdf filter. The second way is simpler i.e. to make use of pdflatex filter and convert TEX document to PDF in one step. We have made use of second approach i.e. pdflatex [9] filter to convert Tex files into PDF. The pdflatex filter is based on the original eTeX sources and Web2c, and has been successfully compiled on UNIX, Win32 and DOS systems. Latex only handles figures that are PS or EPS. Currently CINDI robot is not downloading associated PS or EPS files; therefore we are only converting LaTeX files with text only. The further

improvement can be to download all latex files into one directory and then convert them as one file. If the file argument has no extension, ".tex" will be appended to it. The syntax of the command is listed below:

```
Shell> pdflatex file.tex file.pdf
```

3.4.3 Converting DOC, RTF, XML and HTML documents into PDF

In order to convert Doc, RTF, XML and HTML documents into PDF we have made use of OpenOffice. We have developed macros in OpenOffice Basic language that convert these kinds of documents into PDF documents.

OpenOffice.org [10]

OpenOffice.org is an Open Source project which provides an office suite available for many operating systems including Linux, Microsoft Windows, MacOS X, Solaris, FreeBSD, OpenVMS and IRIX. OpenOffice.org aims to compete with Microsoft Office and emulate its look and feel where suitable. It can read and write most file formats found in Microsoft Office, and many other applications; an essential feature of the suite for many users. OpenOffice.org has been found to be able to open files of older versions of Microsoft Office and damaged files that newer versions of Microsoft Office itself cannot open without specified add-ons [11].

OpenOffice.org is a collection of applications [12] that work together closely to provide the features expected from a modern office suite. Many of the components are designed to mirror those available in Microsoft Office.

OpenOffice Basic [13]

OpenOffice.org Basic is a programming language similar to Microsoft Visual Basic for Applications (VBA) and is based on StarOffice Basic. OpenOffice.org Basic is available in the Writer and Calc applications. It is written in functions called subroutines or macros, with each macro performing a different task, such as counting the words in a paragraph. OpenOffice.org Basic is especially useful in doing repetitive tasks that have not been integrated in the program.

OpenOffice Basic is a macro language of OpenOffice. We can start up OpenOffice from the Linux or Windows command-line prompt with instructions to run a particular macro, and we can even pass a filename as a parameter to that macro. Adding the -invisible switch to the command line tells OpenOffice to start up without the graphical user interface (GUI). Putting all these features together and *we can use a command line that can convert a Microsoft Office file to an OpenOffice file or an Acrobat file with no use of the GUI*. To convert a hundred files, we can use a scripting language or a shell script to create a batch file. Below is the list of documents which have used this technique to convert parent form into PDF using OpenOffice Writer and OpenOffice Basic macro language.

3.4.4 Conversion of DOC documents into PDF: To convert DOC documents into PDF we have made use of OpenOffice macros. To create a macro module in OpenOffice to store these macros, pick Macros from the OpenOffice Tools menu and then pick Organize Macros, OpenOffice.org Basic, Organizer, and New to create a new module. Name the module *DOCtoPDF* and click the Close button. The new Module will show up

as */My Macros/Standard/DOctoPDF* in the macro module tree listing, as shown in figure 3.3.

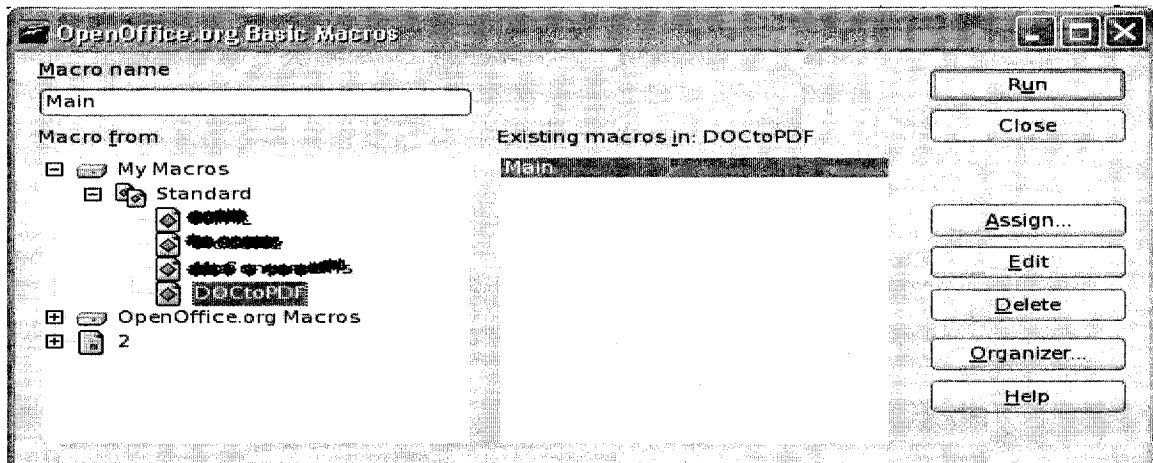


Figure 3.3 Module tree listing of Macro DOctoPDF

Now we select DOctoPDF macro and then click Edit. Here we add following listing as code and save the module.

' Save document as an Acrobat PDF file.

```
Sub SaveAsPDF( cFile )
    cURL = ConvertToURL( cFile )
    oDoc = StarDesktop.loadComponentFromURL( cURL, "_blank", 0, _
        Array(MakePropertyValue( "Hidden", True ),))

    cFile = Left( cFile, Len( cFile ) - 4 ) + ".pdf"
    cURL = ConvertToURL( cFile )

    ' Save the document using a filter.
    oDoc.storeToURL( cURL, Array(
        MakePropertyValue( "FilterName", "writer_pdf_Export" ),)

    oDoc.close( True )
End Sub
```

Here have made use of “writer_pdf_Export” filter provided by OpenOffice to convert our doc document into PDF with the same filename and .pdf extension. This module runs as a part of a batch script from command line and when ever we find a doc file in our temp non-PDF directory we call this macro. The syntax to call this macro is listed below.

ooffice -invisible

```
"macro:///Standard.DOCtoPDF.SaveAsPDF(/home/c_intgrt/temp/test.doc)"
```

This command will run in invisible mode and will provide test.pdf as output. After conversion is done we move the converted file into PDF repository of CINDI.

3.4.5 Conversion of XML, HTM and HTML documents into PDF

We have followed the same procedure describe above to convert XML, HTM and HTML files into PDF. We have created a macro HTMLtoPDF that converts these files into PDF.

The code for this macro is listed below.

```
Sub SaveAsHTML(cFile)
  cURL = ConvertToURL( cFile )
  oDoc = StarDesktop.loadComponentFromURL( cURL, " _blank", 0, _
    Array(MakePropertyValue( "FilterName", "HTML (StarWriter)" ),_
    MakePropertyValue("Hidden",True)) )

  cFile = Left( cFile, Len( cFile ) - 5 ) + ".pdf"
  cURL = ConvertToURL( cFile )

  ' Save the document using a filter.
  oDoc.storeToURL( cURL, Array(_
    MakePropertyValue( "FilterName", "writer_pdf_Export" ) ) )

  oDoc.close( True )
End Sub
```

The code for converting HTML and XML into PDF is similar to converting DOC into PDF except for use of HTML (StarWriter) filter that parses the HTML.

The call to this procedure is listed below.

ooffice-invisible

```
"macro:///Standard.HTML.SaveAsHTML(/home/c_intgrt/temp/test.html)"
```


3.4.6 Conversion of RTF into PDF

Conversion of RTF document into PDF takes place in two steps. In first step we convert RTF document into OpenOffice document and in step two we convert the converted OpenOffice document into PDF.

Step1: Convert RTF to OpenOffice format: This is achieved by creating another macro to convert RTF into OpenOffice format. The code for this macro is listed below.

' Save document as an OpenOffice file.

```
Sub SaveAsOOO( cFile )
  cURL = ConvertToURL( cFile )
  oDoc = StarDesktop.loadComponentFromURL( cURL, "_blank", 0, _
    Array(MakePropertyValue( "Hidden", True ),))

  Select Case LCase(Right(cFile,3))
    Case "doc"      ' Word file.
      cFileExt = ".doc"
    Case "rtf"      ' Rich Text Format File.
      cFileExt = ".rtf"
    Case "xxx"
      cFileExt = ".xxx"

  End Select

  cFile = Left( cFile, Len( cFile ) - 3 ) + cFileExt
  cURL = ConvertToURL( cFile )

  oDoc.storeAsURL( cURL, Array() )
  oDoc.close( True )
```

End Sub

Step 2: After the above macro convert RTF file into PDF file we can use the same conversion module used for converting DOC into PDF. The syntax to achieve this conversion is listed below.

ooffice -invisible

"macro:///Standard.RTFtoOOO.SaveAsOOO(/home/c_intgrt/temp/test.rtf)"

ooffice -invisible

"macro:///Standard.DOCtoPDF.SaveAsPDF(/home/c_intgrt/temp/test.xxx)"

The above command will give out test.pdf as output file.

3.4.6 Conversion of Open office format into PDF

In order to convert OpenOffice files into PDF we can make use of same macro that we have created for converting DOCToPDF. So, this conversion will be achieved in one step syntax is stated as follows.

```
ooffice -invisible  
"macro:///Standard.DOCToPDF.SaveAsPDF(/home/c_intgrt/temp/test.odt)"
```

Working of Document Conversion Subsystem

In the above section we discussed how we achieved to convert different types of files into the unified PDF format. In this section we will discuss how they are integrated together to work as CINDI converter. We have created a C++ daemon on Linux that monitors the non-PDF directory. This is achieved by creating Linux cron job. Cron is the name of program that enables UNIX or Linux users to execute commands or scripts automatically at a specified time/date. When cron job finds that there are files that are added by robot in the directory it following the procedure listed in algorithm below.

Algorithm Convert into PDF

Input: TXT, PS, RTF, HTML, XML, DOC and TeX documents

Output: PDF Documents

```
for file in /non-pdf repository  
{  
    do  
  
        if  
        // Text File  
        {  
            a2ps file.txt file.ps  
            ps2pdf file.ps file.pdf  
            mv file.pdf /pdf-repository  
        }  
        else if  
        // LaTeX File
```

```

    {
        pdflatex file.txt file.pdf
        mv file.pdf /pdf-repository
    }
    else if
    // PostScript File
    {
        ps2pdf file.ps file.pdf
        mv file.pdf /pdf-repository
    }
    else if
    //HTML OR XML File
    {
        ooffice -invisible
        "macro:///Standard.HTML.SaveAsHTML(/home/c_intgrt/temp/file.html)"
        mv file.pdf /pdf-repository
    }
    else if
    // DOC file
    {
        ooffice -invisible
        "macro:///Standard.DOCtoPDF.SaveAsPDF(/home/c_intgrt/temp/file.doc)"
        mv file.pdf /pdf-repository
    }
    else if
    //RTF File
    {
        ooffice -invisible
        "macro:///Standard.MyConversions.SaveAsOOO(/home/c_intgrt/tempfile.rtf)"
        mv file.pdf /pdf-repository
    }
    else if
    //OpenOffice Format
    {
        ooffice -invisible
        "macro:///Standard.MyConversions.SaveAsPDF(/home/c_intgrt/temp/file.xxx)"
        or
        ooffice -invisible
        "macro:///Standard.MyConversions.SaveAsPDF(/home/c_intgrt/temp/file.odt)"

        mv file.pdf /pdf-repository
    }
done
}

```

Algorithm: Document Conversion CINDI Gleaning Subsystem

Chapter 4

CINDI Registration and Upload subsystem

4.1 Overview

The CINDI Registration and Upload subsystem [14] enrolls different kinds of users to become contributing authors of the CINDI system. These users play different roles while making use of CINDI search; they can choose their role to be a contributor or a user.

- Guest is a person who is allowed to make search and download documents from CINDI repository.
- User is a person with an authorized access to make annotations for the resources in the CINDI repository.
- Contributor is a person who is a potential contributor of academic document to CINDI system. A contributor can upload a document to the CINDI system; such documents are verified and if found suitable are added to the CINDI repository.

An unregistered user can register in the CINDI system by filling in a registration form. The form inputs number of fields and on successful completion of the form an email is sent giving a randomly generated password and the username which the user email is given in the time of registration. Once user is registered and has a username and password they can login into the CINDI system. The system checks whether the username and password are correct by sending information to the server, if the information is incorrect then an error message is displayed. If the username and password match the users get redirected to Search interface, as users can only search and

annotate, whereas if the user is contributor they are redirected to upload resource interface.

The figure 4.1 shows the architecture of the CINDI registration and Upload system.

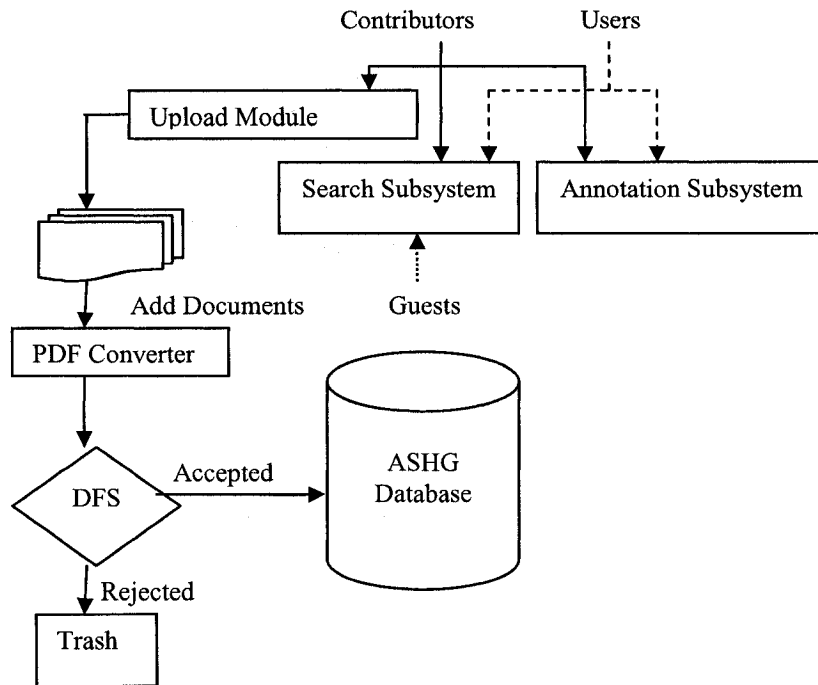


Figure 4.1 Architecture CINDI Registration and Upload Subsystem

The architecture of the CINDI registration allows guests and registered users to search in CINDI repository without signing in. Authors can upload their resources into CINDI system by becoming contributors. When a document is uploaded into CINDI registration it is passed to File Conversion subsystem if it is not in PDF format it is converted and then document is filtered to verify if it fulfills the requirement to fit in CINDI repository. If the document gets accepted by filter it is added to CINDI else it is sent to trash. The contributor is not yet notified about the document getting accepted or rejected by the CINDI.

The contributor is allowed uploading the document in one of these formats HTML, RTF, DOC, PS, TXT, PDF and TEX. The PDF converter checks if these are in format other than PDF.

The registration and upload subsystem validation checks are developed using AJAX [15, 16] technology. The reason for using AJAX is to improve the user experience, faster response time and providing dynamic content without post backs some validation is required. The upload of the document module is redesigned from previous version developed by Y. Xiaome in 2001 as a part of master's major report, to select one or number of files to upload at one time. The upload module allows the contributor to upload files in batch mode. The files are uploaded to the CINDI directory from where it is picked up by PDF converter.

4.2 Problems

In this section, we will come across most common problems that were found while testing CINDI Registration and Upload subsystem. Some of these problems are development techniques, interface designing and others were modifications. Below is the list of problems that were found:

4.2.1. *Upload Module:* After contributors get validated they can upload their research paper. Incase user have more than one file to upload they need to upload one by one. This was tiring job when contributor has number of documents, besides there was no way by which user can cancel upload if by mistake he made selection to wrong file.

4.2.2. *Unnecessary Post backs of Information:* The old system makes use of PHP and MYSQL, which makes posting of whole webpage to do a small

validation check. This means if the username was already in use the user needs to click on submit button and now the page sends data to server which sends error that username in use. There are many other ways in which these posts back of information created long overheads while registering users/contributors.

4.2.3. *Forced Login:* The user/contributor was required to login to system in order to make a search.


4.3 Revision of Registration and Upload subsystem Interface

Contributor/ User Interface: The first step in order to become user/ contributor to CINDI system is to be registered. This is done by entering the registration form which is developed in PHP using AJAX techniques. The figure 4.2 shows the input for contributor registration form for registering to CINDI. The following information is required to be filled for getting registered into the CINDI registration system.

- First Name: First Name is the First Name of user/contributor; it is a mandatory field of information.
- Last Name: Last Name is the Last Name of user/contributor; it is a mandatory field of information.
- Nickname: Nickname is the screen name of user/contributor; it is a mandatory field of information, it has to be unique within CINDI system. This information is used by Annotation subsystem of CINDI to post annotations.
- Email: Email is the email address of the user/contributor; it is also used as login id for the CINDI registration and annotation system. It is a mandatory field of information.

- **Country:** This piece of information is required when user registers as a contributor. It is mandatory field in case of contributor, for normal users this option is not visible.
- **Organization:** This piece of information is selected by contributors stating their organization or institution. Organizations are listed only after country is selected as they vary from country to country. If the organization is not in CINDI database user can send an email to webmaster and come back later to register as contributor.
- **Secret Question/ Answer:** This information helps users to reset their password. If user forgets his password he needs to provide this information that allows him to reset password right away, otherwise he needs to reset it by providing his email.

CINDI Indexing and
Discovery System



Contributor Registration Form

First name:

krishma

Last name:

dutta

Nickname:

cindi

Check Nickname

Sorry that username is not available try
these

☐ cindicindi

☐ cindiconcordia

**Email /User
ID**

k_dutta@encs.concordia.ca

**Re-type
Email: /User
ID**

k_dutta@encs.concordia.ca

Country:

Belize

Organization:

American University of the Caribbean

**Secret
Question:**

What is your father's middle name?

Answer:

hello

SUBMIT

Figure 4.2: Showing registration form for user, no post back required for checking fields availability

The figure 4.2 shows that the nickname 'cindi' is not available and asks user to select some different username and lists a few examples. The user can now either select from the listed provided by the system or try with some other username. When registration is successful an email is sent to user with information regarding username and password. Now user logs into the system by the welcome registration page. This page checks for username and password, if information provided is not correct XMLHttpRequest is returned with a value of false and an error message is displayed without any kind of post back. This can be seen in figure 4.3

Welcome to the CINDI System

Username or password incorrect. Please try again

Error checking using Ajax

The CINDI system is a virtual library that caters to academic and professional users and contributors; currently it is limited to the domain of Computing Science. The users are defined as those who search through the library virtual catalogs to find the information that they need and download it. The contributors are defined as those who classify and upload the information to enrich the library source.

If you would like to contribute high quality original articles, reports etc., please access Contributor Registration Now. If you would be interested in using our system, please Sign up for User Registration.

In case of any problems in registration you can contact [webmaster](#).

Please enter Username and Password

Login: bob

Password: *****

Login Reset

[New Contributor SignUp](#) [New User SignUp](#)

©2006 - CINDI Research Group

Figure 4.3 showing login screen CINDI registration system

4.3.1 Upload Interface

The contributor after logging in to the system can upload files to CINDI repository, edit personal information, Search CINDI system and make annotations. The figure 4.4 shows the file upload module developed in PHP, HTML, JavaScript and AJAX allowing multiple file uploading. The reason for using AJAX is that the browser-based file uploads (used in previous registration and upload system) involves the HTML `<input type="file">` tag. When uploading file size exceeds 10MB it often causes a very poor user experience. Once a user submits the file, the browser will appear to be inactive while it attempts to upload the file to the server. While this is happening in the background, many impatient users would start to assume that the server has “hanged” and would try to submit the file again. This of course, only makes matters worse.

In an attempt to make uploading of files more user-friendly, many sites display an progress animation (such as a rotating icon) once the user submits the file. Although this technique may be useful in keeping the user distracted during the upload to the server, it offers very little information on the status of the file upload. Another attempt at solving the problem is to implement an ftp program that uploads the file to the server through FTP. The drawback with this solution is that it limits users to have knowledge of ftp software. Hence we have applied a fresh approach and implemented an AJAX-powered component that will upload the file to server, monitors the actual progress of a files upload request and allow multiple file upload.

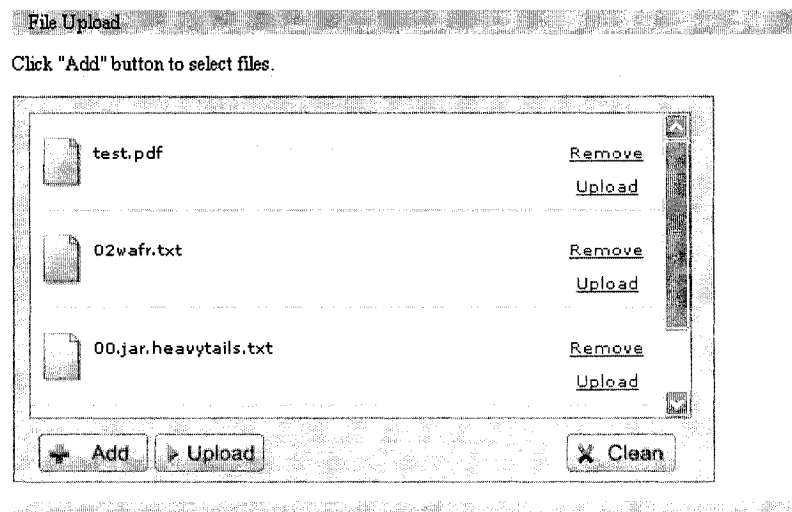


Figure 4.4: Showing multi file upload Contributor Interface

Steps to upload files to CINDI repository are as follows:

1. Contributor clicks on add button to add a document to upload list. More than one file can be added by clicking on add button and browsing the location of file.
2. The document can be removed from the list by clicking on remove link. All the files in the upload list can be uploaded in at batch by clicking on upload button or

individually by clicking on upload link. There is a progress bar for each file getting uploaded to the server providing information about the amount of file uploaded.

3. To clear previous uploaded list contributor can click clean button which refreshes the upload module and erases the history of Upload.

4.3.2 Personal Information Interface

We have provided interface to CINDI users to manage their personal information. Users can modify their personal information that they have entered at time of registration. This allows them to change their first name, last name or reset password.

If user doesn't remember their password they can reset their password by receiving an email or answering the challenge question filed during time of registration. The user can delete their account from CINDI by filling unregister form.

The users and contributors can make annotations on documents in CINDI repository by making use of Search subsystem and Annotation subsystem explained in Chapter 5 and Chapter 6.

Chapter 5

CINDI Search Subsystem

Overview of Search in CINDI

While an immense amount of material is now easily accessible on the Web, locating specific information remains a difficult task. An inexperienced user may find it next to impossible to find the information she wants; even an experienced user may miss relevant Web pages. Due to the size of the World Wide Web and its inherent lack of structure, finding what one is looking for can be a challenge.

The CINDI system discovers research documents using the CINDI robot by a pull technique from websites and a push technique using the ConfSys and CINDI's registration and Upload subsystem. All documents are converted to a unified format PDF and are passed through CINDI Filtering subsystem to delete inappropriate ones. These documents together form a repository for which CINDI search Module is built. The information about these documents is stored in MySQL database.

5.1 Enhanced Search Subsystem

Due to the enormous amount of academic documents available in CINDI repository, the search module is essential for finding specific and appropriate information. We have made enhancement for Search module interface and backend functionality associated with it for implementing these features. We preserved features from old version and have implemented them in manner that it fits into usability of present search module.

5.2 Architecture of Search Subsystem

The CINDI Search system architecture consists of three main components: (i) a document parser and database creator Automatic Semantic Header Generator (ASHG) [3], (ii) a module that locates the desired document, (iii) a database browser interface that supports search by keyword and browsing by similarity links. Figure 5.3 is the diagram of the architecture of CINDI search module.

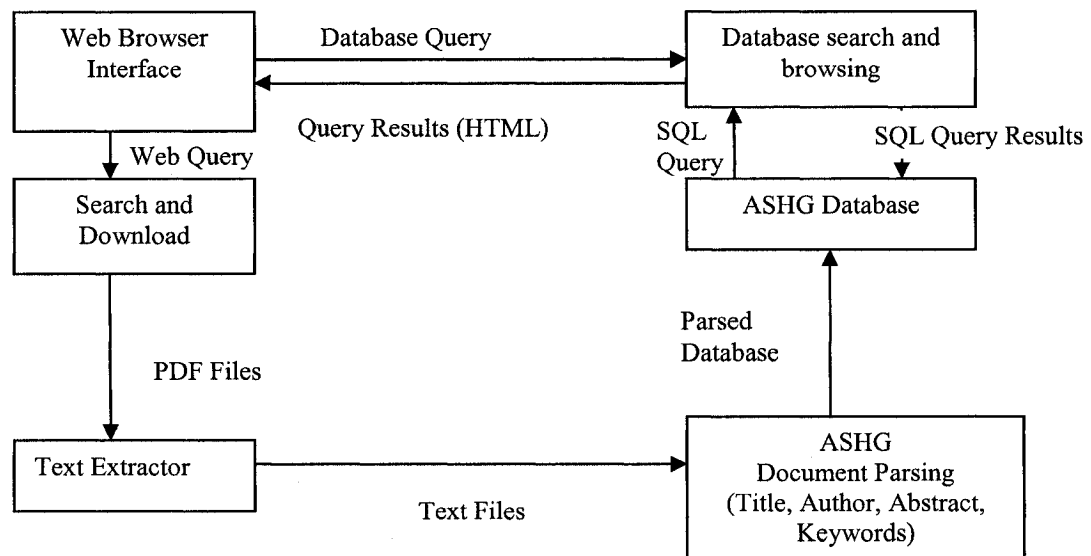


Figure 5.1 Architecture of CINDI Search subsystem

5.3 Automatic Semantic Header Generator (ASHG) Storage

The automatic semantic header database [3] consists of information about the documents that are parsed through automatic semantic header generator. The documents in CINDI repository are academic documents that are uploaded into repository in two ways:

- 4.3.1. Push Paradigm: means that the documents is submitted by authors
- 4.3.2. Pull Paradigm: means that the documents are fetched by the CINDI robot

These documents are parsed and title, authors, abstract, keywords and subject information is extracted and inserted into database. This information is located for all the documents and is inserted into the MYSQL database. This forms the final database that is used for searching and locating the information in the CINDI repository.

5.4 Full-text Index

A *full text index* consists of an entry for every occurrence of each word, and hence is generally the largest in size of the indexes discussed; the index also allows the highest *accuracy* (but not necessarily highest precision). This large index increases I/O needed for searching, but on the other hand there is no need to scan documents for stop words [17]. Hence a full-text index is implemented to meet the search goals of CINDI Search module.

Full-text index allows fast and flexible retrieval of keyword-based query of text data stored in a database. In contrast to the LIKE predicate, which only works on character patterns, full-text queries perform linguistic searches against this data, by operating on words and phrases based on rules of a particular language. The performance benefit of using full-text search can be best realized when querying against a large amount of unstructured text data. A LIKE query against millions of rows of text data can take minutes to return; whereas a full-text index query can take only seconds or less against the same data, depending on the number of rows that are returned.

The full-text indexes can be built on columns that contain char, varchar2, text, date and BLOB data. The full-text index provides easy implementation for advance querying, phrase querying and stop words elimination support [39].

We have created full-text index on ASHG database fields. Figure 5.2 shows database model of CINDI search database on which we have created full-text index.

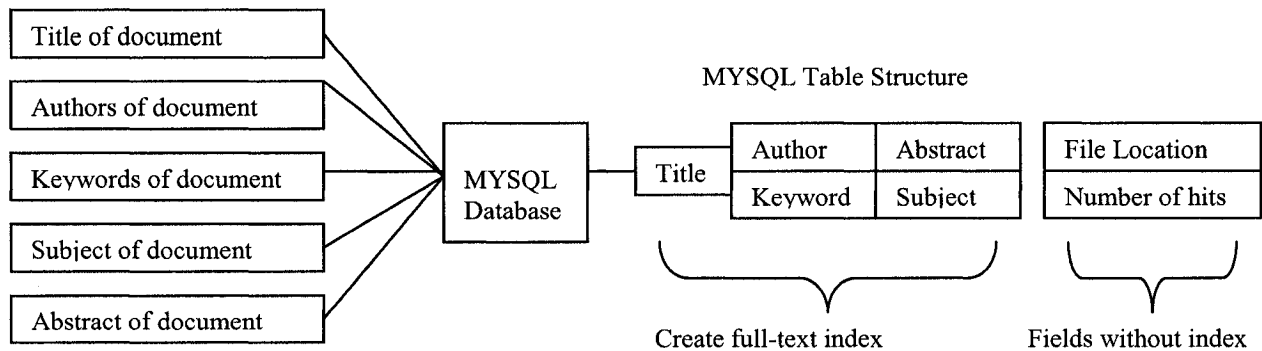


Figure: 5.2 Show database model of the search module

5.5 Graphical User Interface

Existing search system

In older version of CINDI system [2] the user needs to register to search to the repository. The graphical user interface of this search sub-system is shown in Figure 5.3 the registered users can input items such as: title, authors, keyword, and subject, sub-subject, and sub-subject of the resource. Alternatively, users can input only the fields that they know. The system uses three level subject hierarchy pulls down menu for searching by subject.

Resource Search (Simple)

Simple Search
Intermediate Search
Advanced Search
Help

All the types of search listed are "AND" type, you can't base all on "OR" type.

Title: Exact

Author: Exact

Keyword: Exact

(Use a comma to separate for "AND")

Source: General: Please select one of the following subjects
 Sub-Source: Please select one of the following sub-subjects
 Sub-Sub-Source: Please select one of the following sub-sub-subjects

Created Date Period: From: 2000 Month Day To: 2003 July 06

(If the current date by default)

Search Reset Logout

Figure 5.3 showing old search module

The system also had support for advanced search where users can specify conjunctive or disjunctive operations among the search items. The figure 5.4 shows the advanced search module for the older version of CINDI.

Resource Search (Advanced)

Simple Search
Intermediate Search
Advanced Search
Help

Please fill the following fields, and select "AND" or "OR" for the last field.

Title: Exact

Author: Exact And
 Exact And
 Exact
 Add One or More Authors

Keyword: Exact And
 Exact And
 Exact
 Add One or More Keywords

Source: General: Please select one of the following subjects
 Sub-Source: Please select one of the following sub-subjects
 Sub-Sub-Source: Please select one of the following sub-sub-subjects

Select AND/OR

Search Reset Logout

Figure 5.4 Shows advanced search in old CINDI system

Two main search engine components are important to carry out a successful search [18]:

- (I) The user interface, where the user types the query keywords and the search results are shown.
- (II) The search process, which seeks the requested information and orders the results by relevance; and

Since individuals interact with the search tool to set up a search task and explore the results, it is essential that user interfaces be easy to use and accessible to all. Therefore designing a good graphical user interface is as important as developing the backend of the search module.

Common problems encountered in navigating in the results of searches can be summed as follows:

- 5.6.1. Lack of context – The user has access to only a small portion of the target documents i.e. only title and author are listed in search results with no details about the text within the document. A better approach would be to present the user an abstract or part of the introduction of the target documents.
- 5.6.2. Keyboard navigation – Some users prefer to use the keyboard over the use of mouse (i.e. pointing, scrolling, selecting, etc.); they move around the page using keyboard commands, such as Tab key, arrow keys, and so on.
- 5.6.3. Information overload – Portions of the information that is displayed when user makes search. For example rather than displaying information about whole documents it might be better to display only title and some abstract. This will help user to select detailed information when desired.

Taking into consideration the above issues and the interface of previous version we use the following principles in redesigning the interface for CINDI search module:

Easy location: - The objects are placed in specific locations and so that they can easily be noticed by the users. A search textbox on every webpage; this allows user to modify the search criteria from any webpage.

Highlighting the search terms in the result: - The task of the highlighter module is to highlight the search terms. The highlighter deletes stop words from query and only non stop words keywords.

Arranging the results: - The results are arranged in a tabular form according to their rank. Each webpage display thirty results per page with their title and abstract. The limit of number of results can be altered by the user by setting preferences. Whenever user finds some document of interest he can click on title to see the full abstract of the document and the number of hits for it.

Help links [19]: Interface has been assigned a scale of importance so that users can reach most important elements quickly. Help links are provided on each step so that user doesn't feel lost in the middle of a search.

In order to refine older graphical user interface of CINDI search module we have used a graphical layout divided into three sections figure 5.5:

1. Sign In and help: Allows registered users to login in to CINDI system to upload resources or to make annotations. The help link provides the help manual for CINDI search.

2. Navigation Bar: The navigation bar consists of links that for advanced search, submit document and more.
3. Search box: This is the textbox where user enters the search terms.

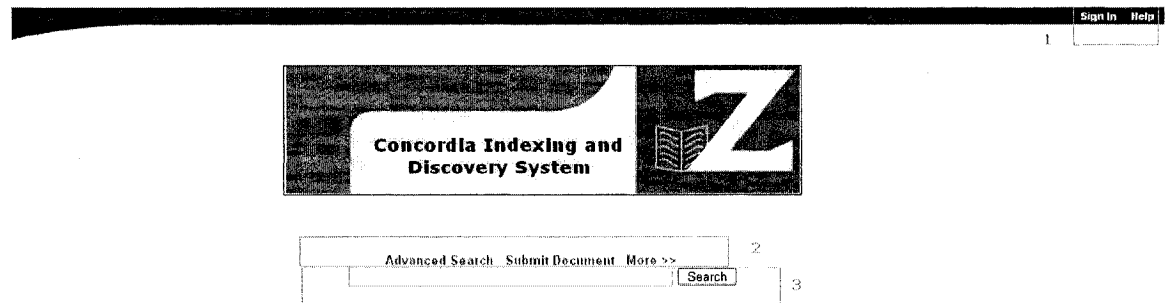


Figure 5.5 Logical sections of modified graphical layout of CINDI search

User inputs search terms in the textbox and submits the query to by pressing return or the search button. The results of the query are displayed in tabular form shown in figure 5.6.

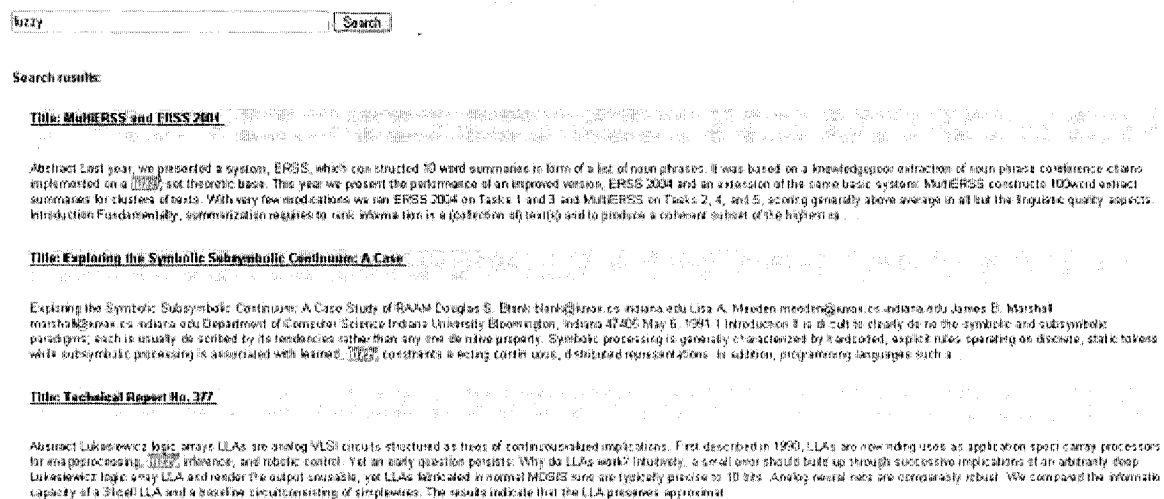


Figure 5.6 Interface of results displayed by CINDI search module

The user gets a brief summary for a document by clicking its title. Figure 5.7 show the sample of the summary displayed when the user clicks on a title. The display show includes the document title and abstract along with other information such as (number of times document has been downloaded by user), Pages (number of pages in the document), comments (number of comments made by other users on the document) and similar documents. User can click on the PDF image to download the document in PDF format.

Overcoming the Majority Barrier in LargeScale Systems		Title	Download Document in PDF
No. of comments added by users		No. of Pages	Pages: 23
Add Comment - 0	Abstract of Document		Number of Hits: 1
<p>Abstract Abstract In asynchronous environments such as the Internet, the majority barrier prevents us from achieving consensus when more than half of the nodes fail. We argue that the majority barrier significantly limits the availability of practical systems such as peertopeer systems. To overcome the barrier realistically, this paper proposes a witness model that probabilistically strengthens the traditional asynchronous model for largescale systems. The model is motivated by the prevalence of fuzzy partitions in today's Internet, namely, reachability is far from transitive. Our assumptions in the model are strictly weaker than those in previous approaches. To show that the model is realistic, we carefully use real Internet measurements to validate....(cont).</p>			
Similar document recommendations Automating the Meta Theory of Deductive Systems -- 56% Modeling Distances in LargeScale Networks -- 56% Locating Data Sources in Large Distributed Systems -- 55% Development of the Domain Name System* -- 53%		Similar documents based on text	

Figure 5.7 shows the information displayed to user when user clicks desired document

In case the user wants to search for some terms appearing in the titles, or within the abstract of the paper. She can make use of CINDI advanced search. Figure 5.8 show the graphical layout of CINDI advanced Search.

CINDI Advanced Search

Containing all of these words

Containing these exact phrases

At least one of the words

Not containing these words

where my search words/phrases occur in

Type one or more phrases, subjects or words separated by comma

☒ Anywhere in the target documents

☐ Return only those target documents where my search words/phrases occur only in the following place(s)

☐ Title

☐ Author

☐ Keyword

☐ Abstract

☐ Subject(Under Implementation)

Search

CINDI Advanced Annotation Search

annotation title and review contains

annotator nickname

Search

Figure 5.8 CINDI Advanced search Interface

When a user searches for the term “database” appearing in title only, then the result would be as shown in Figure 5.9.

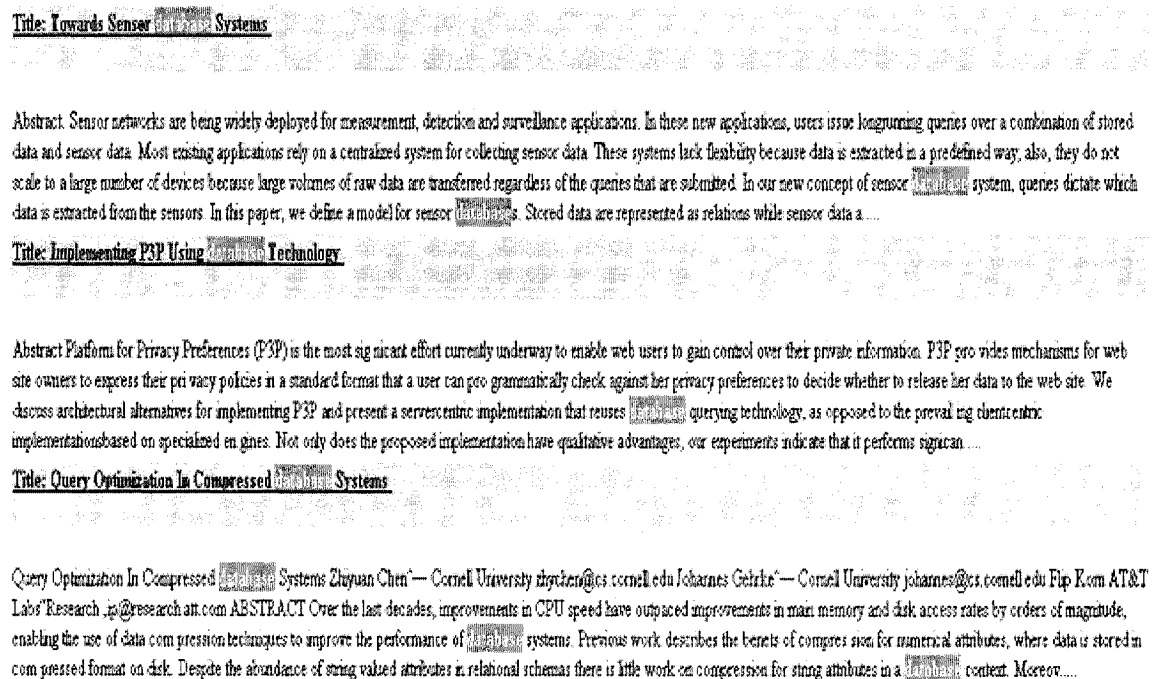


Figure 5.9 CINDI advanced search result for search ‘database’ in Title as mandatory parameter

5.6 Implementing Basic Search

The first query to the CINDI Search module is keyword search, which can be used to return a set of documents matching the query. One of the key advantages of keyword search is its simplicity – users do not have to learn a complex query language, and can issue queries without any prior knowledge about the structure of the underlying data. Since the keyword search query interface is very flexible, queries may not always be precise and can potentially return a large number of query results, especially in large document collections.

Keyword Query Results

We now define the results of keyword search queries for CINDI System. There are two possible semantics for keyword search queries. Under *conjunctive* keyword query semantics, elements that contain *all* of the query keywords are returned. Under *disjunctive* keyword [20] query semantics, elements that contain *at least one* of the query keywords are returned. In CINDI search the conjunctive keyword query are ranked higher than disjunctive keyword query. Hence the result will present documents that contain all of the keywords followed by document that contain some of the keywords. Consider a keyword search query consisting of n keywords:

$Q = \{k_1 \dots k_n\}$. Let $R_0 = \{v \mid v \in NE \wedge \forall k \in Q (\text{contains}^*(v, k))\}$ be the set of elements that directly or indirectly contain all of the query keywords. Here NE is set of columns in database where keywords are to be searched; R_0 is the current record among list of record that is being search, v search term entered by user and k contains the keyword.

Result (Q) thus contains the set of elements that contain at least one occurrence of all of the query keywords, after *excluding* the occurrences of the keywords in sub-elements that already contain all of the query keywords.

CINDI search

The query for the basic search starts when a user inputs some text in the search textbox. The text is broken down into tokens of separate words. Then CINDI search module eliminates stop words string. The remaining tokens are transformed using the porters stemming algorithm [26] to get stemmed tokens. Now the weights are assigned to the tokens, the search terms (i.e. non truncated tokens) given by the user are assigned more weight than the truncated tokens. This query string is passed to CINDI search subsystem that locates relevant documents. Before displaying the results to the user the documents in the results are ranked based on the number of tokens found in each. If the query contains more than one term we perform AND operation $K_1 \text{ AND } K_2$ (K_1 and K_2 are search terms) followed by results $K_1 \text{ OR } K_2$ gets stored in List1. List2 is created by listing records in List1 in ascending order; by number of annotations and number of hits. This List1 is then merged with List2 of documents created by the popularity of the document. The rank value is added to documents in list 1 and the result is displayed in ascending order. The algorithm for keyword search is given below:

Input: Keywords $\{k_1, \dots, k_n\}$

Output: Ranked documents containing keywords $\{k_1, \dots, k_n\}$

{

list <- **Tokenize** (keywords)


```

For each keyword in list do

    If keyword  $k_i$  = stop words

        then delete keyword

    If keyword  $k_i$  != stemmed

        then potters_algorithm(keyword)

    new_list<-(keywords + stemmed keywords)

document_list <- ranking( all keywords > some keywords >stemmed keywords)

document_list<-max_hits_ascending(document_list)

return document_list

}

```

Algorithm CINDI Keywords Search in basic search module

In table 5.1 we compare the steps for a query CINDI search with Citeseer and Google Scholar.

	CINDI	Google Scholar	Citeseer
Automatic Stemming	In CINDI we have used Porter stemmer to implement stemming	The Google scholar provides automatic stemming using Porter stemmer	The Citeseer does not implement any stemming
Implicit AND	The CINDI system makes use of Implicit AND for the keyword based search	This feature is provided by Google scholar	There are no advanced search features provided by Citeseer
Common-Word Exclusion	CINDI Search is using MySQL stop words list to ignore stop words from the query.	Stop words are excluded from the search	Stop words are excluded
Word Limit	CINDI search is using MySQL at the backend. Hence word limit is decided by max_allowed_packet which can be increased or decreased	The Google has search limit of 32 characters in the query entered by the user	There is no limit to the search query entered by user
Case Insensitivity	CINDI search is case-insensitive	search is case-insensitive	search is case-insensitive
Ignoring Punctuation	CINDI ignores punctuation, special characters and words whose length is shorter than 3 characters. For words shorter than 3 characters user needs to	ignores punctuation , special characters and words, but no minimum length	ignores punctuation , special characters and words, words whose length is shorter than 2 characters

	specify * in the end i.e. acm*, ibm*		
Spell Check and recommendations	This feature is not yet implemented in CINDI search.	Spell check is available	No spell check implemented
Citations	This feature is not yet implemented in CINDI search.	This is implemented in Google Scholar	This is the heart of CiteSeer. The ranking is based on citations
Page Ranking	CINDI search page ranking is based on number of hits the document gets and number of annotations written for the document	Page ranking is based on Citations	Page ranking is based on citations
Advance Search	Advance search is provided to search: with all of the words, with the exact phrase, with at least one of the words, without the words within title, abstract, keywords and author. Subject will be provided shortly.	Advance search is provided to search: with all of the words, with the exact phrase, with at least one of the words, without the words within title, anywhere. Search for publication, subject and author is provided	No advance search provided

Table 5.1 Comparison between CINDI search with Google Scholar and CiteSeer.

5.7 Advanced Search

The CINDI advanced search allows users to search using Boolean operators. The most common Boolean operators are AND (you're looking for all terms), OR (you're looking for at least one of the terms), and NOT (you're excluding a term). Here is the syntax that is followed by CINDI Search module. These operators can be used in basic search module and for users who prefer graphical interface we have advanced search interface.

- + A leading plus sign indicates that this word must be present in every row returned.
- A leading minus sign indicates that this word must not be present in any row returned.

> < These two operators are used to change a word's contribution to the relevance value that is assigned to a row. The > operator increases the contribution and the < operator decreases it.

() Parentheses are used to group words. Parenthesized groups can be nested.

~ A leading tilde acts as a negation operator, causing the word's contribution to the row relevance to be negative.

" A phrase that is enclosed within double quote (") characters matches only rows that contain the phrase literally, as it was typed.

* A phrase that can have any ending after * is written (i.e. app* will result apples, apple, applause)

Search Examples

database design Find rows that contain at least one of the two words.

+database +design Find rows that contain both words.

+database design Find rows that contain the word “database”, but rank rows higher if they also contain “design”.

+database –images Find rows that contain the word “database” but not “images”.

+database +(>design <images) Find rows that contain the words “database” and “design”, or “database” and “images” (in any order), but rank “database design” higher than “database

images”.

data*	Find rows that contain words such as “database” or “data
“database design”	Find rows that contain the exact phrase “database design”

CINDI users can easily create powerful queries by entering words into a combination of the text boxes in the advanced search interface.

all of these words: Documents must contain all of the words.

this exact phrase: Documents must contain these exact words in the order in which user have typed.

any of these words: Documents must contain at least one of the words.

none of these words: Documents that contain these words will be omitted from the result.

Besides CINDI advanced search also enables users to search the desired keywords within title, abstract and introduction. When user selects particular keyword within title, search subsystem checks only titles of the documents are found: this information is provided by ASHG. Below is the algorithm used in CINDI exact phrase search.

```
[1] For each word in the search phrase {  
    [2] for each subsequent word in the search phrase {  
        [3] is there a word that continues the match  
            {  
                Starting at current pointer, scan forward for a  
                match in the same file;  
                If no match found, go back to [1]  
                Else, if we're looking at the last word  
                    {  
                        Accept the match  
                    }  
            }  
        else {
```

Algorithm: Phrase matching

Example of Advanced Search

When a user search using CINDI advanced search for documents that contain images but not databases in abstract of documents. First of all the stop words in the query are deleted from the search string and then query will be locate documents that contain the keyword 'image' those documents from this set that contain the keyword 'database' are deleted from the set. Figure 5.10 shows the final result of query provided by the CINDI Search module.


Title: IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. XX, NO. Y, MONTH 1999 100

Abstract This article presents a family of techniques that we call *congealing* for modeling image classes from data. The idea is to start with a set of models, *congeal* them together, and make them appear as similar as possible by removing variability along known axes of variation. This technique can be used to eliminate nuisance variables such as affine deformations from handwritten digits or unwanted bias from magnetic resonance images. In addition to separating and modelling the latent *structure*, i.e. the *image* without the nuisance variables, we can model the nuisance variables themselves, leading to factorized generative image models. When nuisance variable distributions are shared between classes, one can share the knowledge learned in one task with another.

Title: Learning HyperFeatures for

Abstract We address the problem of identifying specific instances of a class (cars) from a set of [images](#) all belonging to that class. Although we cannot build a model for any particular instance (as we may be provided with only one training example of it), we can use information extracted from observing other members of the class. We pose this task as a learning problem, in which the learner is given image pairs, labeled as matching or not, and must discover which image features are most consistent for matching instances and discriminative for mismatches. We explore a patch based representation, where we model the distributions of similarity measure means denoted on the patches. Finally, we describe an algorithm that selects the most sale

Title: Joint Nonparametric Alignment for Analyzing Spatial Gene Expression Patterns

Abstract To compare spatial patterns of gene expression, one must analyze a large number of as current methods are only able to measure a small number of genes at a time. Binding of corresponding tissues into alignment is a critical first step in making a meaningful comparative analysis of these spatial patterns. Significant image noise and variability in the shapes make it hard to pick a canonical shape model. In this paper, we address these problems by combining segmentation and unsupervised shape learning algorithms. We first segment  to acquire structures of interest, then jointly align the shapes of these acquired structures using an unsupervised nonparametric maximum likelihood algorithm along the lines of congeal.

Title: Probability Models for High Dynamic Range Imaging

Abstract Methods for expanding the dynamic range of digital color photographs by combining multiple images taken at different exposures have recently received a lot of attention. Current techniques assume that the photometric transfer function of a given camera is the same (modulo an overall exposure change) for all the input images. Unfortunately, this is rarely the case with today's cameras, which may perform complex, non-linear color and intensity transforms on each picture. In this paper, we show how the use of probability models for the imaging system and weak prior models for the response functions enable us to estimate a different function for each image using only pixel intensity values. Our approach also allows us to characterize the uncertainty in the estimated functions.

Title: Automatic Image Annotation and Retrieval using

Automatic Image Annotation and Retrieval using Cross-Media Reference Models | Jeon, Y. Lavenko and P. Mannanatha Center for Intelligent Information Retrieval Computer Science Department University of Massachusetts Amherst, MA 01003 [jeon.lavenko, mannatha@cs.umass.edu] Abstract Libraries have traditionally used manual image annotation for indexing and then later retrieving their image collections. However, manual image annotation is an expensive and labor intensive procedure and hence there has been great interest in coming up with automatic ways to retrieve images based on content. Here, we propose an automatic approach to annotating and retrieving images based on a training set of images. We assume that regions in an image can be described

Figure 5.10 Show result of query with “images” but not “databases” in document’s abstract

5.8 Similar documents

Finding similar documents in natural language document collections is a difficult task that requires general and domain-specific world knowledge, deep analysis of the documents, and inference. However, a large portion of the pairs of similar documents can be identified by simpler, purely word-based methods [27, 28]. In CINDI we have implemented this approach. In the preliminary processing step, we count the number of tokens in the title and the abstract of the document and remove the ‘stopwords’ from the token. At run-time, a query consisting of keywords is submitted to the system. The query is immediately converted to tokens (stopwords removed), expanded by stemming and then compared with the tokens extracted of each document. The steps for textual analysis of data are processed in two steps.

Step1: We select the title and abstract of the document selected by the user by clicking on the title of the document. The title and abstract of this document is stored in two strings S_T and S_A .

Step2: We make use of full-text index and match these S_T and S_A .

```
convert_lower( $S_T, S_A$ )
```

```
delete_stopwords( $S_T, S_A$ )
```

```
list_title->search_text_title ( $S_T, S_{TD}, match\_value$ )
```

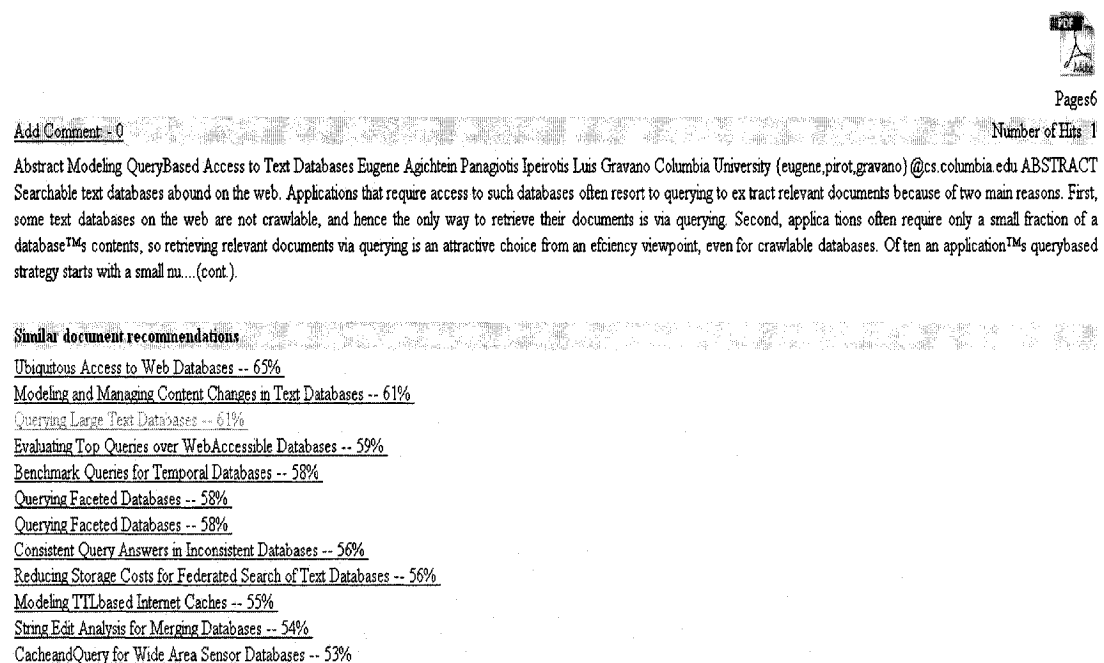
```
list_abstract->search_text_abstract ( $S_A, S_{AD}, match\_value$ )
```

S_{AD} and S_{TD} are ASHG database title and abstract.

Step3: Sort the list_title and list_abstract increasing order of their match_value. Each document receives an increment in their match_value if same token exist in S_T and S_{TD} or S_A and S_{AD} . The match_value is then converted in percentage (matched tokens S_{TD} or S_{AD} / total number of tokens S_A or S_T * 100). Now we find final_list = list_title \wedge list_abstract and the final_list is sorted in ascending order. This information is listed to user.

The figure 5.11 shows the interface and list of similar documents by title and abstract generated by CINDI search subsystem.

Modeling QueryBased Access to Text Databases



PDF
Pages 6

Add Comment: 0 Number of Hits: 1

Abstract Modeling QueryBased Access to Text Databases Eugene Agchtein Panagiotis Ipeirotis Luis Gravano Columbia University (eugene.piot,gravano)@cs.columbia.edu ABSTRACT Searchable text databases abound on the web. Applications that require access to such databases often resort to querying to extract relevant documents because of two main reasons. First, some text databases on the web are not crawlable, and hence the only way to retrieve their documents is via querying. Second, applications often require only a small fraction of a database's contents, so retrieving relevant documents via querying is an attractive choice from an efficiency viewpoint, even for crawlable databases. Often an application's querybased strategy starts with a small nu... (cont.)

Similar document recommendations

- [Ubiquitous Access to Web Databases -- 65%](#)
- [Modeling and Managing Content Changes in Text Databases -- 61%](#)
- [Querying Large Text Databases -- 61%](#)
- [Evaluating Top Queries over Web Accessible Databases -- 59%](#)
- [Benchmark Queries for Temporal Databases -- 58%](#)
- [Querying Faceted Databases -- 58%](#)
- [Querying Faceted Databases -- 58%](#)
- [Consistent Query Answers in Inconsistent Databases -- 56%](#)
- [Reducing Storage Costs for Federated Search of Text Databases -- 56%](#)
- [Modeling TTLbased Internet Caches -- 55%](#)
- [String Edit Analysis for Merging Databases -- 54%](#)
- [Cache and Query for Wide Area Sensor Databases -- 53%](#)

Figure 5.11 List of similar documents generated by CINDI Subsystem

Chapter 6

Annotation

6.1 Overview of Annotation

Annotation is a note that describes, explains, or evaluates a particular document. Annotations can be in form of comments, notes, explanations, questions, references, examples, advice, correction or any other type of external remark that can be attached to a document or a selected part of the document. As they are external, it is possible to annotate any document independently, without needing to edit that document. Annotations can be stored locally or in one or more servers. When a document is browsed, system queries each of these servers, requesting the annotations related to that document.

Annotation System Characteristics

Gramlich [29] identifies a number of issues related to WWW based public annotation systems. These issues were studied while developing the CINDI Annotation system and are discussed below:

1.Application

Annotation Systems are designed to cater to some particular type of application or systems. In our case it will be catering to the users who will be adding their comments to the CINDI documents.

2.Architecture

Annotation systems are developed using client/server architecture. We are developing it using same technique where our client side development is done in HTML, JavaScript and server side is in PHP and MYSQL.

3.Authentication

Every system needs to have some sort of authentication. In annotation systems annotators should be authenticated before they can add any sort of comments or edit existing comments. For CINDI Annotation System users need to be registered and logged in order to post any sort of comments to a document and only the system administrator after valid login can delete any comments.

4.Authoring

What is the support that system is going to be providing? How users add annotations to documents? Is the annotation is plain text or HTML.

5.Granularity

Course grained systems only allow annotations to the whole document. Stick them all on the end of the document.

Fine grained systems allow annotations to be added anywhere in the document.

For our CINDI Annotation system we are using partly course grained systems i.e. users can add annotation separate from the documents stored in annotation database.

6.Moderation

The control of the system needs to define while developing the annotation system. How is control of the system divided between the contributor, administrator and the users?

7.Notification

Some annotation systems notify contributors and users who have commented on documents of the changes via email. We are not implementing this feature in our system.

8.Visibility

Visibility refers to access of the system to normal users. Are comments visible to users who are not registered or not. Some support allowed only to users with particular rights. For CINDI Annotation systems visibility will be allowed to anyone who browse the webpage, but posting will be limited to registered users.

9.Storage

Annotation systems store the annotations along side the original documents at the server. The main exception to this is the use of referential annotations (user supplies URL of annotation rather than annotation itself). We are storing our annotation information on server separate from the documents but linked to them.

6.3 Architecture CINDI Annotation System

In this section, we describe the basic system structure, the user view of the annotation system and the current interface design CINDI. The annotation subsystem works for multiple independent authors and users of the CINDI system to add annotations for the documents in the CINDI repository. These annotations are stored in annotation database which is linked with the ASHG Search subsystem database. Users interact with a browser to retrieve documents from ASHG Search database, depending on the context

searched by the user, the subsystem decides which annotations to retrieve and display the search results. The figure 6.1 shows the architecture of CINDI Annotation Subsystem.

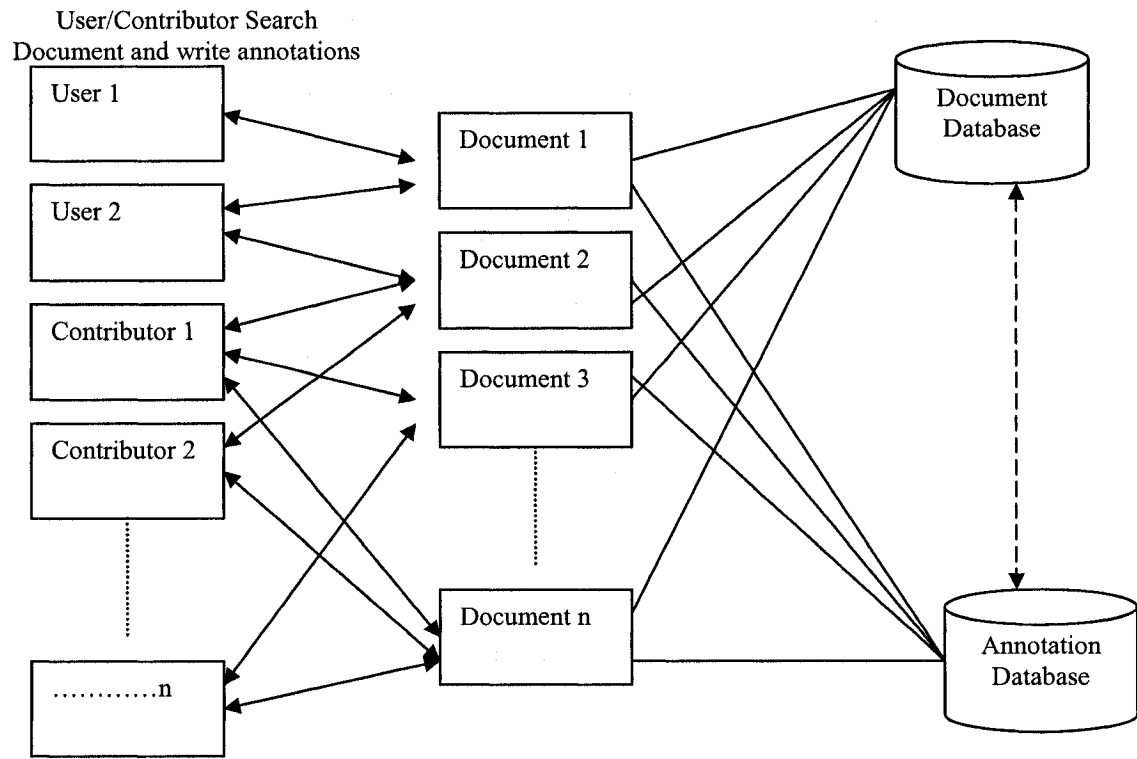


Figure 6.1. Architecture of the CINDI Annotation Subsystem

The CINDI Annotation Subsystem consists of registered users. Annotations can be made by users or contributors; for the annotation subsystem we refer to these users as annotators. Whenever an annotator adds an annotation to a document his nickname is associated with the annotation. This information is used by Annotation search module to search annotations by annotator nickname.

Whenever a user performs a search he gets a list of documents. Annotator can select any of these documents to annotate. These annotations are filtered for impolite expressions

and replaced them with ***. Below is the information that we record for each annotation.

Annotation Text: Annotation text is the description of the annotation provided by annotator.

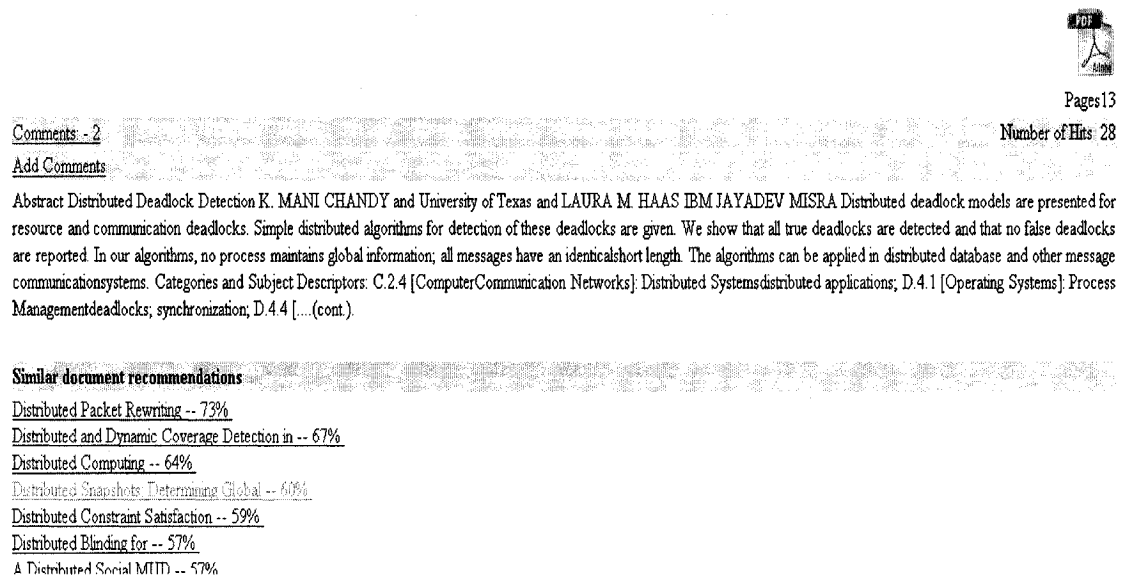
Annotation Author: We use the nickname as the author of the annotation.

Target Document: This is information about document for which annotation is made.

6.4 User Interface of Annotation Subsystem

User can browse annotations associated with documents fetched by CINDI Search subsystem. The browsing of annotations does not require login to the CINDI system. When user selects a document for more information by clicking on the title of the document, they are redirected to the page similar to page as shown in figure 6.2.

Distributed Deadlock Detection



The screenshot displays a document page with the title "Distributed Deadlock Detection". In the top right corner, there is a PDF icon, the text "Pages 13", and "Number of Hits: 28". Below the title, there is a section for annotations. It shows "Comments: - 2" and a link "Add Comments". The main text of the document is visible, starting with "Abstract Distributed Deadlock Detection K. MANI CHANDY and University of Texas and LAURA M. HAAS IBM JAYADEV MISRA". Below the main text, there is a section titled "Similar document recommendations" which lists several documents with their respective similarity percentages: "Distributed Packet Rewriting -- 73%", "Distributed and Dynamic Coverage Detection in -- 67%", "Distributed Computing -- 64%", "Distributed Snapshots: Determining Global -- 60%", "Distributed Constraint Satisfaction -- 59%", "Distributed Blinding for -- 57%", and "A Distributed Social MTD -- 57%".

Figure 6.2. Displaying current document has two annotations and a link to add new annotations.

To add an annotation to a document annotator needs to click on the 'Add Comment' link. The system finds out whether the annotator is logged in the CINDI Registration subsystem or not. If the user is not logged in he is redirected to login screen, to provide username and password. The unregistered user is required to register before she is allowed to make an annotation. Figure 6.3 shows interface to make annotations.

Annotations:

Description Logic Systems with Concrete cindi

We present a first treatment dealing with the semantics of visual spatial query languages for geographic information systems using a suitable description logic. This decidable space logic is described and its usefulness for geographic information systems is exemplified.

Description Logic Systems with Concrete cindi

Introduction For accessing spatial databases or geographic information systems (GIS), different query specification techniques have been proposed. For instance, the visual spatial query system VISCO developed in our group can be used to query a spatial database (GIS) in a visual way. In contrast to conventional textual query systems the us

Annotation:

Submit
Reset

Figure 6.3 Interface for making Annotations to CINDI system

Figure 6.4 shows the steps that user follows to make annotation

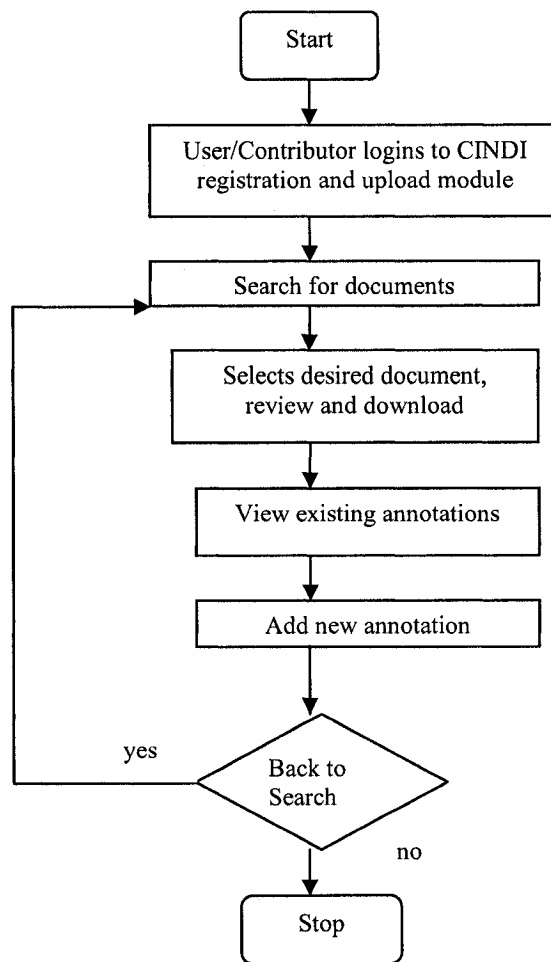


Figure 6.4 CINDI annotation flow of information

6.5 Annotation Search

Annotation search module provides users search features for annotations made for documents in CINDI repository. Basic Annotation search is fundamentally based on keyword search, here users submit keywords to the search engine and a ranked list of matching documents with the words in the annotation is returned. The technique followed for development of keyword search is similar to the one discussed in search

module [Chapter 4]. It is implemented in CINDI search subsystem as a part of advance search module.

When user types some keywords, these keywords are searched in the annotation database and the results are sent to the browser. The figure 6.5 shows the results when user types database as a search string. The result includes the title of the target document, annotator nickname and annotation text. The user can click on the document title to view the document.

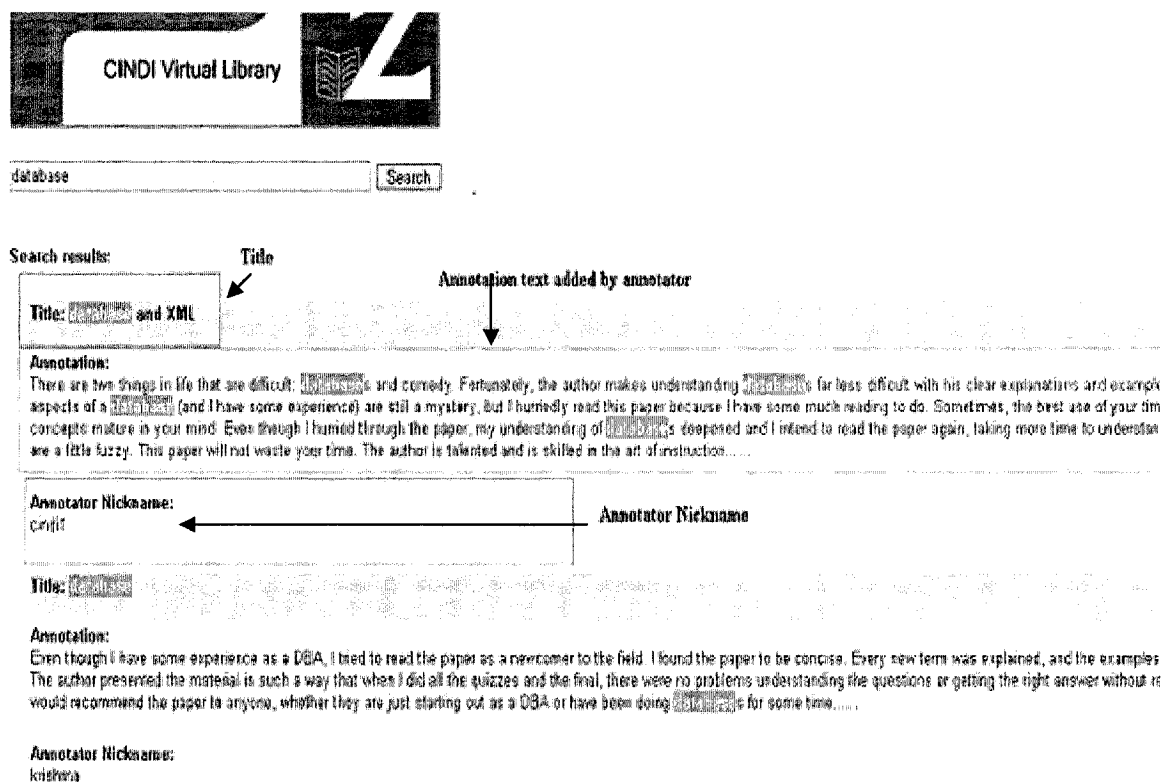


Figure 6.5 Annotation Search results displaying title of document and annotations

Chapter 7

Administration of CINDI Registration and Annotation Subsystem

Administrative Panel

The administrative subsystem provides the administrator of CINDI system to manage users and contributors of CINDI system, accept and delete annotations made by users, send mass email to group of CINDI users and view resources uploaded by the contributors. The figure 7.1 shows the interface of the CINDI Administrative Panel for the subsystem.

CINDI Administrative Panel

CONTROL PANEL

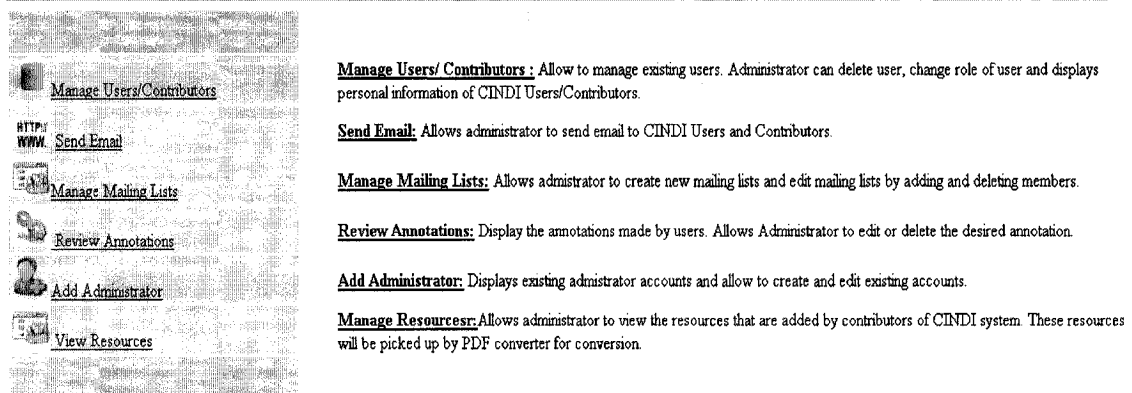


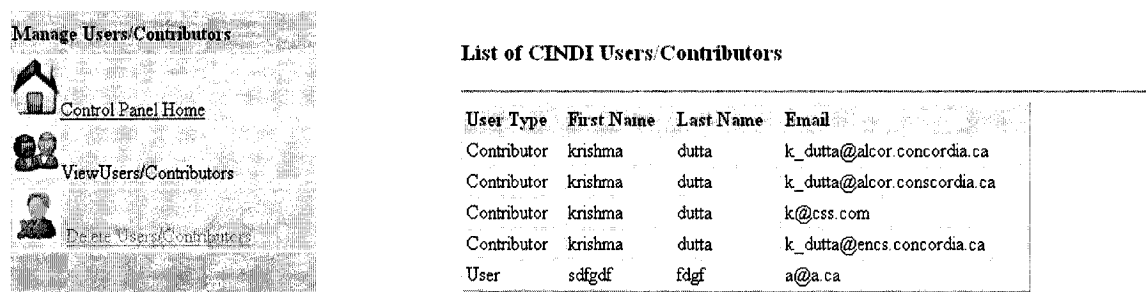
Figure 7.1 CINDI Administrative Panel and its features

The administrative system consists of following modules:

Manage Users/ Contributors: Allow features to manage existing users. Administrator can delete user, change role of user and display their personal information of CINDI Users/Contributors. Figure 7.2 shows manage users/contributor interface.

CINDI Administrative Panel

CONTROL PANEL



User Type	First Name	Last Name	Email
Contributor	krishna	dutta	k_dutta@alcor.concordia.ca
Contributor	krishna	dutta	k_dutta@alcor.concordia.ca
Contributor	krishna	dutta	k@css.com
Contributor	krishna	dutta	k_dutta@encs.concordia.ca
User	sdfgdf	fdgf	a@a.ca

Figure 7.2 Manage user module Interface

Manage Mailing Lists: Allows administrator to create new mailing lists and modify mailing lists by adding and deleting members. These mailing lists can be used by administrator to send email to the users of CINDI system. Figure 7.12 shows interface to create mailing list using CINDI administrative panel.

CINDI Administrative Panel

CONTROL PANEL

	Nickname	Firstname	Lastname	Email
<input type="checkbox"/>	krishna	krishna	dutta	k_dutta@alcor.concordia.ca
<input type="checkbox"/>	krishna007	krishna	dutta	k_dutta@alcor.concordia.ca
<input type="checkbox"/>	cindi	krishna	dutta	k@css.com
<input type="checkbox"/>	cindi007	sdgdf	fdgf	a@a.ca
<input type="checkbox"/>	cindi1	krishna	dutta	k_dutta@encs.concordia.ca

Figure 7.3 Interface to create new mailing lists

Send Email: Allows administrator to send email to CINDI Users, Contributors and mailing lists created by manage mailing list module. The figure 7.4 shows interface to send email to contributors of CINDI system.

CINDI Administrative Panel

CONTROL PANEL

To:
k_dutta@encs.concordia.ca,k@css.com,k_dutta@alcor.concordia.ca,k_dutta@alcor.concordia.ca

Subject:

Message:

Send Mail

Figure 7.4 Administrative Interface for send email

Manage Resources: Allows administrator to view the resources that are added by contributors of CINDI system. Figure 7.5 shows interface of manage resources interface.

CINDI Administrative Panel

CONTROL PANEL

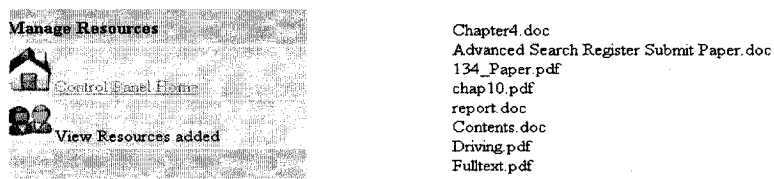



Figure 7.5 Administrative Interface for manage resources


Review Annotations: Display the annotations made by users. Allows Administrator to edit or delete the desired annotation. Add Administrator: Displays existing administrator accounts and allow creating and editing existing accounts. Figure 7.6 shows interface to view and delete annotations.


CINDI Administrative Panel

CONTROL PANEL

Manage Annotations

 [Control Panel Home](#)

 [View/Delete Annotations](#)

 [Edit Annotations](#)

List of Annotations

	Annotation Title	Annotation	User
delete	Deadlock	This is good paper	cindi1
delete	Deadlocks	Hi I really liked author techniques :). Great work	cindi1
		There are two things in life that are difficult: databases and comedy. Fortunately, the author makes understanding databases far less difficult with his clear explanations and examples. I confess that some aspects of a database (and I have some experience) are still a mystery, but I hurriedly read this paper because I have some much reading to do.	
delete	Database and XML	Sometimes, the best use of your time is to slow down and let concepts mature in your mind. Even though I hurried through the paper, my understanding of databases deepened and I intend to read the paper again, taking more time to understand some of the concepts that are a little fuzzy. This paper will not waste your time. The author is talented and is skilled in the art of instruction.	cindi1
delete	Database	Even though I have some experience as a DBA, I tried to read the paper as a newcomer to the field. I found the paper to be concise. Every new term was explained, and the examples were more than adequate. The author presented the material in such a way that when I did all the quizzes and the final, there were no problems understanding the questions or getting the right answer without referring back to the text. I would recommend the paper to anyone, whether they are just starting out as a DBA or have been doing databases for some time.	krishma

Figure 7.6 Interface to view and delete annotations

Search: Allows administrator of the CINDI system to search users and contributors and view their detailed information. The search module allows searching for users by their name, email, nickname and other information. The administrator can also search users for particular region by searching users via country or organization. Figure 7.7 shows the interface for search by name, email, first name or last name.

CINDI Administrative Panel

CONTROL PANEL

Nickname	Email
krishma	k_dutta@alcor.concordia.ca
krishma007	k_dutta@alcor.concordia.ca
cindi	k@css.com
cindi1	k_dutta@encs.concordia.ca

Figure 7.7 Search interface to search users by name, email and other information

When user clicks on more info link listed in figure 7.7 he will be redirected to detailed information about the that user. This is shown in figure 7.8

CINDI Administrative Panel

CONTROL PANEL

Email	Firstname	Lastname	Nickname	User Type	Country	Organization
k_dutta@encs.concordia.ca	krishma	dutta	cindi1	Contributor	Canada	Concordia University

Figure 7.8 Search interface when administrator asks for more information

Here we see that following information is listed: email of the user, first name, last name, nickname, user type, country and organization.

Chapter 8

Integration

8.1 Overview of Integration

This chapter describes the integration of a number of subsystems into a unified system. There are four levels of integration User Interface Integration, Data Integration, Process Integration and Services Integration, for integration subsystems of CINDI we are addressing to data integration. Data integration occurs at the database and data source level within an organization. The integration is achieved by migrating data from one data source to another. The chapter provides an insight of the subsystems; the interface shared by the various subsystems, the working of each subsystem and the integrated CINDI system.

CINDI Robot Database

CINDI Robot creates and operates on 16 MYSQL database tables. These database tables contain the information about seeds, urls of documents downloaded, history, crawler information and more. The table that will be used in integrating other subsystems of CINDI with robot is DOWNLOAD_STATUS. This table consists of information as stated below in figure 8.1

Field	Type	Null	Key	Default	Extra
ID	bigint(20) unsigned		PRI	NULL	auto_increment
prefix_url	blob	YES		NULL	
orig_file_name	varchar(200)	YES		NULL	
file_name	varchar(200)	YES		NULL	
temp_location	varchar(100)	YES		NULL	
final_location	varchar(100)	YES		NULL	
ddate	date	YES		NULL	
size	int(10)	YES		NULL	
file_type	varchar(20)	YES		NULL	
level	int(10)	YES		NULL	
pdf_flag	smallint(1)	YES		NULL	
ashg_flag	smallint(1)	YES		NULL	
filter_flag	smallint(1)	YES		NULL	
is_diff_format	smallint(1)	YES		NULL	
is_renamed	smallint(1)	YES		NULL	
parentID	bigint(20) unsigned	YES		NULL	
num_ref_by	int(100)	YES		NULL	

Figure 8.1 Table schema showing fields from DOWNLOAD_STATUS

In above information prefix_url is url of the website from where the file has been downloaded, org_file_name is the filename of file when it got downloaded, file_name is the filename renamed by CINDI robot, temp_location is location address for files if they are non pdf, final_location is location of files if they are PDF, ddate, size, file_type is the date of download, size of file and type of file. The attributes pdf_flag, ashg_flag, filter_flag is used by CINDI Converter, Automatic Semantic Header Generator and Document Filter subsystems. Attribute is_diff_format gives information about if the file is downloaded file in different format, if the file is rename is_renamed is set, parentID is the ID of the webpage and num_ref_by is number of times the document is referred in other websites. We have used this information while integrating CINDI Robot with File Conversion and Document Filtering System.

File Conversion Storage

The file conversion subsystem enters information listed in figure 8.2 after converting files into PDF. We have used this information while integrating File Conversion subsystem

with Document Filtering subsystem and Automatic Semantic Header Generation subsystem.

Field	Type	Null	Key	Default	Extra
ID	int(50)		PRI	NULL	auto_increment
docID	bigint(20) unsigned		MUL	0	
filename	varchar(100)	YES		NULL	
convert_date	date	YES		NULL	
location_pdf	varchar(100)	YES		NULL	

Figure 8.2 Schema of information stored about PDF converted document

8.2 Data Integration of CINDI

In integrating CINDI system we have used the bottom-up approach, therefore data integration of lowest level modules were performed at first level. For example the FCS and DFS subsystems were integrated into a File Conversion Subsystem.

8.2.1. Data integration of File Conversion Module and Document Filtering Subsystem:

The File Conversion subsystem is system that converts non PDF documents TXT, PS, HTML, DOC, LATEX, TEX, RTF and XML into PDF documents. In earlier system this was performed on windows system using Omnisformat and PDF995. During module testing it was found that Omnisformat doesn't convert RTF, TEX and LATEX into PDF format as these extensions are not supported by windows operating system. Modifications were made and the new File conversion module was ported to the Linux platform. Since DFS was already implemented for Linux. The ported File Conversion module was integrated with DFS for the new File Conversion Module and Document Filtering Subsystem. The File conversion module picks up documents from robot database and transfers it to its own database. There are two types of files PDF and non

PDF; one of PDF the files get transferred to final PDF directory. The non PDF documents are transferred to non PDF directory for conversion. The conversion is performed using combination of openoffice PDF API, ps2pdf, text2pdf and pdflatex. After conversion is done these files are transferred to Final PDF directory and CONVERT_PDF table is populated with the information about filename, location and conversion date. After this converter updates pdf flag in DOWNLOAD_STAUS table that verifies corresponding file is now PDF. The final PDF directory is further passed to filter directory where filter picks up these files and extracts corresponding information from CONVERT_PDF table. This procedure is explained underneath figure 8.3

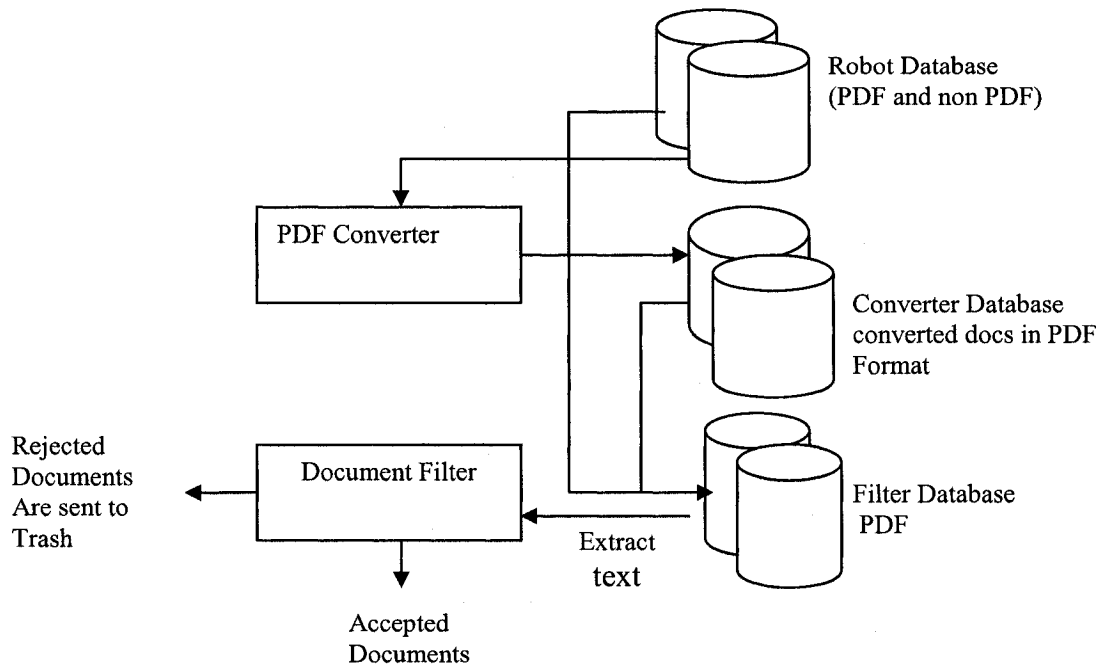


Figure 8.3 Integration of Document Filter and File Conversion module

The DFS extracts text out these documents and verifies if the document is academic. If the document is accepted filter flag in DOWNLOAD_STATUS table is set to 1 else it is set to 2 i.e. rejected documents.

8.2.2. *Data integration of CINDI Robot and Gleaning Subsystem:* CINDI Robot and Gleaning system data integration was the next step as they share same database tables which are further used by ASHG and ASHG Search subsystem. The data integration took place in two steps, first the data from the Document filtering and Document Conversion subsystem was integrated together and then this information was integrated with the data from CINDI robot. The figure 8.4 shows the data integration process between both the subsystems.

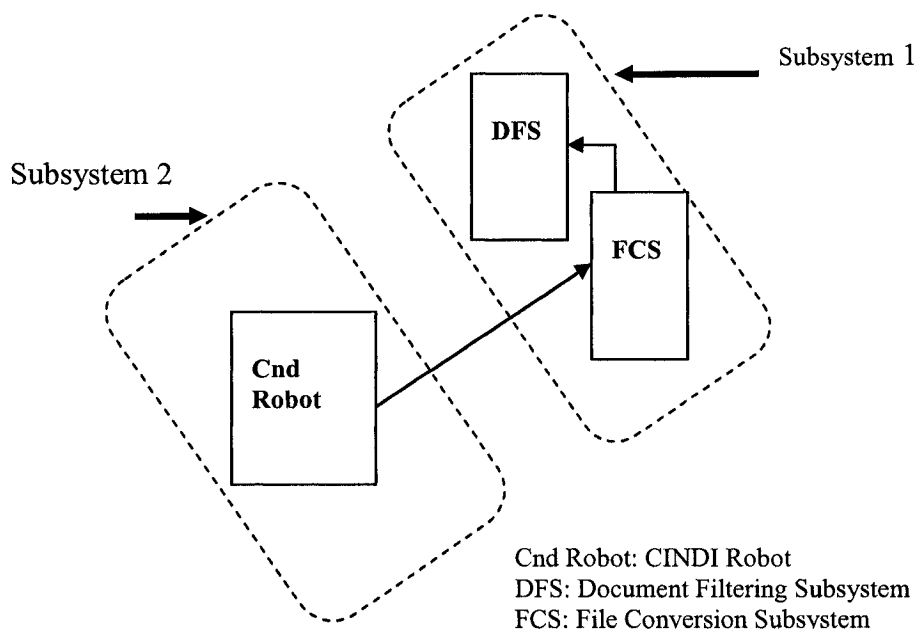


Figure 8.4 Data integration of two subsystems of CINDI

Here we can see that as we have made use of bottom up data integration for Document Filtering and File Conversion Subsystem and then we moved up in hierarchy integrating CINDI robot to the system. The database table shared by CINDI Gleaning subsystem and CndRobot is DOWNLOAD_STAUS. DOWNLOAD_STATUS table is create by the CndRobot, it basically consist of all the information required by the Document Conversion module to download the documents from robot repository. Figure 8.2 shows the schema of the fields present in the table. When CndRobot downloads a document it

adds corresponding description in this table hence data integration is achieved by creating a watch to the table. Whenever a new entry is added to table, Conversion subsystem downloads the document and further passes it to DFS.

8.2.3. *Data integration of ASHG and Document Filtering Subsystem:* The Automatic Semantic Header Generator receives its input data from Document Filter Subsystem. When DFS filters out documents that are research papers, these documents are picked up by ASHG to generate semantics. Semantics include locating information about document's Title, Author, Keywords, Abstract, Introduction and Subject hierarchy shown in figure 8.5.

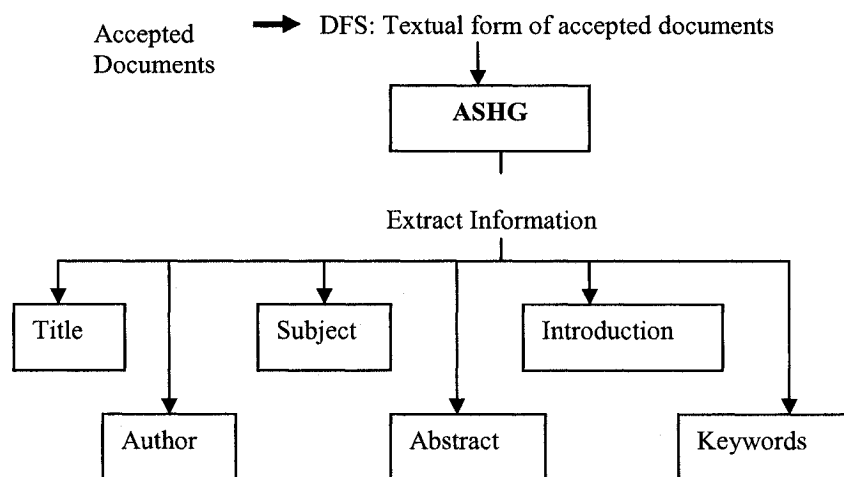


Figure 8.5 Integration of Document Filtering Subsystem and ASHG

The figure 8.5 shows that textual information from Document Filtering Subsystem is required by ASHG which is further broken down into semantics that will be used by ASHG Search Subsystem. When Document Filtering subsystem accepts a paper it sets filter flag to 1 in DONWLOAD_STATUS table. This information is used by ASHG along with the information CONVERT_PDF about the location of the file.

8.2.4. Data integration of ASHG and ConfSys Subsystem:

ConfSys manages academic documents submitted by various authors during conference. The documents are uploaded to ConfSys with associative information regarding the document. The author inputs title, authors, abstract, subject and keywords that describes the document. When ConfSys transfers the documents from ConfSys database to ASHG it sends the information inputted by the author in a XML file. This file is picked up by ASHG and is inserted into Document table, figure 8.6 shows details about the table.

Field	Type	Null	Key	Default	Extra
title	varchar(256)	NO			
author	varchar(256)	NO			
abstract	varchar(512)	NO			
keyword	varchar(256)	NO			
subject	varchar(512)	NO			
filename	varchar(128)	NO			
directory	varchar(128)	NO			
publisher	varchar(64)	YES		NULL	
numOfPage	smallint(6)	NO		1	
numOfHit	smallint(6)	NO		1	
dId	int(10)	NO	PRI	NULL	auto_increment

Figure 8.6 ASHG inputs information into table Document in MYSQL database

In the figure 8.6 structure of table consists of title of the document, author of the document, abstract of the document, subject of the document, name of the document file, directory in which document is located, publisher if available, number of pages in document, number of times document gets downloaded and unique id for each document.

8.2.5. Data integration of ASHG and ASHG Search Subsystem: As explained above ASHG extracts information from the documents downloaded by CINDI robot and ConfSys. This information is added to database which is used by the CINDI Search Subsystem. CINDI Search Subsystem performs search on various criteria so, it is

important to extract this information to fasten up the seeking time. The data integration of ASHG and Search Subsystem was achieved in one phase integration. The Figure 8.7 shows the information that is used by CINDI Search Subsystem for integrating the two systems together.

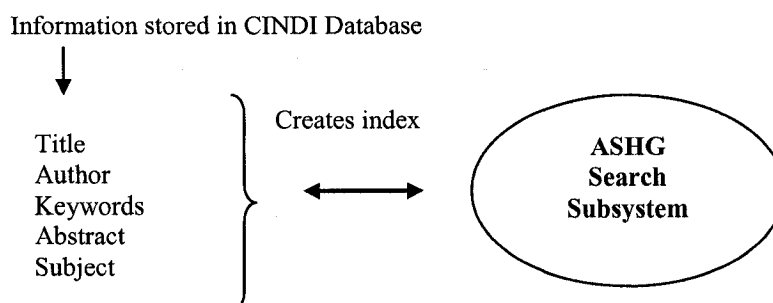


Figure 8.7 Integration of ASHG and Search Subsystem

CINDI Search subsystem creates a full-text index on the information provided by the ASHG and use the index to locate information in the document. The keywords and authors are separated by comma (,). The search module explodes the string whenever it get comma in either of these columns.

8.2.4. ASHG Search Subsystem, CINDI registration and Upload Subsystem and Annotation Subsystem: CINDI ASHG Search subsystem is parent module of CINDI Registration and Upload and Annotation subsystem. The search module allows users of CINDI system to search for documents in CINDI repository. The users can make annotations on the documents that they download, but to make annotation they need to register with CINDI Registration and Upload subsystem. The users can be contributor, users with this role are allowed to upload documents and make annotations or users who are allowed to make annotations to the Search subsystem documents. The information

about the registered users in CINDI system is shared by these three subsystems. While integrating the three subsystems took place in two steps: First ASHG Search subsystem was integrated with the CINDI Registration and Upload module and then this integrated system was integrated with CINDI Annotation subsystem. CINDI annotation system is integrated with search module and registration module by storing the document information and user who made annotation for every annotation made.

Integrated Architecture of CINDI system

The architecture consists of the subsystems of CINDI and their corresponding databases in which they store information at each step. The final working model of the integrated CINDI is shown in figure 8.8.

The revised architecture in figure 8.8 works as follows:

Step 1: CINDI system receives documents from CndRobot, ConfSys and Registration and Upload subsystem.

Step 2: Documents from CndRobot and Registration and Upload system are verified for their format. If they are in PDF format are sent to Final PDF repository else they are sent to non PDF repository for conversion into PDF.

Documents from Confsys are in PDF format and have an associated XML file stating their title, author and other details. These documents are sent to ASHG, where they get inserted into ASHG database

Chapter 9

Experiments and Results

The CINDI system subsystems are developed in various programming languages. CndRobot, File Conversion subsystem and ConfSys are developed in Java, Document filtering subsystem and ASHG is developed in C++, Registration and Upload and CINDI search subsystem is developed in PHP, JavaScript and AJAX. We have used free source MySql as the backend database for these subsystems.

In this chapter we will illustrate the performance of CINDI system after it has been integrated into a single system. We will test each subsystem and the information retrieved by the CINDI Search subsystem.

9.1 File Conversion Subsystem

Statistics of File Conversion subsystem shows enhancement in progress when we ported the system from Windows to Linux platform we have save lot of time over head. The old FCS subsystem downloaded documents from Linux platform to Windows, converted the documents in batch mode and uploaded the converted documents to Linux. So, the total time of conversion is calculated by summing up download time, conversion time and upload time. For the modified FCS subsystem there we don't have downloading and uploading step. Hence conversion time is the only time to be calculated. The table 9.1 shows results of time overhead of old FCS and modified FCS. Note that we have tested this against old FCS which is in working condition. The number of files consists of files in different format converted by old FCS and modified FCS.

Number of Files	size of files	Time using Old FCS (Download File time + Conversion time +Upload File time) in seconds	Time using New FCS (Conversion time) in seconds
4	6MB	$10 + 18 + 9.2 = 37.2$	18.6
2	3.83 MB	$6.3 + 8 + 5.2 = 19.5$	7.6
1	22.20 KB	$2 + 4 + 1.6 = 7.6$	3
8	24 MB	$32 + 54 + 30.3 = 116.3$	54.3
12	12 MB	$23.5 + 36.3 + 22.8 = 82.6$	35.9
10	10 MB	$21 + 32.7 + 20.3 = 74$	27
20	32 MB	$38 + 61 + 41.2 = 140.2$	62.2
22	11 MB	$22 + 34 + 21.8 = 77.8$	32.1
4	150 KB	$4 + 5.2 + 3.8 = 13$	4
32	152 MB	$68 + 72.3 + 69.3 = 209.6$	83
12	3.80 MB	$6.3 + 8 + 5.6 = 19.9$	7.9
2	22.20 KB	$2 + 4.2 + 1.6 = 7.8$	3.3
4	22 MB	$30 + 54 + 30.3 = 114.3$	52.1
8	12 MB	$23.5 + 39.3 + 22.8 = 85.6$	37.6
4	12 MB	$24 + 32.7 + 20.3 = 74$	32
12	11 MB	$24 + 34 + 21.8 = 79.8$	34.2

Table 9.1 Time overhead saved by porting FCS from Windows to Linux Platform

9.2 Registration and Upload subsystem

The registration and upload subsystem had been modified to support multiple file upload. In order to test the feasibility of the module we conducted tests to and performed multiple file upload in different set of batches. The table 9.2 shows the different number of files uploaded in batch, time taken in seconds and the total size of each upload.

Number of Files Uploaded	Time (seconds)	Total size of files
3	3.6	1.83 MB
6	6.0	2.26 MB
17	17.2	8.46 MB
20	19.8	9.98 MB
22	21.3	10.83 MB
48	47.2	22.86 MB
52	51.0	28.72 MB
68	56.2	33.54 MB
72	58.5	35.17 MB
80	60.2	40.12 MB
92	56.5	32.17 MB
100	62.2	40MB

Table 9.2 Statistics of number of files uploaded, time taken to file upload and the total size of files

9.3 Search Subsystem

Basic Search

The experiments and testing was conducted on the CINDI search subsystem to test the query response time for the different search terms entered by the users. The Query term in table refers to the keyword entered to test the search subsystem, time refers to time taken to execute the query and results refer to number of results that were fetched. The table 9.3 provides the statistics of the search terms with respect to time taken to fetch results and the number of results fetched for Basic Search module.

Query term	Time (seconds)	Number of Results
database	.58	555
database –image	.60	525
database –image +text	.60	416
data*	.80	3296
“database retrieval”	.56	160
+database +(>design <image)	.57	190
database design	.74	1978
fuzz* retrieval	.59	330
fuz* retrieval -image -text	.59	242
“image retrieval” – database	.58	120

Table 9.3 Statistics of query term searched, time take to process and number of results fetched

Advance search

Advanced search module provides users to search within title, author, abstract, keywords and subject of the documents. We conducted tests on advanced search module to test retrieval of desired query term in the ‘title of document’. Experiments were conducted to test the retrieval of query term “image” within the title of documents. Table 9.4 shows the title of the first 30 documents retrieved and whether they contained the term “image” in their title. Note that the results are case insensitive i.e image, IMAGE or iMAGE are all weighted same. The first column is the title of the document retrieved by the CINDI advanced search and the second column states whether we were able to find the queried term.

Title of Document	Results appropriate
Combining Local and Global Image Features for Object Class Recognition	Yes
Jigsaw Image Mosaics	Yes
Automatic Image Annotation and Retrieval using	Yes
IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 6, NO. 1, JANUARY 1997 103	Yes
DESIGN AND IMPLEMENTATION OF A SINGLE SYSTEM IMAGE	Yes
SuperResolution from Image Sequences A Review	Yes
A TwoPass Realistic Image Synthesis Method for	Yes
The Reyes Image Rendering Architecture	Yes
COLORSPATIAL IMAGE INDEXING AND APPLICATIONS	Yes
DIGIPAPER: A VERSATILE COLOR DOCUMENT IMAGE REPRESENTATION	Yes
Fast and Inexpensive Color Image Segmentation for Interactive Robots	Yes

An Automatic Hierarchical Image Classification Scheme	Yes
Efficient GraphBased Image Segmentation	Yes
Histogram Refinement for ContentBased Image Retrieval	Yes
SATBased Image Computation	Yes
Using SAT based Image Computation for Reachability	Yes
Fast and Cheap Color Image Segmentation for Interactive Robots	Yes
Automatic Image Captioning*	Yes
Unsupervised Image Classification, Segmentation,	Yes
GCap: Graphbased Automatic Image Captioning	Yes
746 IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 11, NO. 7, JULY 2002	Yes
IMAGE PROCESSING WITH CRAYONS	Yes
Motion estimation from image and inertial	Yes
Feature Selection in Example Based Image Retrieval Systems	Yes
266 IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 10, NO. 2, FEBRUARY 2001	Yes
Pictorial Query Trees for Query Specification in Image Databases	Yes
IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. XX, NO. Y, MONTH ZZZZ	Yes
Parallel Lossless Image Compression	Yes
IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 10, NO. 10, OCTOBER 2001	Yes
CAMEL: Concept Annotated iMagE Libraries	Yes

Table 9.4 List of first 30 results of the query term “database” searched within title of documents

We performed tests to retrieve the authors of the documents in CINDI repository. For the first experiment we searched for author named “desai”. Table 9.5 lists the results that were retrieved by CINDI advanced search.

Title of Document	Authors	Results appropriate
ASHG: Automatic Semantic Header Generator	S. Haddad B. Desai	Yes
Safe Inflight Mobile Telephony	Aditya Dua Aman Kansal Arjunan R Sumitra Ganesh Vivek Raghunathan U.B Desai	Yes
Bidirectional Expansion For Keyword Search on Graph Databases	Varun Kacholia Shashank Pandit Soumen Chakrabarti S. Sudarshan Rushi Desai Hrishikesh Karambelkar	Yes

Table 9.5 Search of author “desai” by using CINDI advance search within Author column in database

Similarity

In all our experiments, some pre-processing of the text was done first. We decapitalised the tokens and removed punctuation and stop words. These tokens were matched against the title and abstract of the target document. Table 9.6 provides the statistics of testing the similarity generated by the CINDI search subsystem and human similarity value tested on 15 documents. The target document title is “On Similarity-Based Queries for Time Series Data” and the result of similar document is listed in Title of Document and Similarity generated by CINDI Search subsystem columns.

Title of Document	Human Similarity Value	Similarity generated by CINDI Search subsystem (percentage)
Similarity Search for Web Services	7/10	65
Similarity Search Over Time-Series Data Using Wavelets	8/10	62
Benchmark Queries for Temporal Databases	6/10	58
Implicit Queries for Email	4/10	53
A novel architecture for active service	4/10	52
Generating Hierarchical Summaries for Web Searches	5/10	52
Cache and Query for Wide Area Sensor Databases	5/10	51
Similarity-based Multilingual Multi-Document Summarization	4/10	51
Entropy-Based Bounds for Online Algorithms	4/10	51
Generating Hierarchical Summaries for Web Searches	5/10	51
Streaming Queries over Streaming Data	5/10	51
Optimization Strategies for Instance Retrieval	4/10	50
Efficient Visibility-Based Algorithms for Interactive	6/10	50
Protein Similarity from Knot Theory	4/10	50

Table 9.6 Similarity matching of document title and abstract generated by search subsystem compared with human similarity value

The experimenting and testing showed that search subsystem is able to extract most of the fields such as title, author name, abstract and keywords.

Chapter 10

Conclusions and Future works

9.1 Conclusions

While conducting the literature review we found out that the CINDI subsystems were required to be integrated into a unified system, the search subsystem was required to be modified to provide advance features and usability. This thesis provides insight to data integration of various systems and enhancements made to the CINDI system. We have discussed integration testing techniques that were performed while integrating subsystems of CINDI which includes: the CINDI Robot, the ConfSys, the Registration and Upload subsystem, the Annotation subsystem, the Gleaning subsystem, the Automatic Semantic Header Generation subsystem and the Search subsystem into a unified system called CINDI digital library. We made enhancements to File Conversion subsystem by porting it to Linux platform, hence reducing a substantial time overhead.

We have enhanced the Search subsystem which now provides enhanced search techniques to the users to find desired information from CINDI Repository; users can make use of advanced search system to search within or any combination of title, abstract, keywords, subject and author of the documents. The annotation system works hand in hand with the Search subsystem providing CINDI users to annotate the documents that they download and thereby increasing page rank. The graphical user interface for Registration and Upload subsystem has been modified to provide better usability. We have designed and implemented Administrative panel to manage information of the Registration and Upload subsystem and the Annotation

subsystem. The overall enhancement to CINDI system shows promising results from statically tests provided in chapter 8.

9.2 Contribution of thesis

The key objective of this thesis to enhance the functionality of the subsystems in CINDI digital library and integrate them into a unified system. This thesis presents an integrated architecture of CINDI system designed to overcome the challenges of interoperability, extensibility and scalability of the old subsystems. It is completely a new architecture that is utilizing an enriched data model. The File Conversion subsystem of CINDI has made use of freely available Linux commands and OpenOffice. This clearly meets criteria of a free PDF converter on Linux platform. The porting of PDF converter from Windows platform to Linux helped the system to achieve same functionality provided by old FCS with minimized conversion time.

We have presented design, implementation and evaluation of the CINDI Search subsystem for keyword, phrase and advance search. The CINDI Search subsystem takes into account (a) User reviews and clicks to increase the rank of the document, (b) provides an interface to search within title, abstract, author and keywords of document by use of automatic semantic header generator. Our experiments and evaluations show the query evaluations and performance gains by making use of full text index. Our observations are based on evaluations in Chapter 8 showing that CINDI Search subsystem is practical to use on moderately large databases and the results are intuitive and useful.

We have enhanced the usability of Graphical User Interface of the old subsystems. We have achieved this by modifying the complex web interfaces in the Registration and

Upload subsystem and Search subsystem into simpler and easy to understandable interfaces. We structured the User Interface in logical sections and grouped the interface elements by their functionality.

9.3 Future Work and Suggestions

Based on the working prototype of integrated system, future investigations will go into several directions. For instance, we have currently taken into consideration searching within title, author, abstract, keywords and subject. However the literature study showed that similar academic search systems make use of documents citations to increase ranking of documents. The spell check can be introduced for query input by the user. Enhancements can be performed to the query parser to provide queries like title: 'abc' or author: 'james' in search textbox. Query expansion and relevance feedback can be implemented to increase number of results. Enhancements can be made to annotation subsystem by allowing annotators to comment on particular paragraph, formula or figure within the PDF document.

References

- [1] Yuhui W, “Enhanced Web based CINDI system”, major report, Dept. of Computer Science, Concordia University, 2002

- [2] Xiaome Y, “Web based CINDI system: graphical user interface design and implementation”, major report, Dept. of Computer Science, Concordia University, 2001

- [3] Sami Samir Haddad, “Automatic semantic header generator”, master thesis, Dept. of Computer Science, Concordia University, 1998

- [4] Tong, Z, “A Gleaning Subsystem for CINDI”, master thesis, Dept. of Computer Science, Concordia University, 2004.

- [5] Cong Zhou, “CNDROBOT – A Robot for the CINDI Digital Library”, Master Thesis, Dept. of Computer Science, Concordia University, December 2005

- [6] Akim Demaille, Miguel Santana, “GNU a2ps”, viewed 24 September 2007, <<http://www.inf.enst.fr/~demaille/a2ps/index.html>>

- [7] George Ferguson, “Ghostscript”, viewed 20 September 2007, <<http://pages.cs.wisc.edu/~ghost/>>

- [8] Dante E.V., June 2007, “LaTeX – A document preparation system”, viewed 20 September 2007, <<http://www.latex-project.org/>>
- [9] Martin Schröder, August 2007, “pdfTeX”, viewed 12 September 2007, <www.pdfTeX.de/tex/pdfTeX/>
- [10] Jean Hollis Weber, “OpenOffice.org Writer: The Free Alternative to Microsoft Word”, O'Reilly Media, Inc., July 2004
- [11] Gennick, Jonathan, May 2003, “OpenOffice.org Saves my Day, Again”, viewed 12 September 2007, <http://www.oreillynet.com/databases/blog/2003/05/openofficeorg_saves_my_day_aga.html>
- [12] Gerry Roderick Singleton, “The OpenOffice.org Documentation Project”, May 2007
- [13] Andrew Pitonyak, 2004, “OpenOffice.org Macros Explained”, viewed 10 September 2007, < <http://documentation.openoffice.org/>>
- [14] Xiaomer Y, “Web based CINDI system: graphical user interface design and implementation”, major report, Dept. of Computer Science, Concordia University, 2001
- [15] Reuven M. Lerner, 2006, “Beginning Ajax”, viewed 8 September 2007,

< <http://www.linuxjournal.com/article/9256>>

[16] Reuven M. Lerner, 2006, “JavaScript, Forms and Ajax”, viewed 8 September 2007,

<<http://www.linuxjournal.com/article/9160>>

[17] Liam R. E. Quin, 1994, “A Text Retrieval Package for the Unix Operating System”,

viewed 10 September 2007, < <http://citeseer.ist.psu.edu/371401.html> >

[18] Patrizia Andronico, Marina Buzzi and Barbara Leporini, December 2004,

“Accessibility and Usability of Search Engine User interfaces”, viewed 8 September

2007, < <http://ui4all.ics.forth.gr/workshop2004/adjunct/accessibility/58.pdf>>

[19] Ramakrishnan Srikant and Yinghui Yang, “Mining Web Logs to Improve Website

Organization”, viewed 8 September 2007, < <http://citeseer.ist.psu.edu/447490.html> > ,

[20] Lin Guo Feng Shao, Chavdar Botev and Jayavel Shanmugasundaram, June 2003,

“XRANK: Ranked Keyword Search over XML Documents”, viewed 8 September 2007,

< <http://www.cs.cornell.edu/people/jai/papers/XRank.pdf>>

[21] Alireza Noruzi, 2005, “Google Scholar: The New Generation of Citation Indexes”,

viewed 8 September 2007,

<http://eprints.rclis.org/archive/00005542/01/Google_Scholar,_The_New_Generation_of_Citation_Indexes.pdf>

[22] “An interview with Anurag Acharya, Google Scholar lead engineer”, viewed 8 September 2007, <http://www.google.com/librariancenter/articles/0612_01.html>

[23] Craig Silverstein, December 2006, “Analysis of a Very Large AltaVista Query Log”, viewed 8 September 2007, <<http://citeseer.ist.psu.edu/silverstein98analysis.html>>, 1998

[24] Kurt D. Bollacker, Steve Lawrence and C. Lee Giles, 1998, “CiteSeer: An Autonomous Web Agent for Automatic Retrieval and Identification of Interesting Publications”, viewed 8 September 2007,
<<http://maya.cs.depaul.edu/~classes/csc575/papers/citeseer.pdf>>

[25] Steve Lawrence, C. Lee Giles and Kurt Bollacker, 1999, “Digital Libraries and Autonomous Citation Indexing”, viewed 10 August 2007,
<<http://clgiles.ist.psu.edu/papers/IEEE.Computer.DL-ACI.pdf>>

[26] Martin Porter, “Porter Stemmer”, January 2006, viewed 10 August 2007,
<<http://tartarus.org/~martin/PorterStemmer>>

[27] Jonathan J. Oliver, 1993, “Decision Graphs- An extension to decision trees”, viewed 10 August 2007, <<http://citeseer.ist.psu.edu/oliver93decision.html>>

[28] Kazunari Sugiyama, Kenji Hatano, Masatoshi Yoshikawa and Shunsuke Uemura, 2005 “Improvement in TF-IDF scheme for Web pages based on the contents of their hyperlinked neighboring pages “, viewed 10 August 2007, < <http://doi.wiley.com/10.1002/scj.20189> >

[29] Wayne Gramlich, 1998, “A Scalable Annotation System for the World Wide Web”, viewed 10 August 2007, < <http://gramlich.net/projects/annotations/index.html>>

[30] Martin Röscheisen, Christian Mogensen and Terry WinogradShared, “Web Annotations as a Platform for Third-Party Value-Added Information Providers: Architecture, Protocols, and Usage Examples”

[31] Ovsianikov, I., M. Arbib, and T. McNeill, 1998 “Annotation Software System Design”, viewed 10 August 2007, < <http://citeseer.ist.psu.edu/roscheisen94shared.html>>

[32] Vincent Quint and Irène Vatton, November 2004 “Techniques for Authoring Complex XML Documents”, viewed 10 August 2007, < <http://wam.inrialpes.fr/publications/2004/DocEng2004VQIV.html>>

[33] Joe Abbott, “Unit and Integration Testing Manual”, Blackwell Publishers, 31 March 1988.

- [34] Rick D. Craig, Stefan P. Jaskiel, “Systematic Software Testing (Artech House Computer Library)”, Artech House Publishers, May 2002
- [35] Thomas J. McCabe, Charles W. Butler, December 1989, “Design complexity measurement and testing”, *Communications of the ACM*, viewed 10 August 2007, < <http://www.computing.dcu.ie/~renaat/ca4papers/p1415-mccabe.pdf>>
- [36] Yuwei Feng, “CONFSYS: Enhancement and Integration”, master thesis, Dept. of Computer Science, Concordia University, December 2004
- [37] Yanhong Li “Enhancement of CINDI System”, master thesis, Dept. of Computer Science, Concordia University, August 2003
- [38] B.C. Desai, May 1994, “A System for Seamless Search of Distributed Information Sources”, viewed 10 August 2007, <<http://www.cs.concordia.ca/~bcdesai/web-publ/w3-paper.html>>
- [39] Paul DuBois, Kim Aldale, Yan Cailin, Jay Flaherty, 2007, “MySQL documentation webpage”, viewed 10 August 2007, <<http://dev.mysql.com/doc/refman/5.0/en/fulltext-search.html>>

Appendix A: Stop Word List

a	ago	always	another
been	but	cannot	do
get	going	had	he
least	like	maybe	most
our	per	saw	she
them	they	till	unless
whether	whoever	will	won't
about	all	am	any
before	by	come	does
getting	gone	has	her
left	make	me	much
ourselves	put	see	should
then	this	to	until
which	whom	with	would
after	almost	an	anybody
being	came	could	doing
go	got	have	here
less	many	mine	my
out	putting	seen	so
there	those	too	up
while	whose	within	wouldn't
again	also	and	anyhow
between	can	did	done
goes	gotten	having	him
let	may	more	myself
over	same	shall	some
these	through	two	upon
who	why	without	yet

anyone	i	at	than
down	none	every	whatever
his	something	into	be
never	was	of	for
somebody	are	sure	it
us	even	were	one
you	if	away	that
anything	not	everyone	what's
each	stand	is	became
how	we	off	from
no	as	take	just
someone	ever	what	onto
very	in	back	the
your	now	everything	when
anyway	such	isn't	because
else	went	on	front

Appendix B: Impolite Words List

shit	ejakulate	pusse
bitch	fatass	pussy
blowjob	fcuk	scrotum
clit	fuk	sh!t
fuck	fuxr	shemale
ass	hoer	shit
asshole	hore	shi+
b!tch	jism	sh!+
bch	kawk	slut
btch	litch	smut
bastard	li+ch	teets
bi+ch	lesbian	tits
boiolas	masturbate	boobs
buceta	masterbat*	bbs
cck	masterbat	teez
cawk	motherfucker	titt
chink	s.o.b.	wse
cipa	mofo	jackoff
clits	nazi	wank
cock	nigga	whoar
cum	nigger	whore
cunt	nutsack	f***
dildo	phuck	fu**
dirsa	pimpis	