CNDROBOT – A ROBOT FOR THE CINDI DIGITAL LIBRARY SYSTEM

CONG ZHOU

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 AT
CONCORDIA UNIVERSITY
MONTREAL, QUEBEC, CANADA

DECEMBER 2005

Your file Votre référence
ISBN: 0-494-14343-6
Our file Notre référence
ISBN: 0-494-14343-6

ABSTRACT

CNDROBOT – A Robot for the CINDI Digital Library System

Cong Zhou

Web robots or crawlers are an essential component of all search engines. Major search engines such as Google and AltaVista use their own robots (GoogleBot and Mercator) to crawl and index billions of Web pages over the Internet. Web robots are also increasingly adopted by digital libraries to collect data and on-line documents. The crawling process requires massive amounts of hardware and network resources as well as time. However, when only information about a predefined topic set is desired, the use of traditional crawling strategy becomes inefficient and cost ineffective.

This thesis presents issues in developing a focused crawler - CNDROBOT, which only explores well-selected domain sites and collects potential on-topic documents for the CINDI digital library. The research was concerned with the studies on various search engines, types of Web robots, and crawling strategies. The research primarily involved the design and implementation of the CNDROBOT as well as the integration of the Document Filtering Subsystem. Finally, a Web application for the CNDROT was developed and an extensive test was conducted for various components and functions of this system. This thesis demonstrates that the CNDROBOT is capable of effectively and efficiently discovering large amounts of desired documents and supplying them for the CINDI digital library.

Acknowledgements

First, I would like to thank my supervisor, Dr Bipin C. Desai, for his support, care and patience. His insight and ideas formed the foundation of this thesis as much as mine did. His guidance helped me go through various hurdles during my graduate years. I admire his down-to-earth work spirit, which has set a great example for me in my work.

Second, I would like to thank my aunt and uncle for their generous love and support. When I was swamped by writing this thesis and working a full-time job, they took care of almost everything for me in my daily life without any complaints.

Many thanks to my team members who worked on various components of the CINDI system, including Zhang Tong, Wang Tao, and Xue Furong. Zhang Tong designed and implemented the DFS subsystem that runs seamlessly with the CNDROBOT. Wang Tao also gave me a hand whenever I needed him.

Special thanks for Mary O'Malley in the Student Learning Service office for the writing assistance on my thesis.

I would also like to thank Halina Monkiewicz, the secretary of the department of Computer Science & Software Engineering, for her patience and advices.

Finally, I would like to thank my parents who are back in China for their love and encouragement. Many years ago, they made a tough decision to send their only son to a country thousands miles away. I am very grateful for their unselfish decision. I know they are proud of me as always. I hope that we will live together one day soon.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Problem Statement

The size of the Web is growing exponentially. Current estimates are that the number of searchable web pages of text, images and various multimedia information on the web have exceeded 8.9 billion [LV03] with numbers doubling in less than one year [PR99]. On one hand, the Web provides us with a vast resource for information and facilitates commercial and academic intelligence research; on the other hand, the enormous size of the Web, its diversity and its dynamic nature makes the task of seeking appropriate information difficult. Exploring the Web and locating relevant documents to automatically build a significant collection (digital library), to serve the general web community has evolved as an active research area in the past decade. Web crawling technology, originally developed for the benefit of search engines, now is being seriously considered as an important strategy for building large-scale digital libraries. However, designing and implementing a crawling tool to effectively and efficiently discover desired Web documents from the large and heterogeneous Web resources poses many challenges. CINDI Web robot integrated as part of CINDI (Concordia INdexing and DIscovering System) digital library project helps collect, populate relevant on-line documents (research papers, technical notes, FAQs) in the computer science field.

1.2 Proposed Solution

Web robots which are also referred to as crawlers, worms, spiders or wanderers, retrieve pages from the Web by recursively following URL links in pages using standard HTTP protocols [MC03].

Digital libraries have typically used exhaustive crawlers to build and update large collections of documents. However, the design of a good crawler has many challenges.

Externally, the crawler must avoid overloading Web sites or network links as it goes about its business [MK95]. Internally, the crawler must deal with a huge volume of data. Unless it has unlimited computing resources and unlimited time, it must carefully decide what URLs to scan and in what order. The crawler must also decide how frequently to revisit pages it has already seen, in order to keep its client informed of changes on the Web [JC98].

CINDI web robot is a focused crawler, which starts with a set of seed URLs; these are the trusted sites with high hub and authority scores. CNDROBOT extracts and follows the hyperlinks from the Web pages, filters unwanted documents (email archives, discussion group, video and audio files etc) [TZ04] downloads and indexes the good research materials e.g. research papers in different file formats to local repository and revisits the Web pages to maintain the freshness of the digital library and discover new resources. After the first crawling, the Document Filter System (DFS) determines quality (good or bad document) for each downloaded files. The statistics analyzer calculates the values (gives scores) for each seed site, which has valid downloaded papers according to the feedback from DFS and then determines frequency of visiting and determines those sites, which should not be revisited.

1.3 Organization of the Thesis

This thesis is organized as follows: Chapter 2 introduces background on Web robots, presents an overview of digital libraries and CINDI System, highlights some existing search engines, describes general-purpose and focused web robots, and illustrates the role of CNDROBOT in the overall CINDI system as well as interaction with other subsystems. In Chapter 3, the architecture and approaches of the CNDROBOT are presented and some heuristics to evaluate and improve its performance are illustrated. Chapter 4 describes the details of implementation of CNDROBOT – a focused web crawler. Chapter 5 discusses the analysis of testing results and some performance improvements. In Chapter 6, we draw our conclusions and present the future work on CNDROBOT.

Chapter 2
Background on Web Robots

2.1 Overview of Digital Libraries and CINDI System

The US Library of Congress, the largest library in the world, has collected more than 130 million items [LC]. It has a user population of approximately 100,000 people. However, there are one thousand times more potential users of the Web with the number still growing rapidly to a level of a few billion [RMC]. Digital libraries modeled on traditional libraries have advantages over them in terms of potential number of users, volume of collections and accessibility. The other benefit of digital libraries is to shift from dependency on expensive human labor to the combination of relatively inexpensive computing power and intelligent algorithms. There are many large scale digital libraries specialized on scientific research documents, such as CiteSeer [SLDL], California Digital Library [CDL], Stanford Digital Library [SDL], ACM Digital Library [ADL] and National Science Digital Library [NSDL].

The CINDI (Concordia Indexing and DIscovering System) system was conceived in 1994 [BC94]. 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 [SHBC].

2.2 Existing Search Engines, Digital Libraries and Web Robots

"*The grandfather of all search engines was Archie, created in 1990 by Alan Emtage, a student at McGill University in Montreal.*" [WT98] Around that time, the Hyper Text Transfer Protocol (HTTP) was being developed and the primary method of storing and retrieving files was via File Transfer Protocol (FTP). Archie combined a script-based data gatherer, which fetched site listings of anonymous FTP files, with a regular expression

matcher for retrieving file names matching a user query. Its gatherer scoured FTP sites across the Internet and indexed all files it found. Its regular expression matcher provided users with access to its database. Nowadays, using search engines such as Google, Yahoo!, Altavista [AVA], InfoSeek [IFSK], Excite [ECT], Lycos [LCS], and MSN is the most popular way to search for information on the Web.

2.2.1 Major Search Engines and Digital Libraries

Search engines can be categorized as general-purpose search engines, MetaSearch engines and specialized search engines. Both search engines and digital libraries rely on the robot to discover resources and collect documents, which are then indexed.

2.2.1.1 General -Purpose Search Engines

Alta Vista

Alta Vista was originally developed at the Western Digital Palo Alto Research Center (PARC) of the Digital Equipment Corporation [SSAV] and had great impact beginning December 15, 1995. Its presence on the web helped to establish search engines as part of the Internet landscape. It was the first in many categories: first to do full-text index, first with multiple language searching capability, first with multi-language translation, with support for non-Latin alphabets such as those used in Chinese, Japanese and Korean, first with a spell checking function and one of the first to do a good job with image and non-text documents. Among its special features is a technology called *RealNames* that checks the search terms entered by the user against an internal database of registered and common-law company, product and concept names and marketing slogans. If *RealNames* finds a match, it points to the name-owner's Web page. Another of AltaVista's powerful features is its ability to index over 200 types of documents. This means it can read the contents of files such as Word documents that are put up on the web. However, AltaVista crawls only the HTTP protocol and therefore cannot find files on FTP, Gopher, or NNTP sites [NL02].

Google

The success of Google has become a legend in information technology and business success. Google (earlier named WebBase) started out as a research project at Stanford University. Google uses a modified "link cadinality" [GH00] algorithm that takes into account the domain of the citing source, giving greater weight to an .edu or .org domain than a .com domain. Link cardinality essentially counts the number of other sites linking to a given site. Google uses it as one of the metrics to identify the most important and credible sources of online information and give rank to a Web page. Google has trademarked this with the name *PageRank*™. Another useful feature is retaining a local copy of the Web page when its crawler originally downloads and analyzes it so that users can always have access to that page even if it no longer exists. In addition, Google can also search for documents in PDF format, a standard for a great many documents. Google has become the most powerful and comprehensive search engine in terms of the volume of its indexed Web pages. According to Google, it has indexed more than 8 billion pages [GB].

MSN

Microsoft released a public preview of its long-awaited web search technology in July 2004, over a year after first embarking on the project. [DS04]. MSN is still beta-testing its search engine, which relies upon results from Overture [OV] (formerly GoTo) and Yahoo. Although the results provided by MSN searches do not vary considerably from Yahoo, Microsoft has been able to employ its considerable market weight in operating systems to spread the word of its upcoming search engine in advance. MSN.com is the default in most IE browsers including Windows 2000. Microsoft claimed that they had made significant improvements to this Beta search engine. Some of these improvements include *vast index of information* (it has indexed 5 billion documents and refreshed continuously); *direct answers* (direct answers are provided in a number of categories, including definitions, facts, calculations, conversions, and solutions to equations); *content-specific search* (MSN Search offers the ability to search for specific type of

information using search tabs, including web, news, and images); *search near me* (users receive search results tailored to their geographic location); and *search builder* (customize search results by emphasizing or de-emphasizing certain search criteria, such as specific sites or domains, country or region or language) [SANDBOX].

2.2.1.2 MetaSearch Engines

Metasearch engines employ a selected number of other search engines to run a search. They act like middlemen, wholesalers, who send a search to their partners and retrieve the cream of the results.

Unlike the individual search engines and directories, the Meta Search Engines, do not have their own databases and do not accept URL submissions.

In theory, this is a good idea, since they should be able to take advantage of the combined strengths and indexing depths of a range of other search engines. In practice, they have some considerable drawbacks. Because they take one search query and translate it into a form that can be processed by each of their feeder search engine partners, there is often a loss of precision, particularly for all but the simplest queries. In addition, because MetaSearch engines usually collect the first few results from each search engine; this strategy would miss some relevant information that users may need [FEIS].

Obviously, MetaSearch engines will never be better than the strongest of the various partner search engines. They deliver poor performance when the search, sent to the different partners, hits a time out (a fairly frequent occurrence) or have difficulties for any reason [MSE].

MetaCrawler

MetaCrawler started as a university project in 1994, at the University of Washington. Professor Oren Etzioni and graduate student Erik Selberg launched it in 1995, and it was

one of the first MetaSearch engines. It employs the following search engines as its "scouts": Google, Yahoo, AskJeeves, About, LookSmart, Overture (formally GoTo) [OV], AltaVista, and FindWhat. Although this list is impressive, any given search will only be sent to a handful of search engines all at once. The results are then blended together into one page. MetaCrawler displays to the users the sorted list of results based on how frequently a web site appears in the top rankings of each partner search engine.

IXQUICK

For searches on the Web, Ixquick utilizes AOL, AltaVista, EuroSeek, Excite, Fast Search (AlltheWeb), FindWhat, GoTo, LookSmart, MSN, and Yahoo. For the document formats such as video, audio files, they utilize a smaller number of specialized search engines, which seem to work quite well.

Ixquick makes a very nice first impression: a good clean interface and the opportunity to use over a dozen, mostly European languages, from Finnish to Turkish. Choice of document formats includes Web documents, News, Movies and Images.

Results are cleanly listed, with enough information to help a user decide whether to visit the site or not. Sponsored links are clearly marked and relatively unobtrusive. The help-screens are very good, and provide not only basic information but also show clearly how to get the most out of IXQUICK.

2.2.1.3 Specialized Search Engines

An article written by Gary D. Price [GP01] lists a few reasons why specialized search engines are vital. First, the specialized search engines can accomplish higher precision rate [1] and recall rate[2] since they focus on searching Web space limited to a specific subject, domain or format. Second, the specialized search engines can support the needs

[1] Precision rate is the ratio of the number of relevant pages retrieved to the total number of documents retrieved.
[2] Recall rate is the ratio of the number of relevant pages retrieved to the number of all relevant pages on the Web.

of users who do not have the time or interest to learn how to use the general engines at an advanced level. Third, the specialized search engines can provide fresher content to the database than general search engines do since they have much short crawling period compared to those for general search engines, which usually take weeks. Finally, specialized search engines often provide access to "hidden" Web space, which are the contents not crawled by general search engines.

There are many specialized search engines that have been built to explore particular Web site(s), specific topics (such as computers or medicine), languages, file types (such as images or research papers) and so on [MCHC03]. These specialized search engines usually provide more precise results and more customizable functions. For example, Wikipedia offers a free content encyclopedia in many languages (German, French, Chinese, Japanese and many more). Industrial Quick Search searches for company information and ZI-Bot specializes in searching for ISBN and ISSN information from more than 600 libraries. There are also content-type-specific search engines. For instance, FindSounds searches the web for sound effects and musical instrument samples; CiteSeer provides the services for users searching for computer and information science papers.

CiteSeer

CiteSeer is a specialized search engine for scientific research literature. CiteSeer is a free public service, and is the world's largest free full-text index of scientific literature. It acclaims that it has indexed 723,152 articles with links to ACM Portal and DBLP [CITE]. It provides many interesting features in algorithms, techniques, and software that can be used in other digital libraries. For example, CiteSeer includes algorithms and machine learning techniques for automatically extracting information such as the title and author from indexed documents and individual citations. It also can automatically create an index of the links between scientific articles. Another impressive feature is that CiteSeer provides full-text indexing of the content of Postscript and PDF documents.

2.2.1.4 Survey on Major Search Engines

Search Engine	Robot	Purpose	Software Language	Software Platform	Owner's Name
Google / Netscape	GoogleBot	Indexing	C++	Linux	Google Inc.
AltaVista	Scooter/Mercator	Indexing	C	Unix	AltaVista
MSN	MSNBot	Indexing	C++	Windows Server 2000/2003	Microsoft Corp.
AOL/InkTomi	Slurp	Indexing, Statistics	C/C++	Unix	InkTomi Corp.
Excite / Webcrawler	ArchitextSpider	Indexing, Statistics	Perl 5 and C	Unix/Solaris	Architext Software
InfoSeek	InfoSeek Robot 1.0	Indexing	Python	Unix/Solaris	Steve Kirsch

Table 1 Survey on Major Search Engines

Table 1 shows a survey conducted of existing major search engines from various sources [WRD] [TOP] on the World Wide Web. The survey collected information on the Robot, Purpose, Language, Platform and Owner for some popular search engines. Note that all of the search engines are used as tools of indexing web pages. It should not be surprising that most web robots are written in C/C++ since it was the most dominant programming language at the time they were developed. Most of them run on Linux or Unix operating systems.

CNDROBOT has been written in Java programming language, which has a reputation for developing large scale, distributed, robust, portable, multithreaded, and dynamic projects, like the CINDI system. CNDROBOT is designed in a modular way and takes advantages of Java's rich libraries as well as its support for several aspects of run-time performance, including garbage collection, and heap allocation.

2.3 General Purpose Web Robots and Focused Web Robots

Web robot is an automated program, which implements as a graph search algorithm that works by treating web pages as nodes and links as edges. Each time a Web page is reached, it is parsed in order to find both its text content and the Universal Resource Locators (URLs) that it links to. The robot adds these URLs to a queue of pages to be visited; this queue is often referred to as the crawl frontier. The robot uses the URLs from the queue in some order, and repeats the process until the queue is empty. A typical crawling process is illustrated in Figure 1:

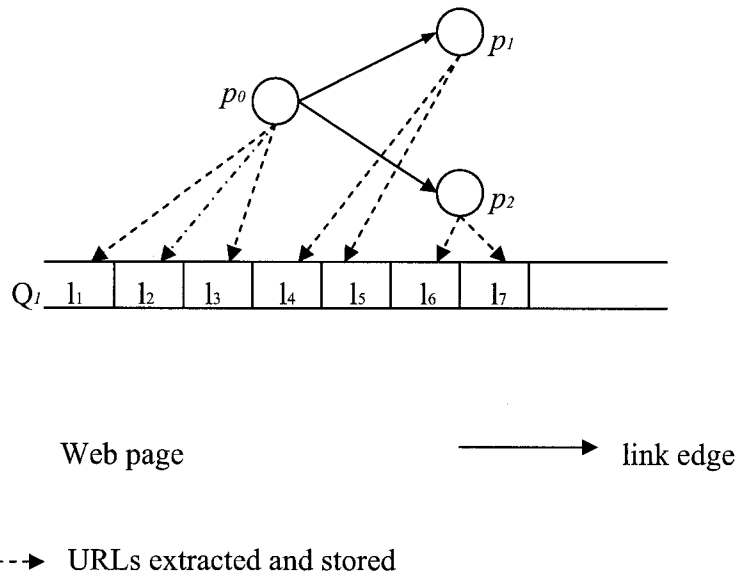

Figure 1 Illustration of a typical crawling process

Figure 1 shows that Web page p_0 is parsed and URLs (l_1, l_2,l_3) are extracted and placed in the queue Q_1. The web robot picks the next link (l_1) from the head of the queue, fetches the corresponding Web page (p_1), extracts the URLs (l_4, l_5) from p_1 and appends them to the queue.

Web robot usually traverses the Web using one of the following methods [VG97]: In one scheme, a set of "seed URL" is used for initiating exploration. The web robot fetches the

seed pages, extracts from them hyperlinks pointing to other Web pages, and then visits each of these URLs recursively using breadth-first or depth-first searching algorithm. Alternatively, the robot begins with a set of popular web sites' home pages with the rationale that these pages contain URLs that point to the most frequently sought information on the local and other Web servers and then searches recursively. A robot may partition the Web space based on Internet names or country codes and assigns one or more robots to explore the space exhaustively.

Both search engines and digital libraries rely on web robot to collect vast amount of documents from the World Wide Web. A web robot can be categorized as either a general-purpose web robot or a focused web robot from the point of view of scalability and goal. Unlike general purpose web robots, which have a goal of providing search capability over the Web as a whole and strive to cover Web pages as widely as possible, focused web robots have a goal of selectively seeking out and following the hyperlinks that may lead to more topic-related Web pages. They take a set of well-selected Web pages related to a predefined set of topics and recursively explores the linked Web pages. Usually the focused crawlers rely on a classifier to maintain the crawler focus by evaluating the relevance of a Web page using domain keywords or exemplary documents and following relevant links and filtering away irrelevant links that lead to off-topic areas of the Web.

Davison [DB00] presented empirical evidence that there is a topical locality on the Web. The author defines topical locality as the degree of relevance of textual content of a Web page to the topic. He concludes that by following links from a page that is relevant to the topic, the chance is significantly higher to get topically related pages than following other randomly selected pages. This locality were used in designing effective techniques for focused crawlers, for example Atrax/Mercator crawler [HA99][NM01], which starts at a few well-chosen points and maintains the crawler within range of these known topics in order to discover new resources.

2.4 CNDROBOT and Other Subsystems

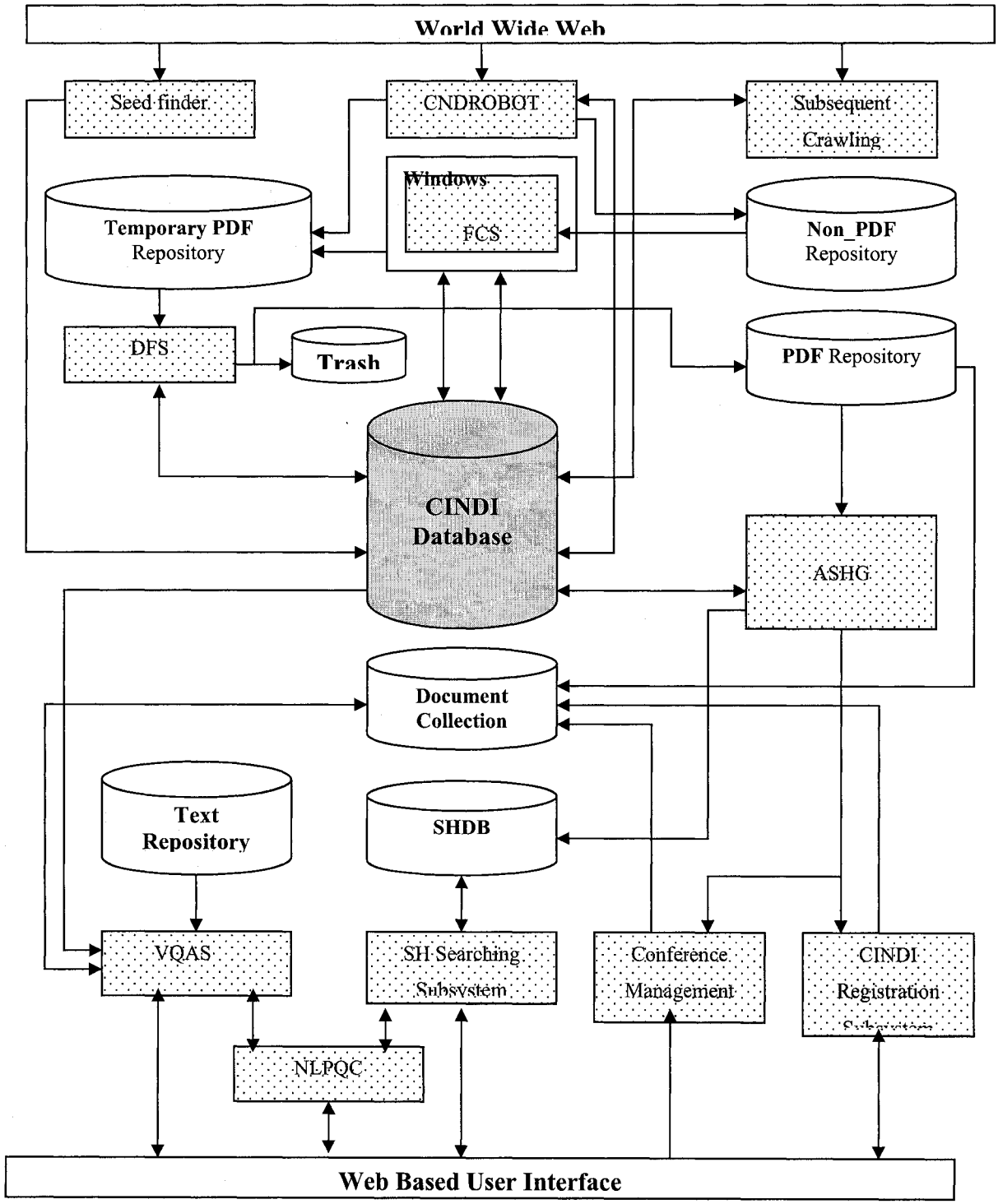

Figure 2 Components of the CINDI System

As illustrated in Figure 2, the CINDI system consists of the following eight essential subsystems: CNDROBOT Subsystem, File Converting Subsystem (FCS), Document Filtering Subsystem (DFS), Automatic Semantic Header Generator (ASHG), Virtual Query and Answering Subsystem (VQAS), Semantic Header Searching Subsystem (SHSS), Natural Language Processor for Querying CINDI (NLPQC), Conference Management Subsystem (CONFSYS) and CINDI Registration Subsystem. This thesis will focus on the CNDROBOT Subsystem and its interaction with the File Converting Subsystem.

The CINDI library collects the academic documents in two ways: using either the push or pull paradigm. In the push scheme, authors submit documents in PDF, TXT, LaTeX, RTF and HTML formats directly through the CINDI Registration Subsystem or Conference Management Subsystem. In the pull scheme, the CNDROBOT traverses the World Wide Web and downloads documents, which follow HTTP or HTTPS protocol in PDF, DOC, HTML, TXT, RTF, XML, PPT, LaTeX, and PS formats.

In the pull method, effectively and efficiently locating documents on the Web, downloading them to a local repository and indexing for future user inquiry are the base of other CINDI subsystems. The CNDROBOT uses Seed Finder to acquire good starting URLs for crawling the Web, HTML Parser to extract the links from the Web page, File Fetcher to download the documents and Statistics Analyzer to evaluate the crawling results by taking into account the feedback from DFS and schedule the subsequent crawling order.

The main task of FCS is to convert the non-PDF documents to PDF format. Then those converted documents are filtered by DFS. DFS evaluates the quality of a document, by examining its content as well as its structure. DFS rejects irrelevant documents such as emails, letters, news, pictures, assignments, application forms, slides and others and throws them into CINDI trash. DFS accepts relevant documents such as research papers, technical reports, FAQs, theses and academic papers, and stores them in a permanent repository.

Automatic Semantic Header Generator (ASHG) subsystem fetches documents from PDF directory that are converted into plain text file; generates Semantic Headers [ZZ02], which include author, title, subject and abstract; and then stores them in the SHDB. It also stores a copy of text files in the Text Repository where VQAS uses them to create indexes for virtual query and answer.

Another CINDI subsystem the Natural Language Processor for Querying CINDI (NLPQC) takes users' queries in natural language, processes them and returns relevant contents. Users can also access the PDF version of documents in the Document Collection Repository through the links provided in the response. CINDI allows users via the Semantic Header Searching subsystem to query SHDB using simple or complex combination of author, title, subject, or keywords.

As shown in Figure 2, the CNDROBOT, FCS, DFS, ASHG and VQAS share the CINDI database. The CNDROBOT stores into this database the information about downloaded documents, which is retrieved and updated by FCS, DFS, ASHG and VQAS subsystems.

This section has described the major components in the CINDI system and the role of the CNDROBOT in the system. The CNDROBOT infrastructure, database tables, crawling approach and some heuristics to improve crawling performance will be presented in the following chapter.

Chapter 3

CNDROBOT Architecture, Approach and Heuristics

3.1 CNDROBOT Architecture

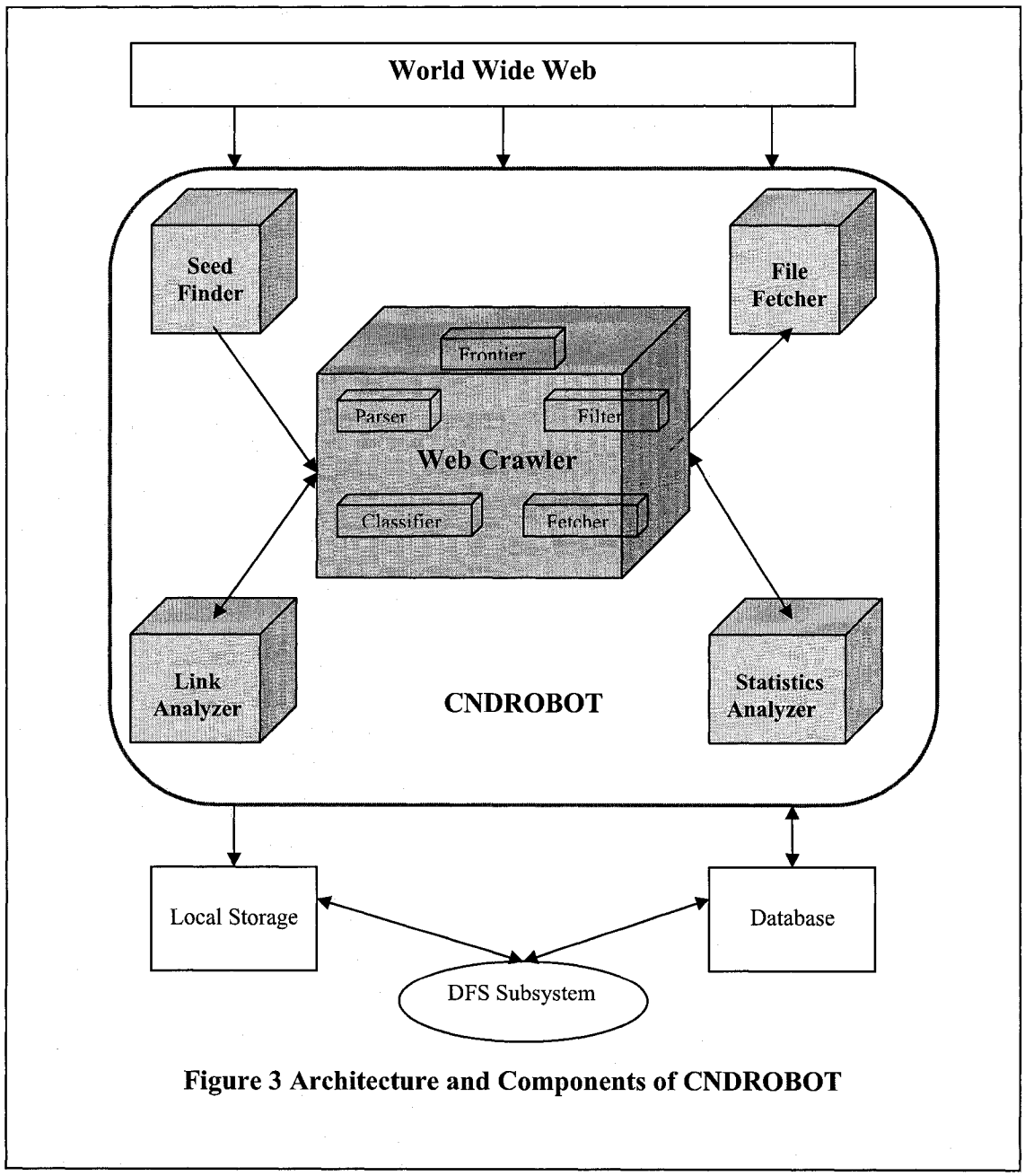

Figure 3 Architecture and Components of CNDROBOT

Fig 3 illustrates the architecture and components of the CNDROBOT. It consists of the following five components: Seed Finder, Web Crawler, File Fetcher, Statistics Analyzer, and Link Analyzer. The Web Crawler works with other components in an interactive way. It first takes the seeds generated by the Seed Finder and then crawls the seed sites to discover potential documents. In the crawling process, the Web Crawler performs various tasks e.g. page download, content parsing, and links extraction, etc. It also stores relevant information into various tables in the CINDI database. Some of this information is used by File Fetcher to download documents to local storage. File Fetcher stores information regarding the downloaded documents, their URLs, types, sizes, and location. DFS retrieves the downloaded documents from the local storage; determines their quality; put them into a permanent location or trash; and writes back the decision to the database. The Statistics analyzer and Link Analyzer take into account the feedbacks provided by the DFS while analyzing the previous crawling data so that it can provide more practical suggestions as to improve the performance of the subsequent crawl.

3.1.1 Seed Finder

Finding sites that would most likely contain topical related documents is the main task of Seed Finder and it is vital for the success of a focused Web crawler. CNDROBOT uses two popular Web search engines Google and AltaVista to locate a set of well-selected seed URLs for crawling the Web. We select a specific search engine and use multiple search engines based on the following criteria: the chosen search engine is able to provide us more results (seeds) than the others and by using multiple search engines. We can find a more significant number of unique sites than by using single one.

	AltaVista	**MSN**	**Google**
Max # of Accessible Results / Pages	1,000 / 100	250 / 25	902 / 91
Web Search Results	23,700,000	23,919,807	67,700,000

Table 2 Phrase to Search: computer science department

	AltaVista	MSN	Google
Max # of Accessible Results / Pages	1,000 / 100	250 / 25	875 / 87
Web Search Results	10,600,000	3,408,297	36,100,000

Table 3 Phrase to Search: computer science publications

We manually submitted the query phrases "computer science department" and "computer science publications" to three major search engines: AltaVista, MSN and Google. Returned search result pages are given in Figure A to J in Appendix A. Tables 2 & 3 summarizes some facts found in those figures. We can observe from Figure A and B that AltaVista found 23,700,000 results for the query of "computer science department" and users can access up to 1,000 of them. From Figure C, D, and E, we can see that MSN and Google display to users 250 and 902 results respectively. Accessible results imply the maximum number of seeds that can be accessed by the seed finder. Use of MSN is ruled out because it has much fewer accessible Web pages than the other two. Both Google and AltaVista can be queried through the application and they have a similar number of accessible results. The choice left is to use one or both of them. We also wrote two programs to extract the sites from Google and AltaVista's Web pages (details in section 4.1). Using the keyword phrase "computer science department", Google acquires 892 unique sites and 835 by AltaVista. There are 432 common sites that are included as results by both of them. There are 403 results that AltaVista has but Google does not and 460 results that Google has but AltaVista does not. The result demonstrates that we can obtain more seeds by combining their results than only using one. The seed finder first combines all the results retrieved from each search engine; eliminates the duplicated ones and stores the remaining into NEW_SITES table. The results are stored in the DB table as presented in Table 4.

SID	url
20	www.cis.upenn.edu
21	www.cs.purdue.edu
22	www.cs.unc.edu

Table 4 Sample of NEW_SITES table

Later, the data in NEW_SITES table would be retrieved by the Web Crawler and inserted into the SEED_URL table. These sites combined with some revisiting sites (will be discussed in section 3.2 & 3.4.2) are the seeds that serve as the starting URLs for the CNDROBOT. The following is an example of seeds in the SEED_URL table.

ID	Link	Host_name	Is_new_seed	Resume_flag	SDATE	EDATE	Num_ref_by
21	www.cs.cmu.edu	NULL	1	1	2005-08-02 16:30:42	2005-08-04 04:58:18	NULL
22	www.cs.umd.edu	NULL	1	1	2005-08-04 04:58:18	2005-08-04 20:46:49	NULL
23	www.cs.cornell.edu	NULL	1	1	2005-08-04 20:46:49	2005-08-07 02:07:42	NULL

Table 5 Seeds and the SEED_URL Table

3.1.2 Web Crawler

CINDI web crawling infrastructure is composed of five parts, which are frontier, preliminary filter, page fetcher, HTML parser and classifier. The crawl frontier refers to a list of unvisited pages. As the crawl starts, the frontier head is always a seed URL retrieved from SEED_URL table. The crawler explores the seed's web structure to retrieve new pages by traversing outer links extracted from previously retrieved ones. The preliminary filter has four objectives: Firstly, it excludes the outer links from adding to the frontier if they are already in the frontier or have been visited already. Secondly, the filter is used to avoid irrelevant subdirectories such as "images" and "vita". Thirdly, the filter sorts out the links ending with certain file extensions (such as gif, jpg, mov, avi, and rm), Lastly, the preliminary filter determines if the link is a foreign link by comparing the seed host name with the current page's host name. If the link has the same host name, the preliminary filter will accept the link and insert it into the VISITED_PAGES table, otherwise, the link will be treated as a foreign link and inserted into the FOREIGN_LINK table. To restrict the robot crawling within the domain site, foreign links should not be explored; however they are placed in the FOREIGN_LINK table. Page fetcher is a HTTP client that sends an URL to the remote server, waits for the response and then downloads the page. The page fetcher is different from the file fetcher that downloads any types of files from the URL provided by the Web crawler and stores

them permanently in local repository. The page fetcher only fetches the Web page in HTML format and stores it in a string buffer temporarily for the purpose of quick content parsing and links extraction. Once a page has been fetched, HTML parser will parse its content to extract information such as title, anchor text and outer links that will supply and direct future crawling path. The outer links are first sorted to find the file links that link to Web documents. File links are inserted into PRE_DOWNLOAD_INFO table waiting to be processed and others are added to the frontier if they are not visited before. Classifier processes the fetched page and searches for predefined keywords set such as abstract, introduction, references, and appendix in the page. Then based on the search results, classifier makes a relevance judgment on whether it is one of our targeted document types namely thesis, academic paper, technical report, FAQs in HTML format or other type of Web page whose content does not satisfy our collection goals. The web crawler stores each parsed page in the cache. Only the page that passes the classification will be stored physically in a non-pdf directory for later filtering by DFS and its related information such as URL, file format and size will be inserted into DOWNLOAD_STATUS table for the indexing purpose.

Each crawling cycle involves picking a URL to be searched from the frontier, fetching the page, parsing the page content, filtering links, classifying the Web page and adding outer links to the frontier. The crawling process terminates if the frontier is empty or the download rate (number of downloaded / number of pages visited) is lower than a predefined threshold (detail in section 4.2.3).

3.1.3 File Fetcher

File fetcher uses file links in PRE_DOWNLOAD_INFO table to connect to the remote servers; takes input stream from them; eliminates duplicated files; filters files with undesired file names and sizes; stores downloaded files in local repositories; checks file digital signature; and inserts fetched file information into the database. The file fetcher can be broken down into 5 major components: URL parser, download filter, file writer, digital signature checker, and file information collector. Web pages are actually the files

in various formats (e.g. HTML, pdf, ps, doc, txt, and etc.). From the perspective of file download, the URLs for Web pages are the file locations on the Web. File fetcher first parses the file URL to obtain the directory level at which the file is stored, and then checks if the file has already been downloaded in order to eliminate the possibility of duplicated files being downloaded from the same source. Download filter ensures that files with certain names such as CV, resume and files with small size (file size of 0) will not to be downloaded because the statistics from our previous crawling experiences indicate that these types of files are most likely not be accepted as candidates for CINDI. The experimental results will be shown in Chapter 5. Before writing the input stream to a file, the file name will be checked against the file names in the local directory. Digital signature checker verifies if two downloaded documents with the same file name are identical. Finally, the file fetcher uses the functions in java libraries to collect the information such as file type, length, and last modified date that is sent back by the server once the connection is open.

3.1.4 Statistics Analyzer

Once the seeds crawling, file fetching and document filtering have been done, statistics analyzer starts. Statistics analyzer analyzes the information and data gathered from the previous crawling process as well as the feedback provided by DFS to acquire some knowledge on crawling history and documents downloaded. In addition, it uses the accumulated statistical results to make the robot more selective during the subsequent crawling. Statistics analyzer is a crucial component to optimize the CNDROBOT system. Generally, statistics analyzer performs the following tasks. Firstly, it generates various statistical results (e.g. document download rate, accepted document rate, rejected document rate, etc.) for each crawled site. Then, initializes a seed list for the subsequent crawling. The seed list includes the sites selected from crawled sites and the sites selected from sorted foreign links. The statistics analyzer also performs other functions such as discovering new sites, updating site references to identify the popularity of a crawled site; updating cross-references through a link to a downloaded document; and setting

predefined searching depth level for the subsequent crawl. Details will be covered later in section 4.4.

3.1.5 Link Analyzer

Given a finite amount of crawling time, avoiding irrelevant directories from a given Web site hierarchy can significantly increase the potentially acceptable document download rate, which is defined as the total number of potentially acceptable documents downloaded over the total number of Web pages crawled. The main task of the link analyzer is to identify possible irrelevant directory names based on previous crawling experiences and the filtering results for downloaded documents produced by DFS so that the efficiency of future crawling can be improved. Details will be discussed in section 4.5.

3.2 CNDROBOT Database

CNDROBOT uses MySQL as its database server due to the fact that MySQL has the capability of supporting over 50,000,000 records [MYSQL]. With JDBC interface, developers can easily write applications to access and manipulate data in the database. CNDROBOT, written in Java, manages database tables and accesses data in tables through JDBC API.

As depicted in Figure 4, CNDROBOT creates and operates on 16 database tables.

NEW_SITES table stores the new sites that have never been crawled by the CNDROBOT. Before the initial crawl, the sites in the table are generated by the seed finder. When the crawl starts, the first 30 sites will be retrieved and stored in SEED_URL table. We set the number of the seeds to 30 based on our experiments (see Chapter 5). We estimate that the time to complete 30 sites crawling, file fetching, document filtering, result analyzing a new crawling cycle are less than one month. For sites with a large crawling cycle, the frequency of revisit would be smaller. After each crawling period, the

table will expand with new sites discovered from foreign link hosts, which are found through the crawling process.

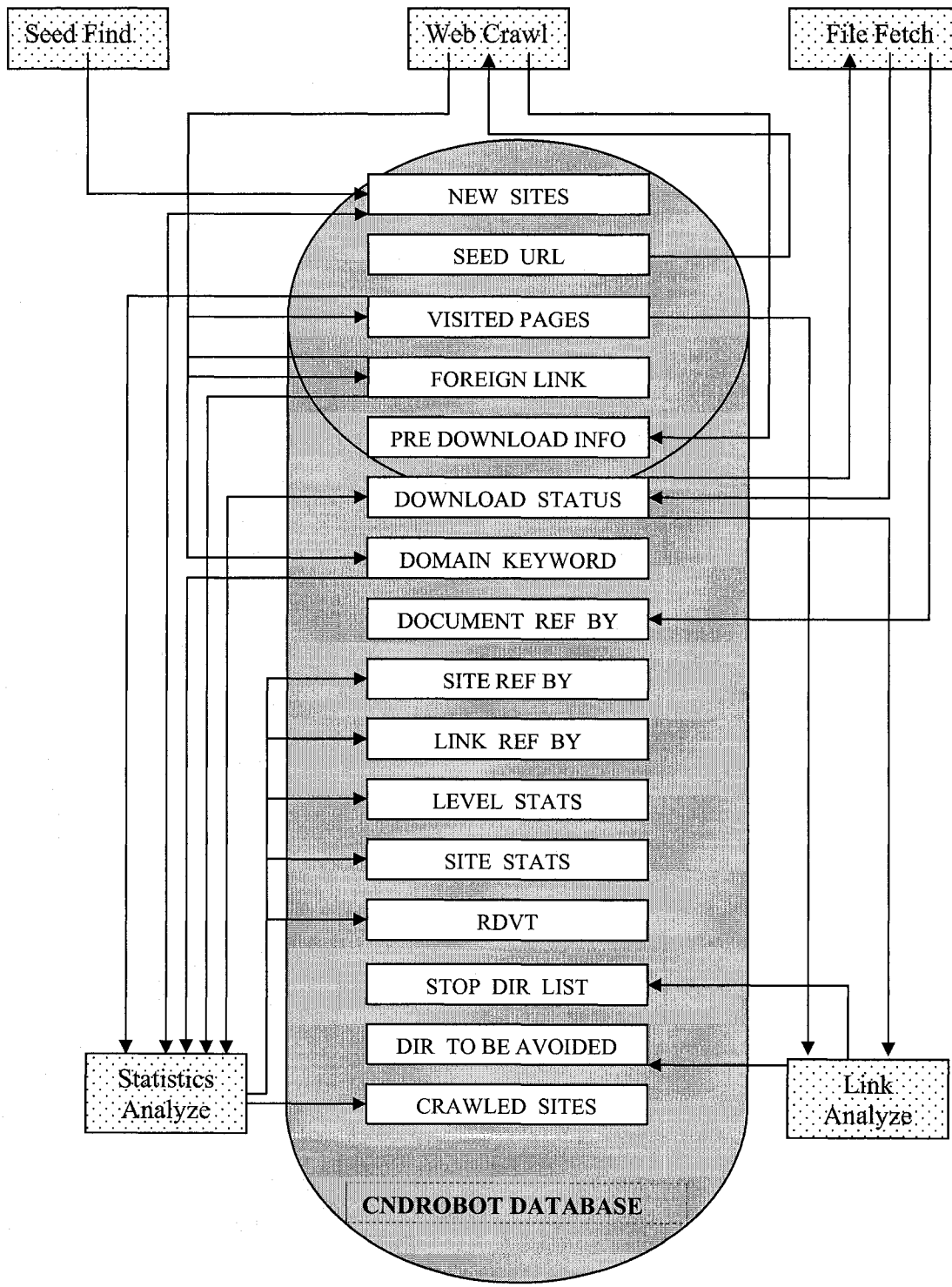

Figure 4 CNDROBOT Database and Interaction with Modules

SEED_URL table stores the seeds used for the current crawling. The seeds usually include the new sites retrieved from the NEW_SITES table and the "already crawled" sites selected from the CRAWLED_SITES table. However, there are two exceptions: In the initial crawl, all the seeds are new sites because no sites have been crawled yet. Later, if there were no data in the NEW_SITES table, all the seeds in the SEED_URL table would be the crawled sites from the CRAWLED_SITES table. The SEED_URL table also keeps a record of start time and finish time for each seed using the attributes of SDATE and EDATE. The crawling time varies significantly from site to site. A huge Web site such as www.w3.org, with more than 150,000 Web pages takes approximately 5 days to finish crawling whereas a relatively small Web site for example www.eecs.berkeley.edu with 11,053 Web pages takes about 6 hours to complete. The crawling time is determined by many factors, for example site size, page size, server network speed and traffic.

Field	Type	Null	Key	Default	Extra
ID	int(200)		PRI	NULL	auto_increment
Link	blob	YES		NULL	
Host_name	varchar(200)	YES		NULL	
Is_new_seed	int(1)	YES		NULL	
Resume_flag	smallint(1)	YES		NULL	
SDATE	datetime	YES		NULL	
EDATE	datetime	YES		NULL	
Num_ref_by	int(100)	YES		NULL	

Table 6 Schema of the SEED_URL Table

The attribute Resume_flag is used, as a guide when resuming the crawling process if the robot terminates abnormally. This attribute Is_new_seed is used to indicate if the seed is an "already crawled" site or a new site. The distinction is imperative because the goal of crawling new sites is to discover new documents whereas the goal of crawling "already crawled" sites is to check for updates, therefore, the crawling algorithm is different for these two types of seeds (see section 4.2). The attribute Num_ref_by keeps the counts of number of times each site is referred through the links in other hosts.

VISITED_PAGES table holds the information for all the visited pages in crawled sites. From the crawl history point of view, the VISITED_PAGES table stamps the path of the crawler as it traverses a seed site. In the course of crawling a seed site, if the page being visited is within the domain of the seed site, the CNDROBOT inserts the page url, name, title, parentID, siteID, is_valid and last modified date into the VISITED_PAGES table (see Table 7). If the host name of the visiting page is different from the page where the link was found, its URL, parent ID and the page's host name will be inserted into FOREIGN_LINK table. The outer links extracted from a Web page are the children of it. When they are inserted into the VISITED_PAGES table, their parent ID is the parent's PID. The starting Web page for each seed is the root for that site and it has the parent ID of 0. By tracing the parent ID of each visited page with the siteID, the Web structure of each seed site can be unveiled.

Field	Type	Null	Key	Default	Extra
PID	bigint(20) unsigned		PRI	NULL	auto_increment
url	blob	YES		NULL	
title	varchar(100)	YES		NULL	
page_name	varchar(100)	YES		NULL	
is_valid	smallint(1)	YES		NULL	
parentID	bigint(20) unsigned	YES		NULL	
siteID	bigint(20) unsigned	YES		NULL	
PDATE	date	YES		NULL	

Table 7 Schema of the VISITED_PAGES table

FOREIGN_LINK table maintains the URLs extracted from the Web page, which are different from the site's host name. The hosts of these foreign links are used to extend the number of sites for the subsequent crawling. From the sample given below, we can see that most of them are good candidates for the new sites, for example, www.geneontology.org and www.ccs.neu.edu. However, some are not because they are off the topic, for example, www.cciw.ca and www.genisis.ch. The approach to sort out new sites from foreign links will be discussed in section 4.4.

FID	url	host_name	PID
9001	http://cermm-s.concordia.ca/abstracts-2005.pdf	cermm-s.concordia.ca	23635
9002	http://www.cs.utoronto.ca/%7Ejuris/cccb04.htm	www.cs.utoronto.ca	23635
9003	http://www.cciw.ca/wqrjc/38-2/38-2-227.htm	www.cciw.ca	23635
9004	http://www.cciw.ca/wqrjc/wqrjce.htm	www.cciw.ca	23635
9005	http://www.geneontology.org/	www.geneontology.org	23650
9006	http://imgproj.cs.man.ac.uk/tambis/	imgproj.cs.man.ac.uk	23650
9007	http://www.ariadnegenomics.com/technology/ontology.html	www.ariadnegenomics.com	23650
9008	http://www.biowisdom.com	www.biowisdom.com	23650
9009	http://dags.stanford.edu/PRMs/	dags.stanford.edu	23651
9010	http://robotics.stanford.edu/%7Ekoller/papers/ijcai99lprm.html	robotics.stanford.edu	23651
9011	http://www.cs.umd.edu/%7Egetoor/talks.html	www.cs.umd.edu	23651
9012	http://robotics.stanford.edu/%7Eerans/	robotics.stanford.edu	23651
9013	http://www.cs.huji.ac.il/%7Enir/	www.cs.huji.ac.il	23651
9014	http://www.stanford.edu/%7Egrenager/prm_tutorial_2003_02_20.ppt	www.stanford.edu	23651
9015	http://web.engr.oregonstate.edu/%7Etgd/classes/539/	web.engr.oregonstate.edu	23651
9016	http://blimp.cs.queensu.ca/	blimp.cs.queensu.ca	23652
9017	http://www.ccs.neu.edu/home/futrelle/bionlp/papers/HirschmanReview1202.html	www.ccs.neu.edu	23652
9018	http://textomy.iit.nrc.ca/cgi-bin/BNLPB_ix.cgi	textomy.iit.nrc.ca	23652
9019	http://www.genisis.ch/~natlang/NLPBA02/IJMIToc.html	www.genisis.ch	23652

Table 8 Sample of the FOREIGN_LINK Table

Once the web crawler finds a URL ending with txt, pdf, ps, doc, xml, rtf, tex or latex from a Web page, the URL and the Web page ID in the VISITED_PAGES table are kept temporarily in the PRE_DOWNLOAD_INFO table and wait for the File Fetcher to download. File Fetcher starts to operate either automatically when one site searching has been complete or manually when we trigger it. As long as there is a record in the PRE_DOWNLOAD_INFO table, File Fetcher retrieves the URL from the table and uses it to download the file to the local repository. In the meantime, file related information such as URL, original file name, system file name, temporary location, file type, file size, location level, is_different_format, is_renamed, and last modified date are inserted into the DOWNLOAD_STATUS table. The schema of the DOWNLOAD_STATUS table is shown in Table 9.

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	

Table 9 Schema of the DOWNLOAD_STATUS Table

In order to facilitate other CINDI subsystems' work, we purposely divide a URL address into two parts: prefix URL and file name. Suppose that a file URL http://www.speech.cs.cmu.edu/air/papers/speechwear.ps is extracted from the Web page http://www.speech.cs.cmu.edu/air/papers.html. It will be stored into table by its prefix URL http://www.speech.cs.cmu.edu/air/papers/ and its file name speechwear.ps. Since it is a non-pdf file, this file will be first stored in a temporary location, the directory "/download/". DFS is responsible for setting the attribute final_location and updates the filter_flag to indicate whether this document is accepted or not. If another file with the same file name is already in the directory e.g. "/downloads/", the file name must be changed to prevent the existing file being overwritten. The values for the attributes orig_file_name and file_name would be the same if the file name were not changed. The attribute is_renamed is set to 1 if the file was renamed; otherwise, it is set to 2. If this page contains an URL, for example http://www.speech.cs.cmu.edu/air/papers/speechwear.pdf, it means that the file is the same file in different format because it has the same trimmed file name and prefix URL. This file is still downloaded but its attribute is_diff_format is set to 1 to indicate that the file has different formats than the existing file. The attribute level is the directory level

where the file is located and it is equal to 3 in the case above. The attribute parentID is the identification number of the Web page from which the URL for the file is extracted and represents the relationship between a downloaded file and a Web page. The attribute num_ref_by records the number of times that the document is referred in other Web sites. The attribute ashg_flag is used by ASHG to indicate that a semantic header has been generated for this accepted document.

Statistics analyzer analyzes the data retrieved from SEED_URL, VISITED_PAGE, DOWNLOAD_STATUS, and FOREIGN_LINK tables and then inserts and updates the results into SITE_STATS, LEVEL_STATS, LINK_REF_BY, SITE_REF_BY, RDVT, STOP_DIR_LIST, DIR_TO_BE_AVOIDED, and CRAWLED_SITES tables.

The site ID and details about it such as total number of visited Web pages, total number of downloaded documents, total number of accepted documents, total number of rejected documents, download rate, accepted document rate, rejected document rate, and time to complete for each site are kept in SITE_STATS table. The values in these attributes indicate the quality of the site. These are important parameters for the CNDROBOT when selecting new sites for recrawling. For example, a site with higher download rate and accepted document rate should have higher priority for recrawling than the one with lower rates because crawling a site with higher download and acceptance rates is more cost effective and the crawler would likely discover new documents in that site. A sample of SITE_STATS table is presented in Table 10.

SID	t_pages	t_downloaded	t_accepted	t_rejected	p_downloaded	p_accepted	p_rejected	time_period
1	18468	3437	665	2772	18	19	80	1201
2	15045	2042	137	1905	13	6	93	516
3	39679	3794	674	3120	9	17	82	3763

Table 10 Sample of the SITE_STATS Table

LEVEL_STATS table maintains the statistic results for directory levels where the documents are located. The results are used to set the maximum search depth level. For example, we can see in Table 12 that most of the documents are located at directory level

of 2,3,4 and 5 and there are no documents below level 7. Therefore, we can control the search depth in the future crawl by tuning the robot to not visit any URL whose level is greater than 7.

level	num_of_document	percentage
7	95	0
6	3537	3
5	25121	24
4	33911	32
3	32703	31
2	10789	10
1	260	0

Table 11 Sample of the LEVEL_STATS Table

LINK_REF_BY table maintains the records of link cross references for downloaded documents. For example, suppose that the document cindi.pdf is discovered while crawling the site www.cs.concordia.ca and later it is downloaded from the URL http://www.cs.concordia.ca/~bcdesai/cindi.pdf. As the robot crawls another site www.umc.edu, the same URL for the file is found as a foreign link while parsing the Web page http://www.umc.edu/paper/2005/. It means that the file is referred once because the author of the Web page in another host recommends this file by providing the link of the file in the page. To keep a record for this type of referencing, the file ID in DOWNLOAD_STATUS table and the link ID in FOREIGN_LINK will be inserted into LINK_REF_BY table.

SITE_REF_BY table contains three attributes and it has the table structure given in Table 12:

Field	Type	Null	Key	Default	Extra
ID	bigint(20) unsigned		PRI	NULL	auto_increment
SID	bigint(20) unsigned	YES		NULL	
FID	bigint(20) unsigned	YES		NULL	

Table 12 Schema of the SITE_REF_BY Table

The SID is the ID for the site in SEED_URL table and FID is the identification number for the foreign link in FOREIGN_LINK table. SITE_REF_BY table represents the relationship between the site in SEED_URL table and the foreign link and its host in FOREIGN_LINK table and indicates the popularity of the crawled site. Assume that the site www.unc.edu with ID of 22 in SEED_URL table is referred by the foreign link http://www.cs.umd.edu/%7Egetoor/talks.html with FID of 9011, the ID of the host www.cs.umd.edu, one of the crawled sites. Therefore, an entry with SID of 22 and FID of 9011 will be inserted into SITE_REF_BY table. The number of occurrences for each SID is the number of times that site is referred by other hosts.

Representative Document Vector Table (RDVT) table contains the anchor texts that appear most frequently in the home pages of sample seeds, which are the sites in the SEED_URL table. These pages are also referred to as exemplary documents. These anchor texts are used to compare the similarity between a Web page and the exemplary documents. The table is used by statistics analyzer to select new sites from the foreign link hosts. Since foreign links were extracted from subpages in those sites, we expect that newly selected sites should at least have some characteristics of sample seeds. A sample of the table is presented in Table 13:

anchortext	count
people	8
publications	7
research	7

Table 13 Sample of the RDVT Table

The attribute count is the number of exemplary documents that contain the anchor text. Only the anchor text with a count greater than a quarter of total number of exemplary documents is stored in RDVT table.

The keywords in DOMAIN_KEYWORD table are extracted from subject classification of INSPEC (the Database for Physics, Electronics and Computing) [INSP] [XUE03].

These 174 keywords combined with sample anchor texts in the RDVT table are the measurement of selecting new seeds from the FOREIGN_LINK table (see section 4.4). Newly selected seeds are inserted into the NEW_SITES table.

CRAWLED_SITES table contains the information for the already visited sites. The CRAWLED_SITES table is the master table of the SEED_URL table and it has a table structure similar to the SEED_URL table as shown in Table 14.

Field	Type	Null	Key	Default	Extra
ID	int(200)		PRI	NULL	auto_increment
Link	blob	YES		NULL	
Host_name	varchar(200)	YES		NULL	
Priority_level	int(10)	YES		NULL	
SDATE	datetime	YES		NULL	
EDATE	datetime	YES		NULL	
Num_ref_by	int(100)	YES		NULL	

Table 14 Schema of the CRAWLED_SITES Table

In the CRAWLED_SITES table, we replace the attribute Is_new_seed used in the SEED_URL table with a new attribute Priority_level. The attribute Priority_level represents the minimum waiting time for the site before the next crawl. It has five values, corresponding to different time periods as shown in Table 15. If the site has never been crawled before, its priority level is set to 1 to indicate that it is eligible to be crawled 15 days after its last crawling end time. If the site were recrawled and found that the accepted document rate decreased (see section 3.3.5), its priority level would be degraded one level down, which means this time it has to wait at least 30 days to be recrawled. Once the priority level reaches 5, the site will be discarded. On the other hand, if the accepted document rate of the site was found to increase, the site's priority level will be promoted one level up. However, if the site's priority level is already 1, then the same level is kept. Once sites crawling, file fetching and statistics analyzing is completed, the seeds in the table will be merged with SEED_URL table. The information about the "first-time crawl" sites: link, host name, starting time and finish time is inserted into the CRAWLED_SITES table and their attribute Priority_level is set to 1 to indicate that they

are eligible for the next crawl as long as they have waited enough time. The time for a site has waited is calculated as follows:

$$wt = cd - ed$$

wt*:** waited time ***cd: current date ***ed:*** finish date of last crawl for the site

A crawled site has waited enough time if its wt is greater than the time value corresponding to its priority level.

Since "already crawled" sites have entries in the CRAWLED_SITES table, the attributes SDATE and EDATE will be updated with the new times and their attribute priority levels will be updated according to the new value for the accepted document rate. After merging, the SEED_URL table is empty and ready for accepting new seeds.

The attribute Num_ref_by in the CRAWLED_SITES table has the same indication as the one in SEED_URL table. It will be updated as more references to the sites are found.

Priority Level	**Time Waiting Period**
1	15 days
2	30 days (1 month)
3	60 days (2 months)
4	180 days (6 months)
5	infinite

Table 15 Significance of Priority Level

The Link Analyzer manipulates the data in the DOWNLOAD_STATUS and VISITED_PAGES tables (see section 4.5) and records the analyzed information into STOP_DIR_LIST and DIR_TO_BE_AVOIDED tables. The STOP_DIR_LIST table holds the directory names under which no downloaded documents are found to be acceptable. It is most unlikely to confirm the genre of documents under them and the directories names (in Table 16) are used to predict the relevance of a URL while crawling the site.

dir_name	num_of_invalids	num_of_downloads	percentage
images	36	36	100
inlp2004a	25	25	100
lectureNotesWeb	76	76	100
msi	16	16	100
pdf-6up	42	42	100
pdf-color	69	69	100
puzzle	18	18	100
review	37	37	100
sections	20	20	100
transparencies	33	33	100

Table 16 Sample of the STOP_DIR_LIST Table

In Table 16, the attribute num_of_invalids denotes the total number of rejected documents whereas the attribute num_of_downloads represents the total number of downloaded documents from the directory with the given name. The attribute percentage represents the rejected percentage of downloaded documents.

DIR_TO_BE_AVOIDED table maintains the site ID, directory name, and URL for the directories under which no document can be found in the crawled sites.

siteID	host	dir_name	url
3	www.cs.indiana.edu	Contacts	http://www.cs.indiana.edu/Contacts/
3	www.cs.indiana.edu	Courses	http://www.cs.indiana.edu/Courses/
3	www.cs.indiana.edu	Academics	http://www.cs.indiana.edu/Academics/
3	www.cs.indiana.edu	Research	http://www.cs.indiana.edu/Research/
3	www.cs.indiana.edu	People	http://www.cs.indiana.edu/People/
3	www.cs.indiana.edu	Calendar	http://www.cs.indiana.edu/Calendar/
3	www.cs.indiana.edu	Resources	http://www.cs.indiana.edu/Resources/
3	www.cs.indiana.edu	Facilities	http://www.cs.indiana.edu/Facilities/

Table 17 Sample of DIR_TO_BE_AVOIDED Table

For example, in Table 17, CNDROBOT could not find any relevant document under the directories of "Contacts", "Courses", and "Academics" and their subdirectories. The rationale and approach of doing this will be discussed in the next chapter.

3.3 Approach

Extensive research and experiments have been made on various Web search strategies, for example, breadth-first search [MN01], best-first search [JC98], shark search [MH98], focused crawling [SB99], reinforcement learning [MA99] and page rank [PR99].

Because one strategy does not fit all situations, search engine robots adopt different search algorithms depending upon their crawling purposes and judgments. Mercator, a Web crawler used by AltaVista, employs the breadth-first search algorithm since they found that this algorithm is able to discover the highest quality pages during the early stages of crawling [MN01]. GoogleBot utilizes best-first search with page rank metrics to crawl the Web [PR99]. They count the back links to a page to produce the rank, the popularity of the page. They believe that pages with more reference count have higher quality and therefore higher crawling priority.

CNDROBOT focuses on exploring the Web pages inside the seed sites, using the breadth-first search algorithm to find computer science literature. We assume that all the pages inside the seed site are relevant to our goals and all the documents contained in the pages are potentially desired document. The breadth-first algorithm places unvisited URLs in the frontier in the order in which they are discovered. They are retrieved from the queue in first-in, first-out fashion. The reason we select breadth-first as our crawling strategy is because it has the advantage of being easy to implement and has proved able to find more high quality Web pages at the early crawling phases [MN01].

The CNDROBOT is designed to run infinitely and attempts to cover as many potential sites as it can. To achieve this, the web crawler continuously discovers new potential seed sites in the course of crawling and the statistical analyzer supplies new selected seeds to the web crawler. Details of crawling approach, new seed selection rules and seed reschedule algorithm will be given in this section.

3.3.1 Overview of Crawling Approach

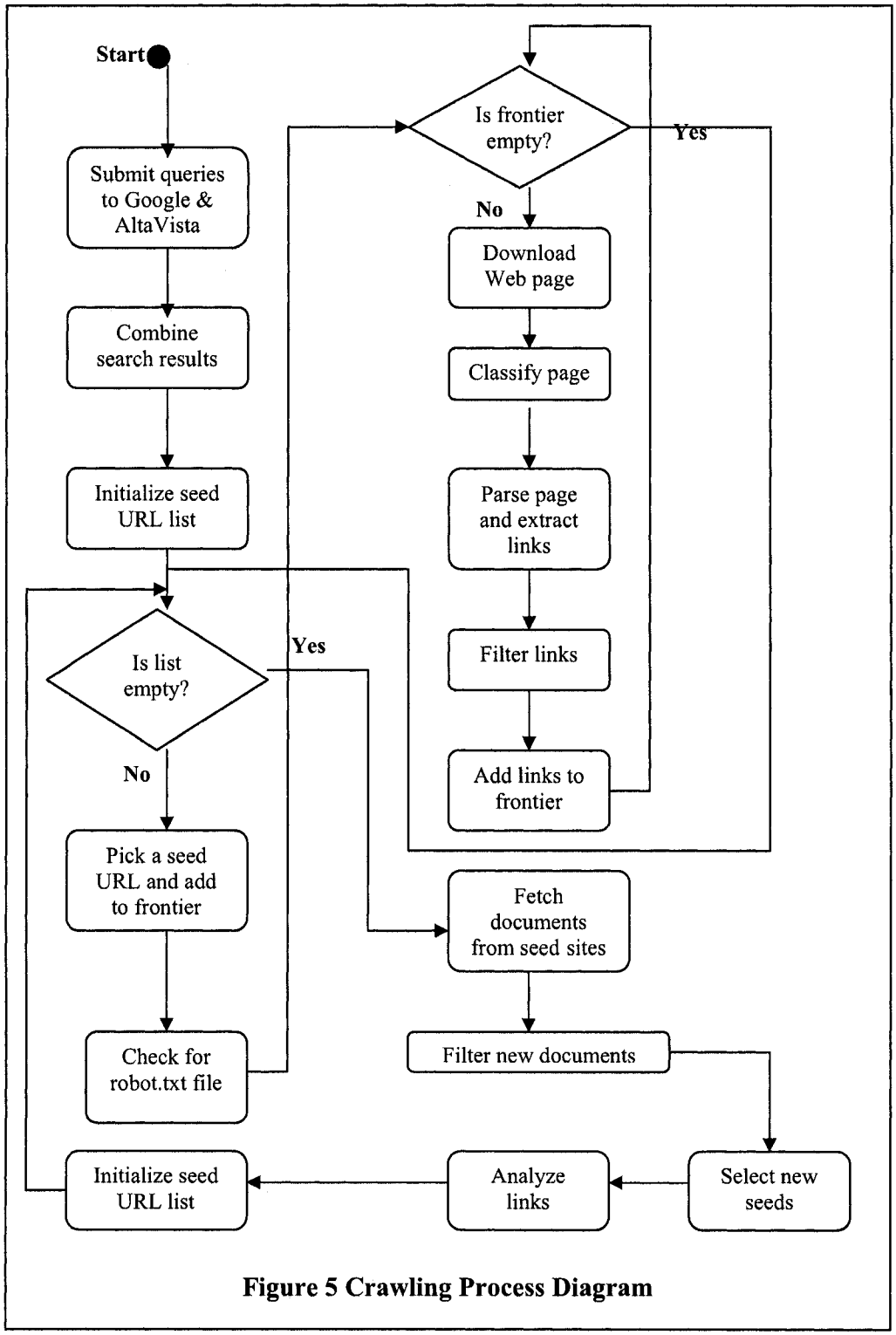

Figure 5 Crawling Process Diagram

Figure 5 shows the basic crawling process of the CNDROBOT, it starts with submitting queries ("computer science department", "computer science publications" and "computer science technical reports") to Google and AltaVista search engines. The combined results from both search engines are stored into the NEW_SITES table and 30 sites from the table are used to initialize the seed URLs list. The CNDROBOT adopts breadth-first search algorithm while traversing the seed site to locate the documents. Crawling frontier maintains a list of unvisited URLs, which are extracted from previously crawled pages. A crawling loop always involves 6 steps: 1) Check if the frontier is empty 2) Download Web page 3) Classify page 4) Parse page and Extract links 5) Filter the URLs 6) Add URLs to the frontier. When there are no URLs in frontier, the crawling loop stops. It indicates crawling for a particular seed site is complete. The next crawling cycle starts by picking the next available seed URL from the seed list. The current crawling process stops when the seed list is empty, i.e., all the seed sites initialized previously have been crawled. At this time, the File Fetcher commences to download the discovered documents from crawled seed sites.

After the file fetching process is complete, DFS starts to check for newly downloaded documents and evaluates the quality of these documents. Once the DFS has completed filtering all the downloaded documents, statistics analyzer starts to analyze the crawling history data and document information; selects revisiting sites from "already crawled" sites; and digs out new sites from foreign link hosts and inserts them into the NEW_SITES table. It also forms the seeds list for the next crawl, which includes the new sites and revisiting sites. Then, the crawl restarts once the new seed URL list is ready, i.e., SEED_URL table is not empty and the link analyzer has created the DIR_TO_BE_AVOIDED and DIR_STOP_LIST tables.

3.3.2 The Standard for Robot Exclusion

The *Standard for Robot Exclusion* [KM94] was proposed to restrict crawlers from accessing certain parts of a site. The standard involves creating a plain-text file named robots.txt and placing it at the root of a web site.

The robots.txt file can be used to advise robots not to index an entire site, specific directory or its subdirectories, or even a particular file. For example, the following entry is part of a robot.txt file at IBM's Web Site. (http://www.ibm.com/robots.txt, see Appendix B). It specifies that robots or crawlers be forbidden to access the directory of http://www.ibm.com/Admin.

```
User-agent: *
Disallow: /Admin
```

This standard has gained widespread acceptance, and basically all search engines abide by it. Although complying with the standard is voluntary, the CNDROBOT respects it by first checking the existence of the robots.txt file at the root of the seed site before starting to crawl the site. If the robot.txt file exists, all the disallowed path entries will be extracted to the robot safe checklist. The outer links extracted from Web pages will be examined against the list before being added to the crawling frontier.

3.3.3 Parse the HTML Document

Generally there are four kinds of hyperlinks in HTML documents: anchor (<A>) tags; Image (<IMG>) tags; Map and Area tags; and Frame and iFrame tags [NL00]. Anchor tags are the most commonly used. They have several attributes including *name*, *title*, *alt*, *on-mouse-over,* and *href.* When CNDROBOT parses a Web page, it determines the title, hyperlinks in the page, and anchor texts that describe the hyperlinks. HTML standard requires the author of a Web page to set the page title between the tag <title> and </title> in the <head> section. The value of a hyperlink pointing to another page is normally found within the tag and the description for the link can be found right after the hyperlink tag and before the tag . Because the goal of CNDROBOT is to discover and collect scientific literature from the Web, we ignore the information contained in Image, Map and Frame tags.

3.3.3.1 Parse AltaVista Web Page

In AltaVista's Web result pages, all the hyperlinks for results in the display are situated after the tag . The class defined in Cascading Style Sheets (CSS) adds the style (e.g. fonts, colors, spacing) to a Web document. Since AltaVista is a commercial search engine, it includes sponsored matches at the top as well as at the bottom of each page (see the highlights in Appendix D). We exclude those commercial sites from the seed list since they are spams, which contain little information related to our search topic. This raises the problem of identifying sites from the sponsored ones. To solve this problem, CINDI seed extractor parses the result pages twice. We notice that for unsponsored URLs in the results, AltaVista offers "More pages ...". Sponsored links do not have such a feature. In the first parsing, we extract all the hyperlinks between and including sponsored links, for example, the sponsored link www.nextag.com and the link www.indiana.edu (see Figure 6) and insert them into a temporary database table. In the second parsing, we first extract the hyperlinks right after the string "More pages from" and then compare this trimmed URL "cs.indiana.edu" with the URL www.cs.indiana.edu in the temporary table. If there is a similar entry in the table, we know that www.cs.indiana.edu is a real seed and it will be inserted into the NEW_SITES table. In this way, we can screen out commercial hyperlinks. One could ask why we do not just make the second parse and affix "www" to the trimmed URL. The reason is that it works for most cases but not all. For example, the URL for the department of computer science at Australian National University is http://cs.anu.edu.au/, however, the host name www.cs.anu.edu.au/ is invalid.

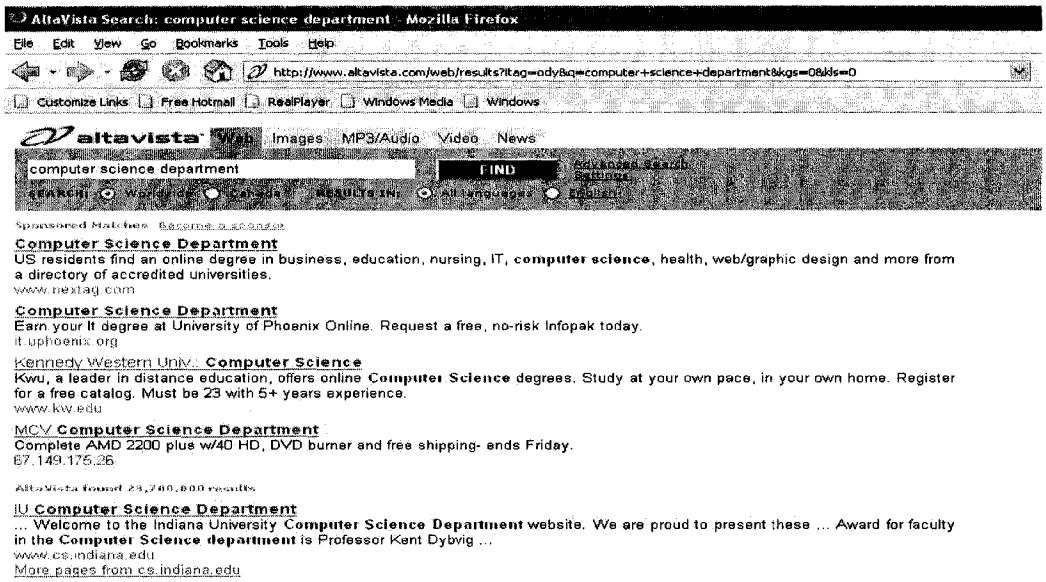

Figure 6 AltaVista Search Result Page

3.3.3.2 Convert Relative URL to Absolute URL

CNDROBOT involves retrieving thousands of Web pages and interpreting tens of thousands of URLs in these pages every hour. In order to obtain precise URLs, one must pay special attention to resolve the various forms of expression of hyperlinks on a page. A hyperlink that specifies the location of a file/directory on the network can be presented in two forms, either in absolute URL or in relative URL. Absolute URL is a complete URL, which gives a full path name to the target file, for example: a document latex.html is retrieved using http://www.cs.concordia.ca/help/latex.html. For the base URL http://www.cs.concordia.ca/, a relative URL can be given as /help/latex.html or help/latex.html. Generally, the absolute URL is a concatenation of the base URL and the relative URL. The forms of the relative URL can be summarized in Table 18.

Relative URL Syntax	Refers to
./	Current directory
../	Parent directory
../../../	Up to two levels of directory
#foo	Fragment identifier in a Web page
foo.html	Concatenation to the base URL
/foo.html	Concatenation to the host URL
//www.xyz.com	Replacing everything from the host URL

Table 18 Interpretation of Relative URL Symbols

3.3.3.3 Meta Tags

There are several meta tags defined by search engines for indexing the Web pages. CNDROBOT is only interested in two of them, i.e. "meta keywords" and "meta http-equiv="refresh"". The "meta keywords" tag is used to describe the content of a Web page using one or more keywords. For example, the "meta keywords" used to describe Sun Microsystems's home page is <meta name="keywords" content="Java, platform" />. CNDROBOT can examine the relevance of the Web page search topic (computer science) by comparing keywords extracted from the tag with the domain keywords in the DOMAIN_KEYWORD table. If no keywords are given or keywords are not matched with ones in the table, the page content and anchor texts around the links are examined to determine the relevance of the page (see chapter 4).

"meta http-equiv="refresh"" is not a standard meta tag; it sends a HTTP signal to the browser or search engine robot to reload the current page or redirect it to a new page. By downloading the home page of a well-known scientific publication Web site (www.elsevier.com), its content is determined as follows:

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Refresh" CONTENT="0;URL=/wps/find/homepage.cws_home">
</HEAD>
</HTML>
```

The value 0 in attribute CONTENT is the number of seconds the robot waits before loading the specified URL. As mentioned before, the HTML parser only extracts the values in link tag <a href = "..."> . Without checking for "meta http-equiv="refresh"", CNDROBOT has no way to determine the new home page location given as www.elsevier.com/wps/find/homepage.cws_home and hence CNDROBOT might miss an entire site. The redirected page could be the home page of a site as in the example above or a Web page containing the links to documents. In either case, the cost of missing them is unnecessary and avoidable. Hence, searching for http-equiv meta tag and finding the redirected page is a mandatory task for CNDROBOT while parsing a Web page.

3.3.3.4 Enforce Canonicalization Rules and Error Correction

Since HTML is a semi-structured language and a HTML page does not need to be compiled, many unpredictable human mistakes in Web pages prevent CNDROBOT from locating correct links or can even result in run time errors. In addition, the nature of the HTML language itself can sometimes also hinder the crawling process. Applying canonicalization rules and implementing the error protection mechanism can significantly reduce the probability of occurrence of these problems, improve the crawling efficiency and avoid an endless crawling loop. The canonicalization rules are described below.

Canonicalization Rules

Gautam Pant [GP04] presented some canonicalization procedures for a typical URL, such as converting the URL string to lower case; considering "index.html" as the default Web page which can be retrieved using the base URL, therefore removing the URL with this page name from the crawling list; and performing URL encoding on some special characters, for example, encoding "?" to "%3f". In implementing canonicalization rules for CNDROBOT, we used Pant's rules as guidelines to improve the robot's performance. Here, we give three examples of adpated rules to illustrate the importance of applying canonical forms rules:

Remove ".." and its parent directory

The symbol of ".." indicates one level up from the current directory. It might appear in an URL, for example http://www.cis.upenn.edu/~rwash/bike/../papers/mjw.html and http://www.cs.cmu.edu/~help/windows/../security/viruses.html. The same Web page can be retrieved if the ".." and the parent directory are removed. Hence, these URLs can be reduced to http://www.cis.upenn.edu/~rwash/papers/mjw.html and http://www.cs.cmu.edu/~help/security/viruses.html

Remove fragment indicator "#"

The symbol "#" in the URL string is used to link to a target location in a HTML document. In the page http://www.cs.concordia.ca/help/help.html, there are 7 URLs ending with "#"s and their targets. For example: http://www.cs.concordia.ca/help/help.html#FAQS and http://www.cs.concordia.ca/help/help.html#HOWTOS. Comparing them literally, they are different URLs. However, they are pointing to exactly the same Web page. Without applying this canonicalization rule, this page would be visited, downloaded, and parsed 8 times. By removing "#" and its target, a single base URL is discovered. This prevents the same page from being crawled more than once.

Avert Spider Trap

A spider trap happens when a Web page is crawled repeatedly and endlessly. For example in page http://www.st-andrews.ac.uk/foi/login_form/enabling_cookies/, there is a relative URL <a href="enabling_cookies">. In most cases, CNDROBOT simply concatenates this type of relative URL to the base URL and then a "new" URL http://www.st-andrews.ac.uk/foi/login_form/enabling_cookies/enabling_cookies is formed and inserted into the queue waiting for its turn to be crawled. When there are no URLs in the queue ahead of it and all other URLs on this page have been visited, it would

be the only URL in the queue to be crawled. Because the "new" URL is still directed to the same page and the same relative URL is extracted, there will be another "new" URL http://www.st-andrews.ac.uk/foi/login_form/enabling_cookies/enabling_cookies/enabling_cookiess/enabling_cookies created. Notice that no matter how many "enabling_cookies" are appended, they still point to the same page. The crawling progress will be stalled at this point because of this endless loop.

There are actually two techniques used by CNDROBOT to avert this spider trap. The first technique is to use a "predefined depth level", which is not designed for that purpose in the first place but has the functionality of stopping the trap when it happens. "Predefined depth level" limits the levels in a Web structure to which CNDROBOT will search, at the same time; it also confines the ability of the robot to expand the URL to a certain level so that the trapping can be discontinued. Details of this technique will be covered in Chapter 4.

The second technique is to compare the current URL's page name with the relative URLs to be extracted. The page name is the character string after the last "/" if it's not empty. For example, we define the page name for URL http://www.cdf.toronto.edu/workathome/index.php3 as index.php3. If the URL ends with "/", the page name is the characters between the last two slashes, which is still index.php3. If the robot finds a link tag such as <a href = "index.php3">, it knows that this relative URL is the same as the current page name and will ignore it for extraction.

Error Correction

Human mistakes can also cause severe problems for the robot. The following is a list of problems encountered by CNDROBOT while parsing HTML pages.

Use of Slash in URL

A standard URL includes slashes "/" to refer to the directory. Some web page authors mistakenly use back slash "\" instead of "/" in the hyperlink. For example: when parsing the page http://www2.cs.uh.edu/~vtbui/, the link <a href= "http:\\www.cs.uh.edu\~vtbui\whoami.htm"> causes the robot a problem

Unconventional Relative URL

In page http://www.cs.concordia.ca/~cliff/dai/dai-home-page.html, the author uses <A href="http:dai-list.html"> to represent a relative URL to the file "dai-list.html".

Miscellaneous Errors

Some authors prefer using single quotation marks instead of double ones for URLs. Typing errors such as one slash "/" instead of two slashes "//" after "http:" occur very often.

The above-mentioned problems are resolved by the auto-correction mechanism in CINIDI robot. For instance, to handle the first case, before adding to the uncrawled list, all the links are checked for compliance to canonicalization forms, and if back slashes in the URL were found, they would be replaced by slashes.

3.3.4 Seeds Rescheduling Algorithm

After the initial crawl, seed sites need to be recrawled periodically to check for updates and for new documents posted in these sites. However, given the limited time and the time needed to spend on crawling new sites, only a fraction of the previous seed sites can be placed in the seed URLs list for the next crawl. We apply the Round-Robin algorithm to schedule revisiting orders for "already crawled" seeds as well as new seeds.

The Round-Robin Algorithm is one of the most popular scheduling algorithms, designed originally for process scheduling in operating systems [AS92]. All the processes are placed in a queue. A new arriving process can be placed in the queue in different ways. However each process is only allocated a slice of processor time, called quantum. By using this algorithm, the average waiting time for each process is significantly reduced. CNDROBOT can be viewed as the processor and the seeds as the processes. The uncrawled sites and new sites discovered from previous crawling have priority; therefore they will be placed at the head of the next seed URLs list. We use the mean completion time of the last crawl as the quantum and calculate it using the formula as follows:

$$\textit{Mean Completion Time (MCT)} = \frac{\sum \textit{Time for Seed Crawling to Complete}}{\textit{Number of Completed Seeds}}$$

Before the initial crawl, the seed finder extracts a total of 3,130 unique seeds and stores them into the NEW_SITES table. In the first crawl, we crawled 30 new sites; downloaded 106,416 documents from them; filtered 51,215 documents and moved the rest of documents to the trash directory. The crawling period including document downloading took 26,672 minutes, which is approximately 18 days. The MCT can be calculated as 26,672 / 30 = 889 minutes = 15 hours.

Assume that each seed site would get 15 hours of time units (one sub-cycle), and then ideally each site would wait no longer than 15*30 time units before its next quantum. However, our principle for the subsequent crawling is to allocate half of the total time for new sites while still maintaining approximately the same completion time as the previous crawl. Therefore, in the case above, there are 15 quanta for the new sites and another 15 quanta left for these 30 crawled sites. 15 new sites are taken from the NEW_SITES table, which has a minimum of 3,130 – 30 = 3,100 uncrawled sites.

Because the number of crawled sites (30) is greater than the number of places (15) available, these crawled sites have to compete for the places. To be eligible for competing, the priority levels for those sites are checked to ensure they have waited long

enough for revisiting. From Table 19, we can see that the last seed finished on July 12. Suppose that the next crawl will start on July 22. Using the priority and time waiting period translation sheet in Table 12, the last three seeds with ID of 28, 29 and 30 are ineligible for competing because it has not waited more than 15 days since its crawling was completed. It will become eligible only after the next crawl finishes.

ID	Link	SDATE	EDATE
1	www.cs.umass.edu	2005-05-14 16:32:41	2005-05-15 12:33:58
2	www.cs.concordia.ca	2005-06-01 14:46:14	2005-06-01 23:23:08
3	www.cs.indiana.edu	2005-06-03 01:57:10	2005-06-05 16:40:24
4	www-cs.stanford.edu	2005-06-07 15:20:03	2005-06-07 15:31:28
5	www.cs.cornell.edu	2005-06-08 02:36:26	2005-06-09 09:54:34
6	www.cs.cmu.edu	2005-06-09 14:15:37	2005-06-11 01:34:23
7	www.cs.umd.edu	2005-06-11 15:50:36	2005-06-13 03:04:42
8	www.cs.uiuc.edu	2005-06-13 11:45:37	2005-06-13 11:54:26
9	www.cis.upenn.edu	2005-06-18 22:55:38	2005-06-19 18:42:12
10	www.cs.purdue.edu	2005-06-20 00:50:39	2005-06-20 10:35:42
11	www.cs.unc.edu	2005-06-20 12:15:38	2005-06-21 11:40:45
12	www.cs.toronto.edu/DCS/index.html	2005-06-22 14:15:39	2005-06-23 00:23:18
13	www.cs.columbia.edu	2005-06-23 13:46:38	2005-06-24 04:09:36
14	liinwww.ira.uka.de/bibliography	2005-06-28 15:31:58	2005-06-29 05:14:16
15	dmoz.org/Computers/Computer_Science/Publications	2005-06-29 15:04:17	2005-06-30 12:56:34
16	www.elsevier.com	2005-07-01 03:05:24	2005-07-01 20:54:16
17	www.computer.org	2005-07-01 20:59:59	2005-07-02 05:58:20
18	www.sciencedirect.com	2005-07-02 15:58:37	2005-07-02 16:09:52
19	www.cs.virginia.edu/studpubs	2005-07-02 16:39:28	2005-07-02 22:05:57
20	www.cs.kent.ac.uk	2005-07-02 23:51:40	2005-07-03 11:15:57
21	cjtcs.cs.uchicago.edu	2005-07-03 16:33:41	2005-07-03 16:38:08
22	www.dmtcs.org	2005-07-03 16:38:08	2005-07-03 16:51:16
23	www.mcs.vuw.ac.nz/comp/Publications/	2005-07-04 00:09:33	2005-07-04 14:04:32
24	reports-archive.adm.cs.cmu.edu/cs.html	2005-07-05 03:26:40	2005-07-05 04:08:22
25	www.cs.bu.edu/techreports/	2005-07-05 13:50:41	2005-07-05 18:26:44
26	cs.anu.edu.au/techreports/	2005-07-05 18:26:44	2005-07-07 07:55:26
27	www.cs.arizona.edu/research/reports.html	2005-07-07 07:55:26	2005-07-07 22:53:51
28	www.cs.rpi.edu/research/tr.html	2005-07-11 16:50:41	2005-07-11 20:38:07
29	www.cs.umich.edu	2005-07-11 20:38:07	2005-07-11 20:40:39
30	www.lib.utk.edu/refs/computersci	2005-07-11 20:40:39	2005-07-12 21:25:24

Table 19 Crawled Seeds in SEED_URL Table

We set a selection procedure for the situation when there are more "already crawled" sites competing for limited places for recrawling. First, we rank the "already crawled" sites according to their accepted document rate (ADR) and seed site reference count (SSRC) (see section 3.4.2). If two sites have the same accepted document rate, the one

with high SSRC will have higher rank than the one with low SSRC. The 27 ranked sites with their ADR and SSRC are presented in the table below.

Rank	ID	Link	ADR	SSRC
1	19	www.cs.virginia.edu/studpubs	22	206
2	1	www.cs.umass.edu	19	77
3	8	www.cs.uiuc.edu	18	91
4	3	www.cs.indiana.edu	17	151
5	18	www.sciencedirect.com	17	147
6	6	www.cs.cmu.edu	16	658
7	7	www.cs.umd.edu	16	225
8	5	www.cs.cornell.edu	15	424
9	9	www.cis.upenn.edu	14	452
10	2	www.cs.concordia.ca	13	25
11	20	www.cs.kent.ac.uk	13	8
12	4	www-cs.stanford.edu	12	137
13	10	www.cs.purdue.edu	11	130
14	11	www.cs.unc.edu	10	198
15	12	www.cs.toronto.edu/DCS/index.html	10	153
16	13	www.cs.columbia.edu	9	221
17	14	liinwww.ira.uka.de/bibliography	7	81
18	21	cjtcs.cs.uchicago.edu	6	9
19	24	reports-archive.adm.cs.cmu.edu/cs.html	6	7
20	25	www.cs.bu.edu/techreports/	5	144
21	26	cs.anu.edu.au/techreports/	5	78
22	23	www.mcs.vuw.ac.nz/comp/Publications/	4	11
23	15	dmoz.org/Computers/Computer_Science/Publications	3	15
24	16	www.elsevier.com	2	100
25	27	www.cs.arizona.edu/research/reports.html	1	122
26	17	www.computer.org	0	384
27	22	www.dmtcs.org	0	0

Table 20 Ranked Sites

We arbitrarily select some cut-off points to decide whether a crawled site is good, medium or bad. If the site rank is above 75% of the total number of eligible seeds, we consider that the site is a good site and all good sites are selected for the revisitng. In our case, there are 25% * 27 = 7 good seeds, i.e., the seeds with rank 7 and above are first selected for the next run. There are 15 – 7 = 8 places left for the remaining seeds. The site rank is between 50% and 75% of the total, i.e., the sites which rank between 7 and 14, are considered as medium seed and 50% of them are selected; hence, the sites with rank 8, 9, 10, and 11 are selected. The seeds, which rank lower than 50%, are treated as bad seeds. We still assign the remaining 8 – 4 = 4 places for them because they might have the potential to become good seeds if CNDROBOT finds some new accepted documents

from the sites in the next run. Therefore, the sites with rank 15, 16, 17 and 18 are also selected. Finally, these selected sites are inserted after the new sites into the SEED_URL table for the next run. Below is the new SEED_URL table that contains the seeds for the next crawl.

ID	Link	Host_name	Is_new_seed	Resume_flag	SDATE	EDATE	Num_ref_by
1	www.cs.wpi.edu	NULL	0	1	2005-07-22 23:55:41	2005-07-23 11:32:43	NULL
2	www.w3.org	NULL	0	1	2005-07-23 13:32:41	2005-07-28 15:31:41	NULL
3	dis.cs.umass.edu	NULL	0	1	2005-07-28 15:31:41	2005-07-28 15:33:43	NULL
4	nas.cs.umass.edu	NULL	0	1	2005-07-28 15:33:43	2005-07-28 15:38:44	NULL
5	cs.conncoll.edu	NULL	0	1	2005-07-28 15:38:44	2005-07-28 15:41:39	NULL
6	lass.cs.umass.edu	NULL	0	1	2005-07-28 15:41:39	2005-07-28 15:51:55	NULL
7	www-edlab.cs.umass.edu	NULL	0	1	2005-07-28 15:51:55	2005-07-28 15:52:15	NULL
8	iesl.cs.umass.edu	NULL	0	1	2005-07-28 15:52:15	2005-07-28 15:52:50	NULL
9	cise.nsf.gov	NULL	0	1	2005-07-28 15:52:50	2005-07-28 15:53:47	NULL
10	www.cse.ucsc.edu	NULL	0	1	2005-07-28 15:53:47	2005-07-29 16:25:24	NULL
11	www.soe.ucsc.edu	NULL	0	1	2005-07-29 16:25:24	2005-07-30 08:15:07	NULL
12	www.eecs.berkeley.edu	NULL	0	1	2005-07-30 17:31:41	2005-07-31 00:15:55	NULL
13	mallet.cs.umass.edu	NULL	0	1	2005-07-31 00:15:55	2005-07-31 00:53:49	NULL
14	ripples.cs.umass.edu	NULL	0	1	2005-07-31 00:53:49	2005-07-31 00:55:58	NULL
15	www.cs.vt.edu	NULL	0	1	2005-07-31 00:55:58	2005-07-31 00:56:00	NULL
16	www.cs.virginia.edu/studpubs	NULL	1	1	2005-07-31 00:56:00	2005-08-01 06:07:06	NULL
17	www.cs.umass.edu	NULL	1	1	2005-08-01 06:07:06	2005-08-01 09:40:00	NULL
18	www.cs.uiuc.edu	NULL	1	1	2005-08-01 09:40:00	2005-08-01 09:46:13	NULL
19	www.cs.indiana.edu	NULL	1	1	2005-08-01 09:46:13	2005-08-02 16:28:33	NULL
20	www.sciencedirect.com	NULL	1	1	2005-08-02 16:28:33	2005-08-02 16:30:42	NULL
21	www.cs.cmu.edu	NULL	1	1	2005-08-02 16:30:42	2005-08-04 04:58:18	NULL
22	www.cs.umd.edu	NULL	1	1	2005-08-04 04:58:18	2005-08-04 20:46:49	NULL
23	www.cs.cornell.edu	NULL	1	1	2005-08-04 20:46:49	2005-08-07 02:07:42	NULL
24	www.cis.upenn.edu	NULL	1	1	2005-08-07 02:07:42	2005-08-07 16:52:05	NULL
25	www.cs.concordia.ca	NULL	1	1	2005-08-07 16:52:05	2005-08-07 20:56:20	NULL
26	www.cs.kent.ac.uk	NULL	1	1	2005-08-07 20:56:20	2005-08-08 00:55:45	NULL
27	www.cs.toronto.edu/DCS/index.html	NULL	1	1	2005-08-08 00:55:45	2005-08-08 16:45:43	NULL
28	www.cs.columbia.edu	NULL	1	1	2005-08-08 16:45:43	2005-08-09 07:46:41	NULL
29	liinwww.ira.uka.de/bibliography	NULL	1	1	2005-08-09 07:46:41	2005-08-09 08:51:48	NULL
30	cjtcs.cs.uchicago.edu	NULL	1	1	2005-08-09 08:51:48	2005-08-09 08:56:23	NULL

Table 21 Sample of New SEED_URL Table

Note that the unselected "already crawled" sites have to compete again with other eligible sites for the third run. However, the number of new sites found could be fewer, in which case more quanta can be allocated to "already crawled" sites.

3.3.5 Priority Level for Site Revisiting

As mentioned in section 3.2, the priority level specifies the minimum time that a site has to wait for the next visit and it will be re-evaluated after each crawl. The priority level will be upgraded if the accepted document rate increases as illustrated in Table 22.

Downloaded Documents	Prior Crawl	Current Crawl
Total	1000	1100
Accepted	10	60
Rejected	990	950
ADR	1%	5%

Table 22 Increased Accepted Document Rate (ADR)

The priority level will be downgraded if the accepted document rate decreases as illustrated below.

Downloaded Documents	Prior Crawl	Current Crawl
Total	100	105
Accepted	20	20
Rejected	80	85
ADR	20%	19%

Table 23 Decreased Accepted Document Rate (ADR)

3.3.6 Subsequent Crawling

Niran and Arnon in their article [NA02] point out that the first crawl is not as efficient as the subsequent crawling because the crawler has no prior "knowledge" about the Web pages at any site. Along with the accumulated crawling experiences and the built knowledge base, the crawler will become more efficient. Three objects in their knowledge base are seed URLs, topic keywords, and URL prediction.

The purpose of our subsequent crawling is to discover documents from new seed URLs as well as to recrawl the previous seeds for updates. In order to improve performance in successive crawling, CNDROBOT takes advantage of the input fed by Document Filter Subsystem (DFC) and the results generated based on past crawling data by Link Analyzer and Statistics analyzer to better select seed URLs, control search depth levels, prune irrelevant directories, and predict URLs, etc. Details will be covered in Chapter 4.

3.4 Heuristics

While designing a Web robot, using heuristics to improve its performance is critical. In the past, researchers have studied and developed metrics used to evaluate the relevance and importance of a Web page and crawling efficiency. Generally, they can be summarized as being of two types: linked-based and similarity- based. Hub, authority, backlink count, forward link count, and page rank [JC98] are commonly used for link-based measures. Topics or keywords similar to the centroid of example pages and to the anchor texts surrounding the links [SM99] are used in similarity-based metrics to gauge the relevance of a Web page.

There are few published reports [SC99] [SK99] on crawling heuristics specifically for digital libraries. The goals of CNDROBOT differ from those of other search engines or focused crawlers in many ways. For example, CNDROBOT pays special attention to the quality of the downloaded documents; this indicates the effectiveness of our robot and the nature of the site. In addition, as an integrated part of a digital library, CNDROBOT uses the heuristics, which might enhance other subsystem's work. In order to bridge the difference between CINDI's goals and those of other search engines, we modified some of the existing heuristics and developed some of our own.

3.4.1 Modified Heuristics

Keyword in Web Page

The heuristic of keyword in Web page is a similarity-based metric that we used to find the number of domain keywords appearing in the text of Web page. New sites are selected from the foreign links extracted from previously crawled pages. We consider the hosts of these foreign links as good new seed candidates since they are contained in the Web pages of the seed site. From the theory of topical locality discussed in section 2.3, we know that the chance of getting topically related pages is higher by following these links. While the robot is selecting new sites, the Web pages of the foreign link hosts are

fetched and searched for domain keywords. Our experiments on 300 foreign link hosts (in chapter 5) show that one keyword found in the page is not significant enough to determine that the Web page is relevant to the topic. We use two keywords as the threshold.

Anchor Text in Web Page

We employ restrictive measures when selecting new sites from foreign link hosts. In addition to considering the keywords in a Web page, we also take into account the number of matches of anchor texts in the current page to the ones in the exemplary documents because the anchor texts in a Web page, to some extent, represent the characteristics of the page. For example, the home page of the computer science department usually includes the anchor texts of "people", "faculty", "admissions", and "academics", etc. and for a computer science publication Web site, its home page often includes anchor texts such as "publications", "research", and "abstract". Sometimes, we are not able to determine that the page is relevant by looking at the anchor texts. For instance, the home pages of non-computer science departments in a university also include "people" and "faculty" as anchor texts; however, these pages are not computer science related. Therefore, we use this heuristic with the keywords in the Web page to determine if a page is relevant or not. To qualify as a new site, at least one anchor text and one keyword must be found in the Web page of the host,

Document Link Citation Count

Document link citation count (DLCC) is similar to the backlink count metric. Backlink is defined as the number of links to a Web page that appears over the entire Web [JC98]. The backlink count is for citation count of a Web page whereas DLCC counts the number of times the document link is cited over the progress of crawling. For example, the document link http://www.csd.uch.gr/~hy558/papers/cho-order.pdf is cited on page

http://www.cs.usfca.edu/~wolber/blogs/internet/000537.html. Hence, we determine that the document "cho-order.pdf" is cited once by the Web page author.

Document Link Count

Document Link count is an extension of the backlink count metric. If CNDROBOT downloads the same document from different sites, for example, the document cho-order.pdf is found at http://oak.cs.ucla.edu/~cho/papers/cho-order.pdf and http://www.csd.uch.gr/~hy558/papers/cho-order.pdf, we increase the document link count by one. The procedure of updating the document link count and DLCC will be discussed in detail in chapter 4. Both heuristics are used to recognize the popularity of a downloaded document in the CINDI digital library.

3.4.2 Additional Heuristics

Seed Site Reference Count

Seed site reference count (SSRC) is defined as the number of times a seed URL is cited by pages from other domain names. We compute it by counting the number of times a seed host is shown in the column attribute host_name in the FOREIGN_LINK table. As mentioned early in this chapter, FOREIGN_URL table keeps the records of hyperlinks extracted from visiting pages and the hyperlinks that have different host names than their seeds. The higher the SSRC score, the more trusted the seed is. It is one of the important metrics that is used when selecting "already crawled" sites for recrawling.

Document Download Rate

Document download rate, which can also be called harvest rate, is a measure of a seed site and we define it as the ratio of the number of documents downloaded over the number of Web pages visited for a site.

$$\textit{Document Download Rate (DDR)} = \frac{\textit{Number of Documents Downloaded}}{\textit{Number of Web Pages Visited}}$$

The document download rate implies the density of the links to files in the Web page. High DDR indicates that the site might be a hub that contains many documents. It might be a digital library or a technical report archive. We use the DDR to determine if the robot should stop crawling this site and move on to crawl the next available one. A site with low DDR (< 1%) will not be further crawled because it is not cost effective. Details will be discussed in section 4.2.3.

Accepted Document Rate

Accepted document rate is defined as the ratio of the number of documents accepted over the number of documents downloaded.

$$\textit{Accepted Document Rate (ADR)} = \frac{\textit{Number of Documents Accepted}}{\textit{Number of Documents Downloaded}}$$

The accepted document rate is the most important heuristic that measures the value of the site to the CINDI library. A site that has a high accepted document rate has priority to be revisited.

Rejected Document Rate

Document rejected rate is defined as the ratio of the number of documents rejected over the number of documents downloaded.

$$\textit{Rejected Document Rate (RDR)} = 1 - \textit{ADR}$$

The rejected document rate is the complement of the accepted document rate. A site with a high rejected document rate should be visited less frequently.

All of the above-mentioned heuristics are computed and analyzed by CINDI statistics analyzer (see section 4.4). The DDR, ADR and RDR for each seed indicate the nature of a site and they are maintained in the SITE_STATS table. The ADR and the SSRC determine the rank of the crawled seeds. A high ranked seed has priority to be selected first for revisiting, and then the lower ranked one because of the likelihood of discovering high quality documents from it. The heuristics "keyword in Web page" and "anchor text in Web page" are used for screening out new seeds from the hosts in the FOREIGN_LINK table. If the Web page of the host contains enough keywords and sample anchor texts, it will be considered as a new seed in the subsequent crawl. The document link citation count and document link count are used to measure the popularity and authority of a downloaded document. They do not have direct effect or implication on the work of CNDROBOT. However, they enrich the feature of CINDI digital library. For example, when the user searches for documents via the VQAS, the title, author name, and document abstract of the returned documents can be displayed with the document link citation count.

Chapter 4
Implementation of CNDROBOT

In this chapter, we describe the implementation of CNDROBOT that runs on a Linux platform. Recall that the seed finder, web crawler, file fetcher, statistics analyzer and link analyzer are 5 modules of the CNDROBOT. We start out with the details of the algorithmic techniques for each module and a demonstration of Web interfaces for this application will be presented at the end of this chapter.

4.1 Seed finder

To find the start URLs for CNDROBOT to crawl with, the seed finder first submits a number of topic related queries to the Google and AltaVista search engines. Two temporary tables are used to hold the results returned from each search engine. The shared results, sites extracted by both search engines, are first stored in the NEW_SITES table followed by the remaining unique results. The order of the sites in the table is an insignificant matter because we consider them equally important. They would be randomly selected for the next crawling cycle.

We use the Google Web APIs in our application while performing the seed finding through Google. Google provides the Web APIs service for the developers to query its Web pages through computer applications. To access this service, a free account needs to be activated and a license key has to be included in the application when submitting queries. Google also places a limit of a maximum of 10 results received for each query up to 1000 results [GAPI]. Therefore, for each search term, 100 queries need to be issued to get 1000 results

Retrieving sites from AltaVista Web pages requires extra work. For example, we need to determine the URLs for each query term and the starting index for the result pages before downloading and parsing the pages. By examining the URL strings of AltaVista's Web

result pages, we can determine the scheme used for the URL string. If we submit the query using the keyword phrase "computer science department", for the first page, the URL string is

http://www.altavista.com/web/results?itag=ody&q=computer+science+department&kgs=0&kls=0&stq=0.

The URL string for the second page is

http://www.altavista.com/web/results?itag=ody&q=computer+science+department&kgs=0&kls=0&stq=10.

Notice that the query phrase is concatenated using "+" and included between "&q=" and "&kgs". We also find that only one element in the string was changed, that is the number after "stq=", which indicates the starting index number for the page. The overall algorithm of seed finding is shown below.

```
Input: Selected keyword phrases
Output: Sites extracted from Google and AltaVista's Web pages
Begin
    Connect_to_CINDI_database;
    //Google search
    GoogleSearch (keywords);
    Begin
       while(num_of_queries < 100) {
          set_key(googleKey);
          set_query_string(keywords);
          set_max_results(10);
          set_start_result(startIndex);
          urls := get_URLs_in_page ();
          foreach url (urls) {
            isDuplicated := check_for_duplicated_url ();
            if(! isDuplicated)
              store_url_into_GOOGLE_SEED_URL_table (url);
          }
         //start index incremented by 10 because each query returns 10 results
          startIndex :=  increment_start_index (startIndex, 10);
       }
    End
    //AltaVista search
    AltaVistaSearch (keywords, number of parse);
    Begin
      //transform to the form of computer+science+department
      k :=  tokenize (keywords, "+");
       c :=  adapt_to_AltaVista_URL_pattern (k);
      //in the first round, extract the links including commercial sites
```

```
    while (num_of_pages < 100 && number of parse = = 1){
        content := fetch_the_result_page (c);
        ps := parse_the_result_page (content);
        //encode URLs e.g. Substitute %2F with "/"
        eps := encode_special_characters_in_URL (ps);
        store_into_TEMP_SEED_URL_table (eps);
     }
    //in the second round, remove the commercial sites
     while (num_of_pages < 100 && number of parse = = 2){
        content := fetch_the_result_page (c);
        //get links near the "similar" tag
        gs := get_similar_web_page_URLs (content);
        egs := Encode_special_characters_in_URL (gs);
        if(has_a_similar_record_in_TEMP_SEED_URL_table (egs)){
           eps := retrieve_from_TEMP_SEED_URL_table;
           store_into_ALTAVISTA_SEED_URL_table (eps);
        }
     }
   End
   //Combine the search results
   comSites := get_shared_results (GOOGLE_TABLE, ALTAVISTA_TABLE);
   diffSites := get_unique_results (GOOGLE_TABLE, ALTAVISTA_TABLE);
   store_into_NEW_SITES_table (comSites, diffSites);
End
```

4.2 Web Crawler

4.2.1 Crawling Algorithms

The Web crawler has been implemented using two algorithms: Crawling with no crawling experience, also called naïve crawling, which is used for the initial crawling; The second algorithm is crawling with some knowledge acquired from previous crawling experiences. This algorithm is intended for subsequent crawling.

Both algorithms are essentially breadth-first search algorithms. They both comprise the same procedures of retrieving seed URLs; building the seeds list; initializing crawling frontier; downloading Web pages; extracting outer links; and expanding the frontier. The difference between them lies in determining if an extracted link should be appended to the crawling frontier. Without knowledge, the crawler cannot predict if the link is relevant or not. The only condition of not adding to the frontier is that the link has been

crawled already or it is not robot safe. Based on the experiences accumulated from the past crawlings, CNDROBOT is capable of determining that certain URLs should be ignored, deciding which site branches should be pruned and predicting if a subdirectory is irrelevant because of the semantics of its name.

In order for the robot to determine if the current crawling is the first time or not, the attribute Is_new_seed in the SEED_URL table is checked at the beginning of the crawl. If the values of this attribute are all 0, it indicates that the crawl is an initial crawl with all new seed sites and then the naïve algorithm will be used. Otherwise, crawling with knowledge algorithm will take effect. The overall algorithm for crawling is given below.

```
Input: Seeds from SEED_URL table
Output: Potential desired document links and new seed links
Begin
    Connect_to_CINDI_database;
    //retrieve the seed URLs
    seed_urls := retrieve_seed_urls_from_SEED_URL_table ( );
   //set resume flag to 0 for all the seeds in SEED_URL table
    Set_resume_flag_in_SEED_URL_table (seed_urls);
    //check if the values of attribute Is_new_seed are all zero
    isAllZero := check_SEED_URL_table ();
    If (isAllZero = = true)
      Crawling_with_no_experience (seed_urls);
    else
      Crawling_with_experience (seed_urls, knowledge);
    Begin
        seeds_list := initialize_seeds_list (seed_urls);
        While (#seeds_list > 0){ //while the seeds list is not empty
            su := pick_first_seed (seeds_list); //get the first available seed URL
             remove_picked_seed_from_list (su);
             //check the robots.txt file placed at the root of the site
             exist := check_robot_exclusion (su, "robots.txt");
             if ( exist = = 1) //there is robots.txt for the seed
               dp := get_disallowed_paths (su ); //dp is a list of disallowed paths
             else
               dp := set_list_empty ();
            //keep a record of start datetime for the current seed in SEED_URL table
            record_start_date_time (datetime, su);
             //the first item in the frontier is the seed url
            frontier := initialize_crawling_frontier(su);
            while (#frontier > 0 && download_rate > threshold){ //condition of stopping
               url := dequeue (frontier, 1);                    //crawling the site
               if (is_a_file_link) //a URL ends with txt, doc, pdf, ps, etc.
                  insert_into_PRE_DOWNLOAD_INFO_table (url);
```

```
            if (is_a_foreign_link) //a URL has different host name than the one of seed site
              insert_into_FOREIGN_LINK_table (url);
            else
              insert_into_VISITED_PAGES_table (url);
            content := fetch_the_page (url);
           pass := classify (content);
            if (pass) {
             write_to_a_file (content); //store the file into a non pdf directory
             store_into_DOWNLOAD_STATUS_table (url);
            }
            links := parse_the_page (content);
            foreach link (links){
              dirName := get_the_directory_name (link); //the string between last two "/"
              level := get_the_depth (link); //get # of levels of the URL
              if (is_initial_crawl) {
                if (is_robot_safe)
                  frontier := enqueue (frontier, link);
              }
              else {
                 if (is_not_crawled_before &&
                    is_robot_safe &&
                    level <= predefined_depth_level &&
                    is_not_in_DIR_TO_BE_AVOIDED_table (link, dirName) &&
                    is_not_in_STOP_DIR_LIST_table (dirName))
                     frontier := enqueue (frontier, link);
              }
            } //end foreach
           }//end while
           record_finish_date_time (datetime, su); //one site crawling finishes, the next
           update_resume_flag (su, 1);              //starts
         } //end while
      End
End
```

4.2.2 Page Classifier

A Web page can be a legitimate document in HTML format or simply a link carrier. The function of page classifier is to determine if a downloaded page is the genre of documents in CINDI library. As shown in the crawling algorithm, the content of each Web page is evaluated after it is downloaded. The classifier first searches for the words "abstract", "keywords", "chapter", "introduction", "bibliography", "acknowledgements", "references", "FAQ", "?", "question", "answer" and "appendix". Then it will calculate the number of times each word appears on the Web page. A page is marked as potential "research paper" if the words: "abstract", "keywords", "introduction" and "references"

appears at least once. If the words: "chapter", "bibliography", "acknowledgements" or "appendix" appear at least once, the page is classified as "thesis/report". If the page contains the words "FAQ", "Frequently Asked Questions" or the number of pairs of "question" and "answer" is greater than 3, we categorize it as a potential FAQ document. At this stage, we use a relatively loose classification policy to lower the probability of omitting a good document. The DFS subsystem, which is scheduled to run after document downloading from a site has been completed, will perform further filtering to eliminate ineligible documents.

4.2.3 Crawling Termination

The entire crawling process terminates in a normal and safe way if all the seeds in the seed list have been crawled. Switching from the current seed to the next seed happens either when the frontier of the current seed is empty or when the document download rate is less than the cut off point.

We specify two checkpoints for stopping the crawling of a site: the initial check takes place after 10,000 Web pages have been visited. Crawling fewer than 10,000 pages takes reasonable time (approximately 20 min/per 1,000 pages * 10 = 200 minutes) with reasonable system and network resources. Therefore, terminating crawling for one site will be considered only after 10,000 Web pages have been visited. The crawler will check if fewer than 100 file links from the site have been found. Less than 1% document download rate (DDR) is neither satisfactory nor cost effective. If 10,000 pages have been visited and DDR is less than 1%, the crawler will set the stop flag to 1 to indicate that a follow-up check is necessary

If the stop flag has been set to 1, the follow-up check is performed after another 1,000 pages have been visited. The document download rate will be re-calculated. If the rate is still less than 1%, the crawler will stop the crawling process for the current seed site and move to the next seed. Otherwise, the crawler will reset the stop flag to 0 and wait until

20,000 pages are visited. The stop condition will be repetitively checked for every 10,000 pages visited.

4.2.4 Crawl Resume

If the crawler ceases abnormally, the crawler needs to know the resuming point, i.e. what the first uncrawled site is so that the crawler does not have to start from scratch. The values in the column attribute Resume_flag are set to 0 for all the sites in the SEED_URL table when the sites are retrieved to initialize the seed list. Once crawling for a site has finished, the Resume_flag for the site will be updated to 1. By checking the value in the Resume_flag column, the crawler is able to find the restarting point.

4.2.5 Robot Script File

A cshell script file has been written to run the web crawler, keep a log file as well as rebuild the program java files.

```
#!/bin/csh
#cshell.ex
echo "Crawler starts at: " > output
date >> output
echo "By User:" >> output
whoami >> output
echo "Use command: at -f robotscript -m hh:mm" >> output
echo "Result: " >> output
echo "##########################################################" >> output
compile >> output
echo "execute using command java cndrobot"
java  cndrobot  >> output
echo "finished at " >> output
date >> output
```

Figure 7 Sample of Robot Script File

As shown in Figure 7, we use the Linux/Unix "at" command to execute the robot script file at a specified time. Output is a log file that keeps a record of the program start time;

finish time and crawling history generated by the "cndrobot" program. The file "compile" is another script file, which is used to clean old java class files and compile and build new java files.

4.3 File Fetcher

While CNDROBOT performs the crawling process, links for documents to be downloaded are maintained in the PRE_DOWNLOAD_INFO table. When the download cycle starts, CINDI File Fetcher (CFF) retrieves the records from the PRE_DOWNLOAD_INFO table and initializes a list of documents to be downloaded. It picks one file object from the download list; the object contains the url, file name and parent ID. Then CFF attempts to connect to the remote server using the given URL. A failure to connect can result from one of the following reasons: the file link is no longer valid or the access is password protected or there are network or server problems. When errors occur, CFF aborts the download process for this file and moves to the next available one.

Once a connection is open, CFF requests the file object. Before sending back the actual content of the URL, the server first sends back some information about the content of the URL, such as its length (size) and when it was last modified. If the size were too small, for example 0 bytes, the file would not contain enough information and would not be valuable for CINDI's collection. However, the size that is too small to be downloaded is to be judiciously determined. To that end, we did an experiment on the 8,376 records in the DOWNLOAD_STATUS table. We found that the download acceptance rate (DAR) decreases as the file size decreases. For instance, for a PDF file type, the DAR is 5.8% for file sizes less than 50K and the DAR is reduced to 0.7% when the file size is less than 10k. According to these empirical data, we establish a rule for the PDF files; we consider them insignificant if the size is less than 8K. Thus, the CFF will not download files smaller than this size.

Before checking if the current file is a new one or not, the URL of the file will be encoded to remove some special characters. The purpose of doing this has been discussed in Chapter 3. To avoid downloading the same file repetitively, CFF searches the DOWNLOAD_STATUS table, where the history of downloaded files is maintained. If there exists a record, which has the same URL address, file name and file size, it means that the current file has been downloaded before and the download process for this file is aborted.

When two files with the same file name and prefix url, but different file sizes are found, the chance is high that the already existing one is the old version and the one to be downloaded is an updated version. In this case, CFF checks the filter_flag for the existing file in the DOWNLOAD_STATUS table. If the filter_flag is 1, which indicates that the file has been accepted as a good document or the value is 0, which indicates that the file has not been filtered yet, CFF will download the file. If the file has been rejected by DFS, it will not be downloaded. For the older version, CFF will move it to the trash directory.

All the PDF files will be downloaded to a temporary directory "/pdf_tmp/", from which the CINDI subsystem DFS would filter them, then move accepted ones to a permanent directory "/pdf/" and throws rejected documents to the directory "/trash/". For non-PDF files, CFF will save them under the directory "/downloads/", where FCS retrieves and converts them into PDF format for filtering.

The Web page author might post the same document in different formats. For example, in CiteSeer, most of the documents are offered in pdf, ps, and ps.gz formats. To recognize the existence of different formats, the file name without file extension and the prefix URL will be used to compare the records in the DOWNLOAD_STATUS table. If both the prefix URL and the file name are matched with one of the records, CFF will recognize it as a file with different format and set the attribute "is_diff_format" to 1 in the table. For the file with only one format, its attribute is set to 2 by default. To provide more viewing and downloading choices for CINDI library users, CFF will download all of them.

To prevent the existing file from being overwritten, CFF checks if there is a duplicate file name in the designated location before writing the file input stream to a file. Consider a file with the name hello.pdf which is to be stored in the directory "pdf_tmp". CFF looks up the DOWNLOAD_STATUS table to find if there is a record whose file name and location are the same as the one for the new file to be stored, in this case hello.pdf. If so, the rename policy would take place. First, the file extension is removed. And then a digital number 0 will be appended to the stripped file name "hello". The new file name "hello0.pdf" will be used for the next search. As long as an instance can be found in the table, rename policy keeps changing the file name by adding a digital number and appending to the tail of the stripped file name. The final file name would be its system file name. We must keep a record of the original file name and system file name in the table since the original one is part of URL where the file was downloaded and the system one is used to track the file in the CINDI system.

As previously mentioned, a file will be treated as a new one if there is no record which matches the prefix url, file name and file size in the DOWNLOAD_STATUS table. But if two files with the same file name and file size are downloaded from different domain hosts, then a further verification is required. CFF resolves this issue by checking the file digital signature using the tool MD5. The MD5 algorithm was invented by R. Rivest at MIT laboratory [RR92] and it has been used to generate the "fingerprint" of a file. Theoretically, no two files will ever produce the same fingerprint unless they are identical. As shown in the algorithm (Fig. 12), the md_check_flag is set to 1 if the condition of further checking is met; otherwise, the md_check_flag is set to 2. The current file is first saved to the designated location after its name is changed if required. Then if the check flag is equal to 1, CFF retrieves the location and file name for the previously downloaded file from the table and executes MD5 on both files. In the following example, the current file "cho-order.pdf" is prepared for digital signature verification. Since the file name already exists in the system, its name is first changed to "cho-order0.pdf". Then the file "cho-order.pdf" from the directory "/cndoc1/pdf" for the

existing file is retrieved. CFF determines if the two files are identical by comparing their 128 bit hash values calculated by using the MD5 signature.

```
MD5 (/cndoc1/pdf/cho-order.pdf) = 30cefae7d6d7daf6f116e6df83558960
MD5 (/cndoc1/pdf_tmp/cho-order0.pdf) = 30cefae7d6d7daf6f116e6df83558960
```

Figure 8 Sample Output of MD5

If two files are found to have the same MD5 signature, but were downloaded from different sources, we are interested in knowing which one is the original and how often it is referred by other sites. The one that is referred by others, we call the *source* and the one that refers to the other, we name it as *referrer*. To determine which document is more "original" than the other, we compare the last modified date for the downloaded files and deem the one with the earliest modified date as the "original". The currently chosen source might become a referrer if an earlier "original" one were found later on. We mark –1 for the referrer and the number of references for the source in the column of "Num_ref_by" in the DOWNLOAD_STATUS table. The value of the attribute "Num_ref_by" is an important measurement of the document "popularity". In addition, we maintain a detailed record for the relationship between referrer and source documents in the DOCUMENT_REF_BY table. Finally, the duplicate file will be removed from the system. The overall file download algorithm is given as follows:

```
Input:   prefix_url, file_name, parentID in PRE_DOWNLOAD_INFO table
Output: 1 if a file has been successfully downloaded under local directory
         0 if file download has been failed
       Begin
         Connect_to_CINDI_database;
         //Initialize a list of file download info po
         po := retrieve_download_info_from_PRE_DOWNLOAD_INFO_table ( );
         while (#po > 0) { //while the list is not empty
           fi := get_one_file_to_be_downloaded_info (po); //fi is one element in po
         is_downloaded (fi); //return 1 or 0
        begin
           fn := get_file_name (fi); //file name with the file extension
           fu := get_file_URL (fi); //get URL address of the file
           dl := get_num_of_back_slash (fu); //find directory level of the file
           ft := get_file_type (fn);
           fu := remove_special_characters (fu); //encode the URL string
           conn := connect_to_server (fu);
           if (conn = = null) //invalid URL
```

```
  return 0; //download fails
 fs := get_content_length (conn); //get file size
 fd := get_file_last_modified_date (conn);
 if (ft = = "pdf")  //pdf files are stored in "pdf_tmp" directory
   lo := set_file_location ("pdf_tmp");
 else //other types of files are stored in "downloads" directory
   lo := set_file_location ("downloads");
 // filter duplicated file
 is_contained := check_is_matched (fn, fu, fs);
 if (is_contained = = 1) // same file has been downloaded from the same link
   return 0; // move to the next file;
 if (is_contained = = 0) //not downloaded before
   continue;
 //check if the file has different formats
 df := check_is_different_format (fn, fu);
 1: if return true;
 2: if return false;
 //rename if a duplicate file name exists in local directory
ofn := set_original_file_name (fn);
 new_file_name :=  rename_policy (fn, lo);
if(new_file_name ! = fn) { // file has been renamed
  rn := set_is_renamed (1);
  fn := new_file_name;
 }
 else {
  rn := set_is_renamed (2);
 }
 //further check if there exists a file with the same name and size but
 //different URL in DOWNLOAD_STATUS table
 dID := file_trace (fu, fn, fs);
 if (dID > 0) //if document ID > 0
  mdf := set_md_check_flag (1); //need further to check file signature
 else
  mdf := set_md_check_flag (2); //no need to check file signature
//filter some small size file and file with name such as cv, resume
 filter_file (fn, fs);
 write_file_to_directory (fn);
 change_file_access_mode (fn); //allow group member to r/w
 if (mdf = = 1){ //check file signature using md5
  m1 := perform_md_check (ofn);
  m2 := perform_md_check (fn);
 }
 if (m1 = = m2) //two files are identical
   id := set_identity_flag (1);
 else
   id := set_identity_flag (2);
 //store downloaded file info into DOWNLOAD_STATUS table
 store_info (fu, ofn, fn, lo, fd, ft, fs, dl, df, rn, parentID);
//update DOCUMENT_REF_BY table
if (id = = 1) {
 //check if the document has been referred before
```

```
            rid := get_referred_file_ID (ofn);
            if ( rid > 0) { //meaning file has been referred before
              reset_ref_by(ofn, rid);
            }
            else {
              set_ref_by (ofn);
            }
            remove_identified_file_from_directory (fn, lo);
          }
          return 1; //indicate the file has been successfully downloaded
        end
}//end while
End
```

4.4 Statistics Analyzer

CINDI Statistics Analyzer (CSA) retrieves and analyzes the data in SEED_URL, VISITED_PAGES, FOREIGN_LINKS, DOMAIN_KEYWORD and DOWNLOAD_STATUS tables to produce a series of statistical results and store them into tables SITE_STATS, SITE_REF_BY, LINK_REF_BY, RDVT, and LEVEL_STATS respectively. It also updates the CRAWLED_SITES table and creates a new SEED_URL table for subsequent crawl.

The first step is to build the SITE_STATS table. CSA first retrieves the site ID and URL for all the sites in the SEED_URL table. For each site, CSA goes to the VISITED_PAGES table to get the total number of pages. The number of pages for a site is calculated as the difference between the index number of the last visited page and that of the starting page. The starting page of a specific site is the first record with the specific siteID in the VISITED_PAGES table and the starting page always has a value of 0 in its attribute parentID. The last visited page of a specific site is the last record with the specific siteID. The index number of a page is the value of the attribute PID. Next, CSA starts to count the number of downloaded documents for each site. CAS determines that a document is downloaded from a specific site if the parentID of the document in the DOWNLOAD_STATUS table is within the range of the starting page index and the last visited page index for the site. Therefore, the total number of downloaded documents for a site is the total number of downloaded documents whose parentID values are greater

than the starting page's index number and less than the last visited page's index number for the site. We can get the total number of accepted documents and rejected documents for a site in the same way, but we need to check the values of the filter_flag for those documents. The download rate, accepted document rate and rejected document rate are calculated (discussed in Chapter 3). The crawling time for a site is computed as the difference between the starting time and the finish time, which are stored in the attributes SDATE and EDATE in the SEED_URL table.

The priority levels for the revisited sites in the SEED_URL table are reevaluated according to the new accepted download rate. The siteID in the CRAWLED_SITES is retrieved and used to find the old accepted document rate for the site in the SITES_STATS table. The two accepted download rates are compared, and then the priority level for the site in the CRAWLED_SITES table is updated according to the policy discussed in section 3.3.5. The priority level for newly crawled sites is 1. Finally, the total number of pages visited, total number of documents downloaded, total number of documents accepted, total number of documents rejected, download rate, accepted document rate, rejected document rate, and completion time for the newly crawled sites are inserted and those for revisited sites are updated in the SITE_STATS table

The next step is to select seeds for the subsequent crawl. For the subsequent crawl, we attempt to maintain half the number of seeds for the new sites if there are enough sites in the NEW_SITES table. Before retrieving new seeds from the NEW_SITES table and storing in the SEED_URL table, CSA first discovers the new seeds from the hosts of foreign links in the FOREIGN_LINK table, as we discussed in Chapter 3. These hosts are first selected for uniqueness and then stored temporarily in a table. The temporary table is joined with the CRAWED_SITES table and the NEW_SITES table to ensure that these seed candidates have not been crawled before nor have been already selected. To qualify as a new seed, two general criteria must be met: 1) the candidate's Web page must contain domain keywords 2) there is similarity between the candidate's Web page and the exemplary documents. For each candidate, CSA downloads the Web page and searches for domain keywords in its content. The number of different keywords that appear in the

text of the Web page is recorded. To compare the similarity, a Representative Document Vector table, which contains the most common anchor texts in exemplary documents, is first built. An exemplary document is the home page of the seed in the SEED_URL table. All the anchor texts in the exemplary documents are first extracted and stored into a temporary table. We group duplicated anchor texts on a page to represent a single anchor text and we count the number of documents in which the anchor text occurs to find what is the representative anchor text used to describe an exemplary document. A temporary table is needed for processing raw data because the version of MySQL we used does not support some of the functions, such as "view" function. If the count of a specific anchor text is greater than 25% of the total number of exemplary documents, this anchor text is representative enough and will be stored into the RDVT table. For example, if an anchor text occurs on more than 7 (25%*30) exemplary documents out of 30, it is representative. The percentage of 25% is chosen based on the experimental results from 30 sites with 584 anchor texts. When a candidate's Web page is downloaded, the anchor texts in it are extracted and compared with anchor texts in the RDVT table. The number of matches together with the number of different keywords recorded is used to determine if the candidate is qualified as a new seed. If both of the numbers are greater than 0 or either one of these numbers is greater than 1, we consider it as a new seed; otherwise, it will be treated as a stop URL, which should never be crawled. Those selected candidates will be inserted into the NEW_SITES table.

All the sites in the SEED_URL and CRAWLED_SITES tables will be checked for the time period that they have waited to determine if they are eligible to compete for revisiting. To obtain a list of eligible crawled sites, the priority levels of sites are retrieved, translated and this time is added to their last end crawl time. This represents the wait time and is subtracted from the current time. If the value is less than 0, it means that the site has waited sufficiently and is eligible for competition. Otherwise, the site has to wait for the next cycle. The number of available places for the "already crawled" sites is calculated by subtracting the number of places taken by new sites from the total number of places available, which is set to 30. The numbers of places for the "already crawled" sites will increase if fewer new seeds are found. If the number of places is greater than

the number of eligible seeds, all the "already crawled" seeds will be selected for recrawling. If not, the following selection procedures will be taken. First, all eligible seeds are ranked according to their document accepted rates. Second, quality evaluation points are calculated. The crawled seeds are selected according to the policy discussed in chapter 3 and then stored in a revisiting site list, which includes all the revisiting sites for the next run.

In order to know the number of times a site has been referred by other hosts in previous crawling, CSA builds the SITE_REF_BY table by joining two tables: SEED_URL and FOREIGN_LINK on a common attribute host name. The number of instances for a site in the table is the number of times of reference for the site. Then CSA goes to the SEED_URL table to update the value of the attribute Num_ref_by for a site that has a record in the SITE_REF_BY table.

The next task is to build the LINK_REF_BY table and update the column num_ref_by (see Table 7) in the DOWNLOAD_STATUS table. LINK_REF_BY is built by selecting the attribute prefix_url in the DOWNLOAD_STATUS table and the url in the FOREIGN_LINK table if the values in the two tables are equal. One instance of a record in the LINK_REF_BY table means that the URL that links to a downloaded document in the DOWNLOAD_STATUS table has been cited in another domain host. The total number of instances will be counted for each document and is used to update corresponding entries in the DOWNLOAD_STATUS table.

While the File Fetcher downloads the document, the directory level for the document is obtained by counting the number of "/" in the file URL. The value of the directory level is kept in the attribute level (see table 9) in the DOWNLOAD_STATUS table. CSA goes through the table and computes the percentage of occurrences for each level. The percentage indicates the probability of the directory level where the document is located. Finally, these statistical results are stored in the LEVEL_STATS table.

Finally, CSA stores the information of the newly crawled sites from the SEED_URL table into the CRAWLED SITES table; updates the information of the revisited sites in the CRAWLED_SITES table and removes newly crawled sites from the NEW_SITES table. Once this has been done, the SEED_URL table will be recreated and will retrieve seeds from the NEW_SITES table and from the revisiting site list. The general statistical analysis algorithm is exemplified as followings:

```
Input: data in SEED_URL, NEW_SITES, VISITED_PAGES, FOREIGN_LINKS,
       DOMAIN_KEYWORD, CRAWLED_SITES and DOWNLOAD_STATUS tables
Output: data in statistics tables: SITE_REF_BY, LINK_REF_BY,
       SITE_STATS, RDVT, NEW_SITES, SEED_URL
Begin
   Connect_to_CINDI_database;
   Begin
      //build site statistics for each crawled site
      sites := get_crawled_sites ();
      foreach site (sites){
         tp := get_total_pages (site); //get total number of visited pages
         td := get_total_downloads (site); //get total number of downloaded
         ta := get_total_accepted (site); //get total number of accepted documents
         tr := get_total_rejected (site); //get total number of rejected documents
         sd : =get_start_date_time (site);
         ed := get_end_date_time (site);
         pd := compute_percentage_of_download (tp, td); //get DDR
         pa := compute_percentage_of_accepted (td, ta); //get ADR
         pr := compute_percentage_of_rejected (td, tr); //get RDR
         time_period := compute_time_period(ed, sd); //crawling period for each site
         update_priority_level (site);
         store_into_SITE_STATS_table (tp, td, ta, tr, pd, pa, pr, time_period);
       }
   End
   Begin
     //select new seeds from foreign links
     foreach sampleSeed (sites) {
       content := download_page (sampleSeed);
       ats := extract_anchor_texts (content);
       store_into_TEMP_RDVT_table (ats);
     }
     //group by anchor text and count the number of occurrence
     sl := sort_TEMP_RDVT_table ();
     store_into_RDVT_table (sl);
     ks := get_domain_keyword_set ();
     sas := get_sample_anchor_text_set ();
     flhs := get_foreign_link_host_URLs ();
     foreach flh (flhs) {
       content := download_page (flh);
```

```
    pats := extract_page_anchor_texts (content);
    n := compare_similarity (sas, pats); //return # of occurrences
    m := search_keywords (ks, content); //return # of occurrences
    if (n > 0 && m > 0) {
      store_into_NEW_SITES_table (flh);
    }
    else (n > 1 || m >1){
       store_into_NEW_SITES_table (flh);
    }
  } //end foreach
End
Begin
  //get eligible crawled sites
  ecl := get_eligible_crawled_sites ();
  ns := get_number_of_place_available_for_crawled_seeds (#sites, #newseeds);
  if (ns >= #ecl){
     //store selected old seeds for revisiting into the list
     list := store_into_revisiting_sites_list (ecl);
  }
  else {
     //-------------------------------------------------
     //|      |    |    |    |    |    |    |
     //      UM   MU   MD   ML   LM   LL
   //rank crawled sites and select some for the subsequent crawl
   rsl := rank_sites_order_by_accept_percentage (ecl); //return ranked site list
   mp := get_mid_point (#rsl); //get middle point of the total number of sites
   um := get_upper_mid_point (#rsl);
   mu := get_mid_upper_point (#rsl);
   ml := get_mid_lower_point (#rsl);
   //select the remaining crawled seeds according to policy
   nsl := select_remaining_sites_from_list (mu, md, ml, rsl);
   //store selected old sites for revisiting into the list
   list := store_into_revisiting_sites_list (nsl);
  }
End
Begin
  //build SITE_REF_BY table
  sids :=build_SITE_REF_BY_table (); //return seed ID list
  //update num_ref_by in SEED_URL table
  update_num_ref_by (sids);
End
Begin
  //build LINK_REF_BY table
  build_LINK_REF_BY_table ();
  //build LEVEL_STATS table
  build_LEVEL_STATS_table();
End
Begin
  //store the site info in SEED_URL table into CRAWLED_SITES table
  store_site_info_to_CRAWLED_SITES_table ();
  //remove newly crawled sites from NEW_SITES table
```

```
        remove_sites_from_NEW_SITES_table ();
        recreate_SEED_URL_table ();
        //retrieve new sites from NEW_SITES and store them in SEED_URL table
        ns := get_new_sites_for_subsequent_crawl ();
       //store both new seeds and revisiting sites to the table
        store_into_SEED_URL_table (ns, list);
     End
  End
```

4.5 Link Analyzer

Link Analyzer (LA) first analyzes the URL of downloaded files and visited Web pages, and then builds the STOP_DIR_LIST and DIR_TO_BE_AVOIDED tables by examining the relationship between the directory name and the URL of the visited Web page that contains all rejected documents as well as the relationship between the crawled seed sites and the directories to be avoided. The STOP_DIR_LIST table contains the general directory names to be avoided for any sites whereas the DIR_TO_BE_AVOIDED table maintains the specific directory names related to some crawled sites. The overall link analysis algorithm is illustrated as follows.

```
Input: data in DOWNLOAD_STATUS and VISITED_PAGES tables
Output: data in STOP_DIR_LIST and DIR_TO_BE_AVOIDED tables
Begin
  //find directory name whose directory contains bad documents
  //get URLs, parentID, count for invalid downloads from DOWNLOAD_STATUS
  //table, group by parentID
  bds := get_invalid_documents_URL ();
  foreach bd (bds) {
     url := get_URL (); //get prefix url of the downloaded file
     count := get_Number_of_Invalids ( );//get # of invalids downloaded from the same page
     pid :=  get_parentID(); //parentID is the PID in VISITED_PAGES table
     //get total downloads from one page
      nd :=  get_number_of_documents_downloaded_from_one_page (pid);
      //parse the URL to get the directory name
     dn := get_directory_name (url);
      store_into_temp_dir_stop_list_table (dn, count, nd);
  }
  //group by directory name, sum(#invalids), sum(#downloads)
  sort_temp_STOP_DIR_LIST_table ();
  store_into_STOP_DIR_LIST_table();
 End
 Begin
```

```
//find directory name, which has no documents at all
//get all the parentIDs in DOWNLOAD_STATUS table
pids := get_parentIDs_from_DOWNLOAD_STATUS_table ();
//get seed sites' PIDs in VISITED_PAGES table
spids := get_seeds_PID();
foreach spid (spids) {
  //for each seed, get page's siteID, PID, url at the first level
  //from VISITED_PAGES table
  pis := get_first_level_page_info (spid);
  foreach pi (pis) { //for each page at the first level
    sid := get_site_ID ();
    pid := get_pageID ();
    link := get_URL ();
    //initialize a flag to indicate if a document found under that directory
    pass := true;
   //find if the page has child
    hasChild := find_children_of_the_page (pid);
    while (hasChild = = true) {
      //get children IDs
      childIDs := get_child_IDs (pid);
      foreach childID (childIDs) {
        //check if there is document downloaded from that page
        hasDocument := check_is_in_DOWNLOAD_STATUS_table (pid);
        if (hasDocument = = true)
          return pass := false;
        else
          find_childrean_of_the_page (childID); //recursive call
        }
      } //end while
      if (pass = = true ) {
        //parse the URL to get the directory name
        dn := get_directory_name (link);
        store_into_DIR_TO_BE_AVOIDED_table (siteID, directory, link);
      }
    } //end foreach
  }//foreach
 End
End
```

The attribute parentID in the DOWNLOAD_STATUS table is the primary key PID in the VISITED_PAGES table (see Fig 9). The parent ID of a downloaded document is the page ID of the visited Web page that contains the hyperlink to the document. We know if the document is accepted or rejected by checking the filter flag updated by DFS. To find the names of the directories that contain all the rejected documents, LA first goes to the DOWNLOAD_STATUS table; groups the rejected documents by parent ID; counts the number of occurrences (*bc*) to know how many bad documents are downloaded from the

parent Web page and then retrieves the prefix urls and parent IDs of these documents. Next, using the parent ID, LA gets the total number of documents downloaded (*td*) from the same page. If *bc* is equal to *td*, it means all the documents downloaded from that page are invalid. LA searches the VISITED_PAGES table; finds the page whose PID matches the documents' parent ID and then retrieves the page's URL, which is parsed to get the directory name. The directory name, *bc,* and *td* are stored into a temporary table with the corresponding columns of dir, num_of_invalids and num_of_downloads for sorting. A sample table of results is shown in Table 24:

dir	num_of_invalids	num_of_downloads
papers	46	105
previous_tests	51	102
pdf	54	62
willie	44	44
techreports	71	252
ranveer	63	87
public_pdfs	61	88
2005	68	90
handouts	42	42
handouts	41	41
concepts	45	78
2003sp	46	49
documents	41	97
Lectures	44	44
Lectures	45	45

Table 24 Sample of Temporary STOP_DIR_LIST table

From the table above, we notice that the directory names "handouts" and "Lectures" appears twice, which means that two sites have these directory names included in one or more of the URLs of the Web pages. All of the documents under them are invalid. Thus, we can determine that directories with either of these names are the ones that the robot should skip crawling in the future. However, if a valid document were found under a directory, we would not skip it. Therefore, to establish the STOP_DIR_LIST table, LA first groups the directory names and adds up the number of invalids and the number of downloads in the temporary table. A directory name is stored into the STOP_DIR_LIST table only if the percentage of invalids is 100.

After a seed is crawled, we can determine which Web pages and consequently which directories in the seed site contain no document at all. For example, in Figure 9, the visited Web pages with PID 1, 2, 3, 4, and 5 do not contain any documents because they

have no entries in the DOWNLOAD_STATUS table. However, their child pages might have documents downloaded, for example, the child page 7 and 8 of the page with PID of 5 have 4 downloaded documents. Thus, the robot cannot skip crawling the page with PID 1, 2, and 5, but can ignore page 3 and 4 because they and their children do not contain any documents. Although it is too early to conclude that the robot should avoid these Web pages because there might be new documents discovered on them in the next visit, it is safe to suspect that these Web pages will be highly unlikely to have any documents. If the same situation persists after the seed has been revisited two or more times, we can determine that these Web pages should be skipped in future runs.

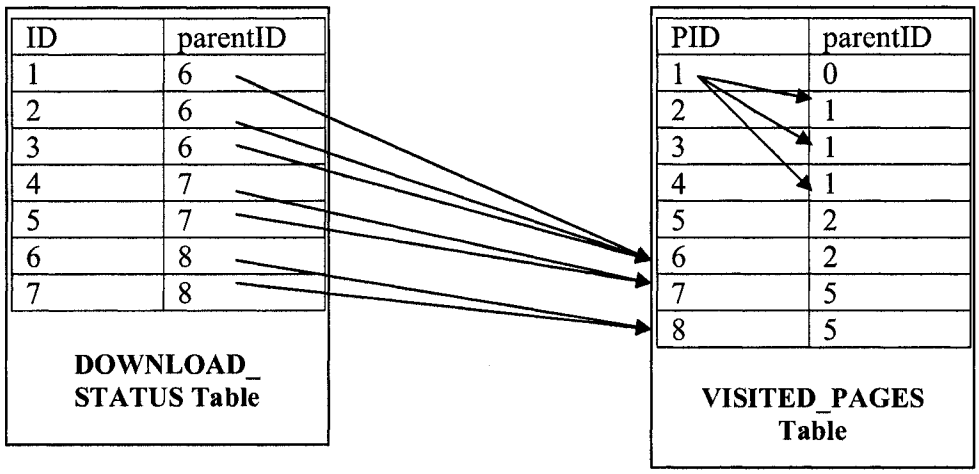

Figure 9 Sample of Relationship Between DOWNLOAD_STATUS Table and VISITED_PAGES Table

A directory and Web pages to be avoided are obtained by using the following procedures: 1) start from the root (the visited Web page with parent ID of 0); 2) check if the current page contains a document link, i.e. has an entry in the DOWNLOAD_STATUS table, return false if one is found, meaning that we abort further searching action and continue to check the next element; 3) check if the page has a child, return true if no child, meaning that no document is found and we store the url, siteID, and PID into the DIR_TO_BE_AVOIDED table; 4) use the recursive call to get child and grand child pages, which should also contain no documents.

4.6 Web Interfaces

To accommodate the need to control and monitor the CNDROBOT related activities through the Web, a secure, dynamic and easy-to-use Web application is designed and implemented using PHP programming language. Through the Web interfaces, authenticated users are able to start, schedule, stop and resume the crawling process, find seeds, fetch files, filter downloaded documents and view database tables. In addition, this application provides a utility to restore missed files if a copy of file is corrupted, lost or deleted accidentally.

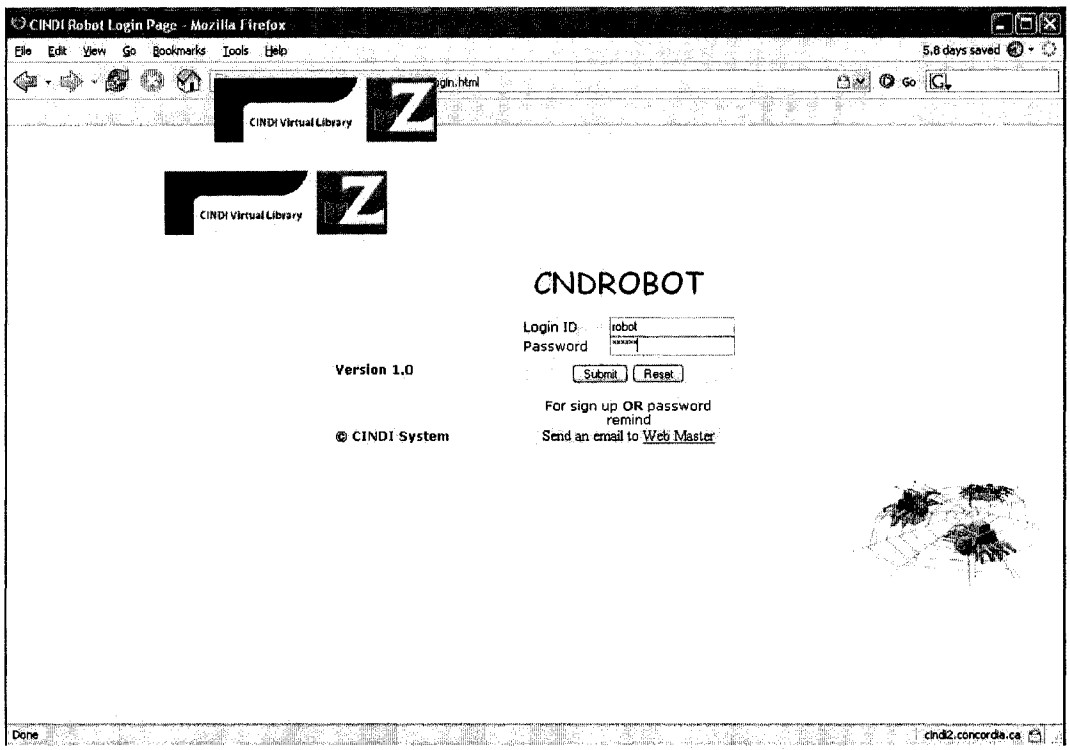

Figure 10 Login Page

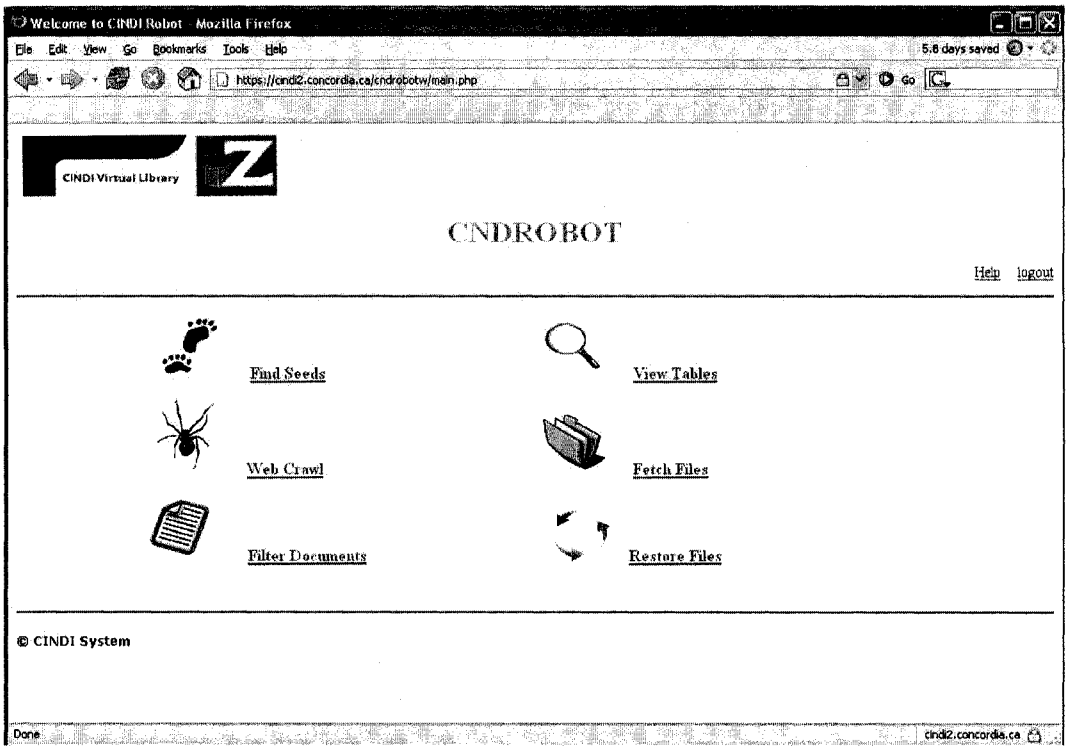

Figure 11 Main Page

CINDI Web server uses Apache Secure Sockets Layer (SSL) to establish a secure connection between a client and a server, over which data can be transmitted securely. In addition, users need to identify themselves from the login page as shown in Figure 10. After identity is verified, users can access to the main page in Figure 11.

It is required to go through the process of finding seeds before the initial crawling. However, at any time, we can find more seeds by following the steps illustrated in Figure 12.

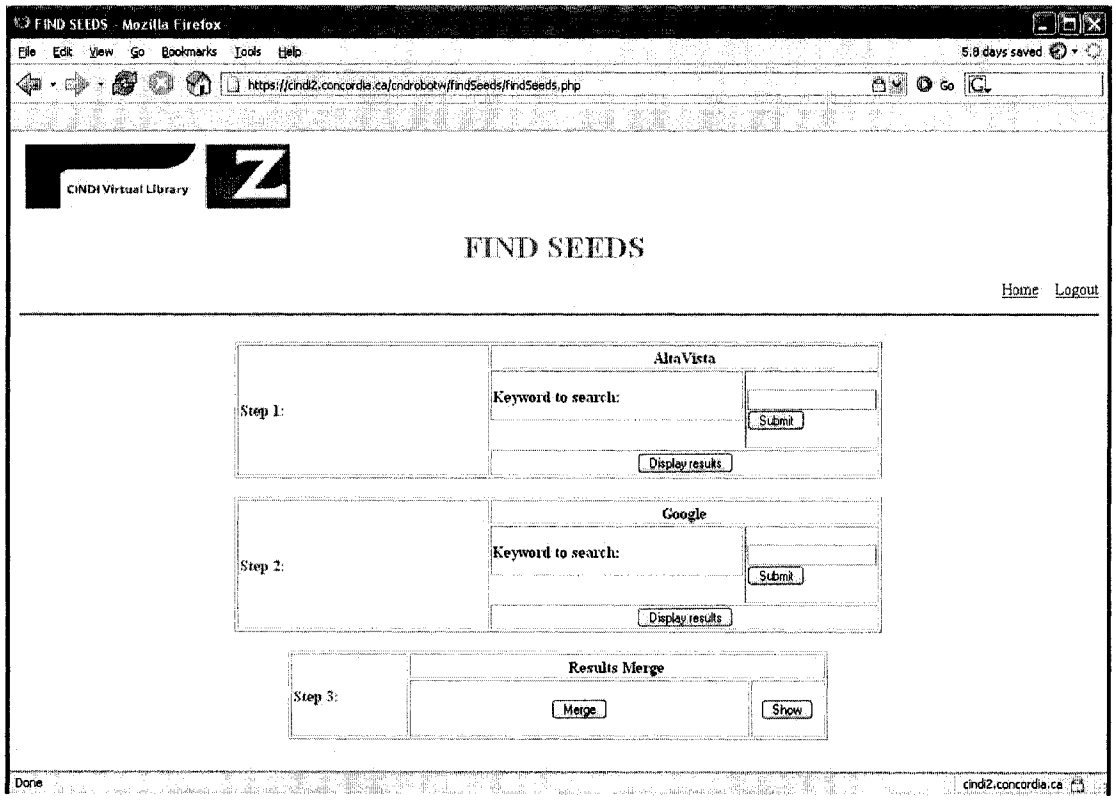

Figure 12 Find Seeds Page

Users need to input the query keywords in the text box and then click the "Submit" button to start the search. They can view the search results of the search engine employed by clicking the corresponding "Display results" button. After clicking the "Submit" button, a processing page (Figure 13) will be displayed. On the page, users have the choice either to stop the process or to continue to view the running status.

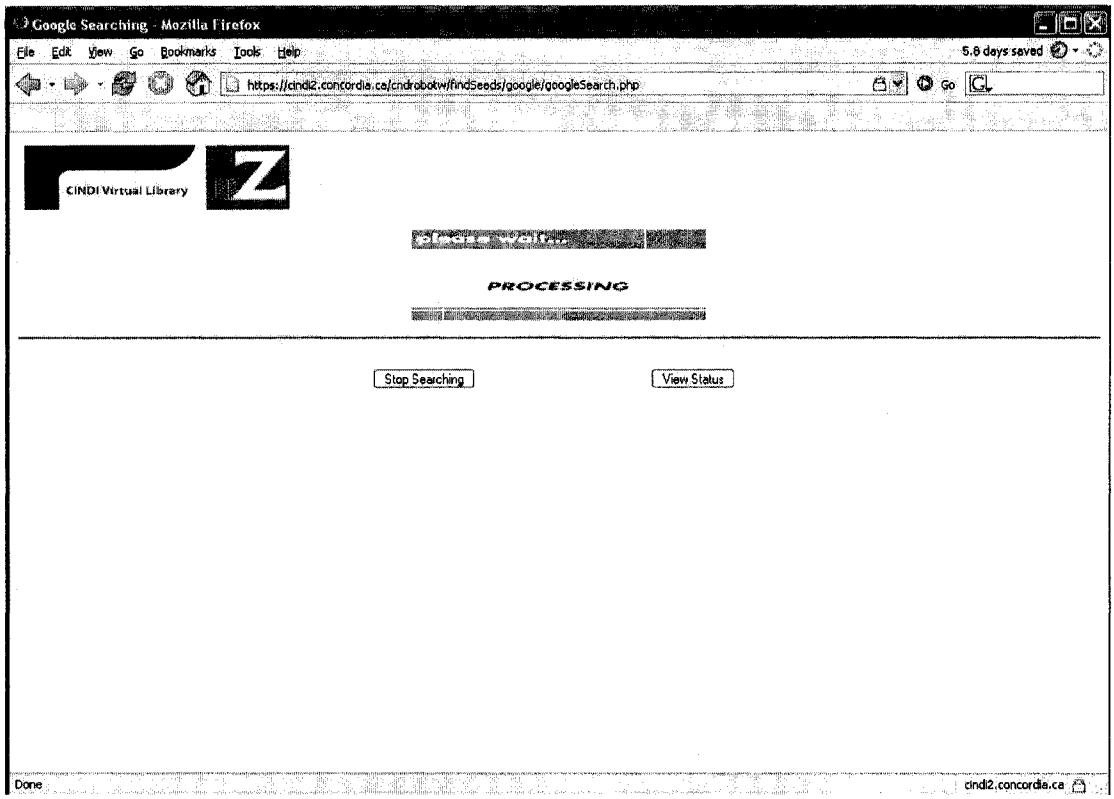

Figure 13 Processing Page

If users select to view the status, they can monitor the search process in the running status page (Figure 14). If users decide to stop the search in the middle, they can click the "Stop process" button, which has the same effect as the "Stop searching" button in Figure 13. They all bring users to the show process page as shown in Figure 15. The show process page lists the processes owned by the current user, including the process that ran the search program. To stop searching, users need to key in the ID belong to that search process and click the "Kill" button.

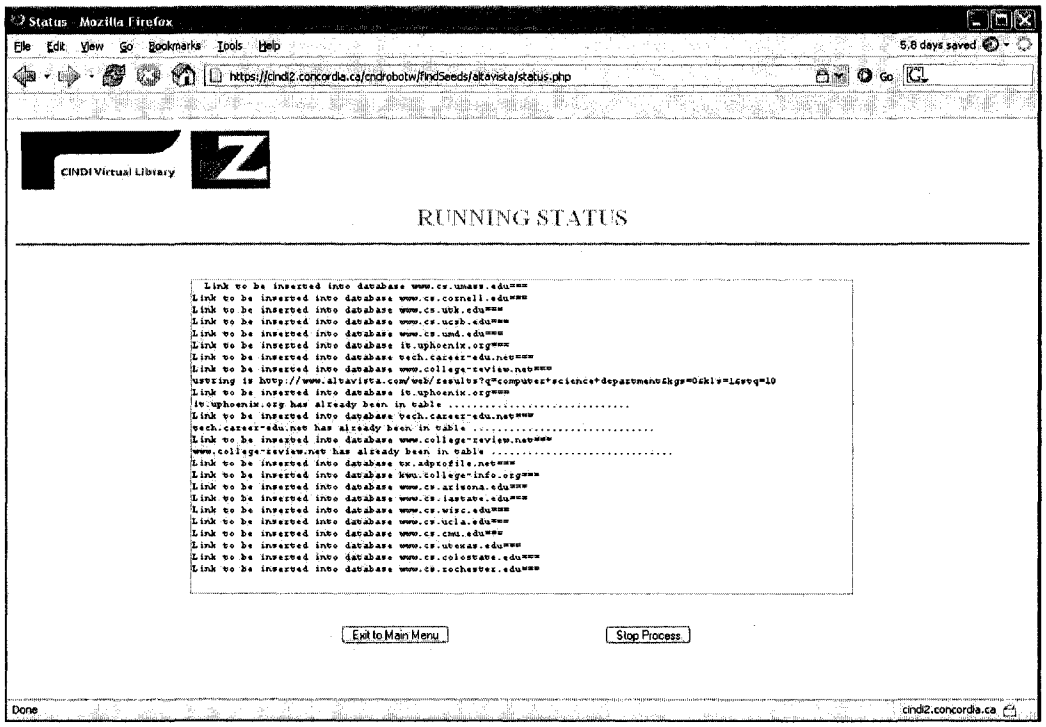

Figure 14 Running Status Page

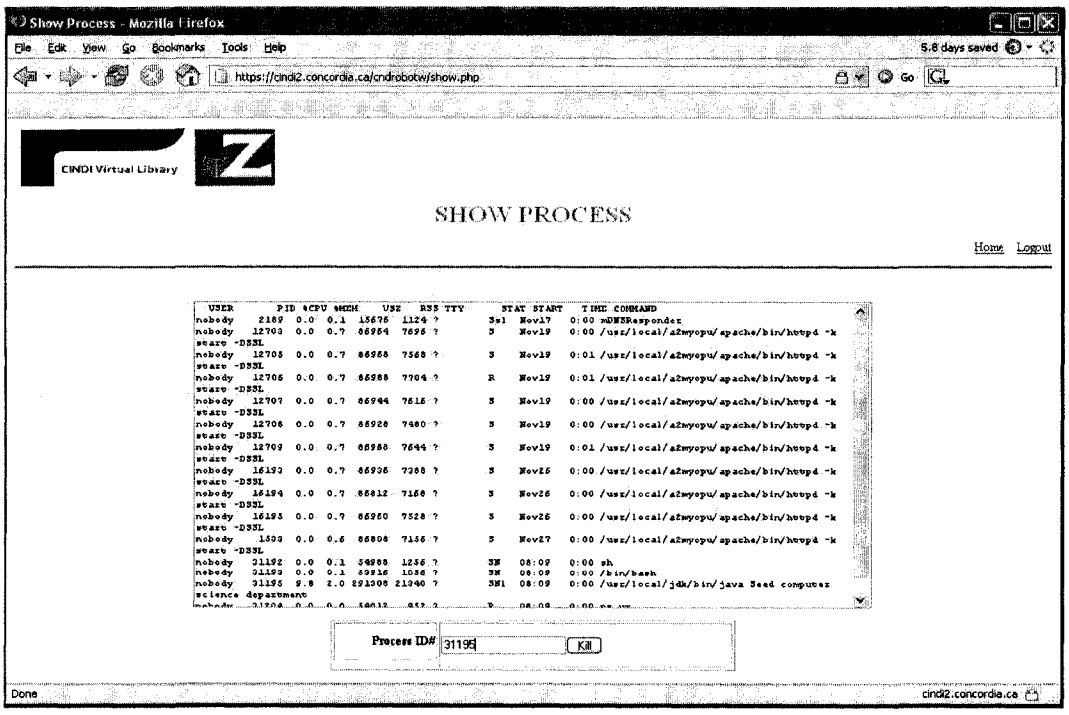

Figure 15 Show Process Page

To run the web crawler, click the link "Web Crawl" on the main page. The web crawl page is shown in Figure 16.

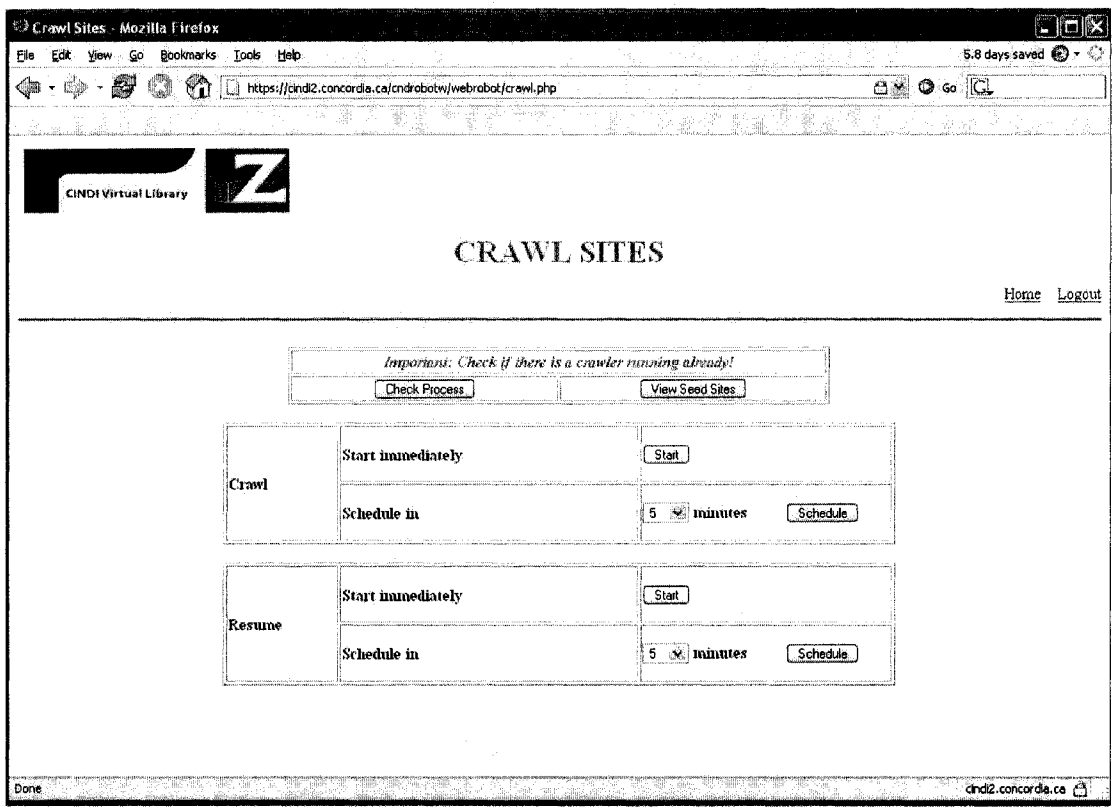

Figure 16 Web Crawl Page

Before starting the crawl process, it is important to check no other crawler running by clicking the "Check Process" button and ensure that seeds are ready by clicking the "View Seed Sites" button. If this is an initial crawl, select the buttons in the middle table. If this is a resume crawl, apply the functions provided in the last table. Both the initial crawl and resume crawl can start immediately or schedule to run at a specified time. Suppose that a user has scheduled to resume the crawl and decide to cancel the task now. After reaching the schedule process page (Figure 17), he/she can remove the scheduled task in the job queue page as shown in Figure 18.

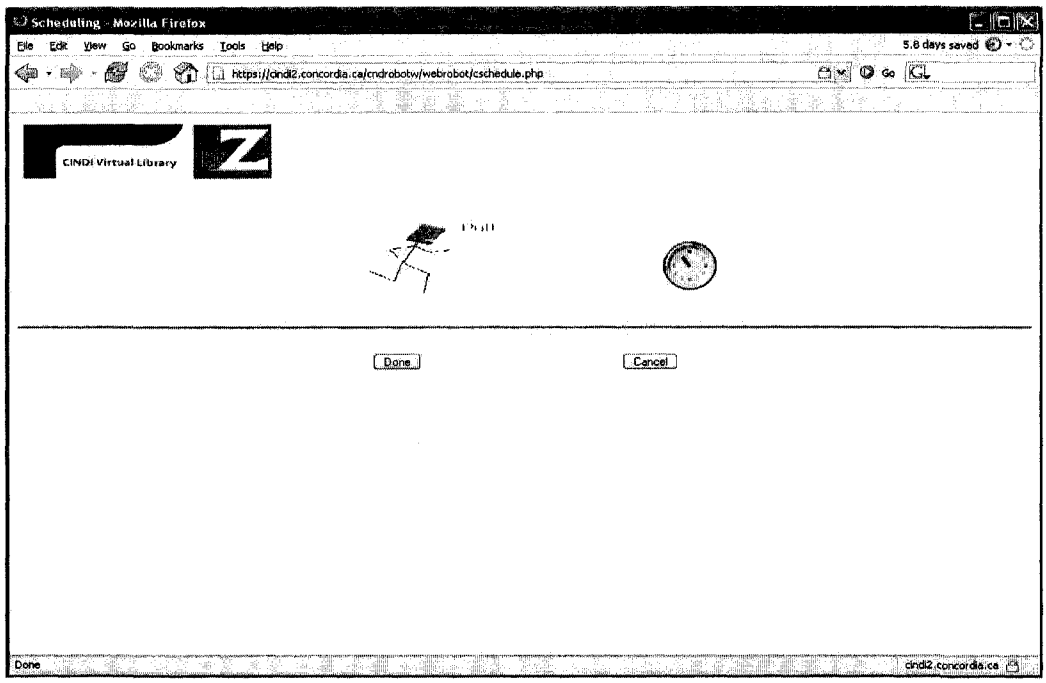

Figure 17 Schedule Process Page

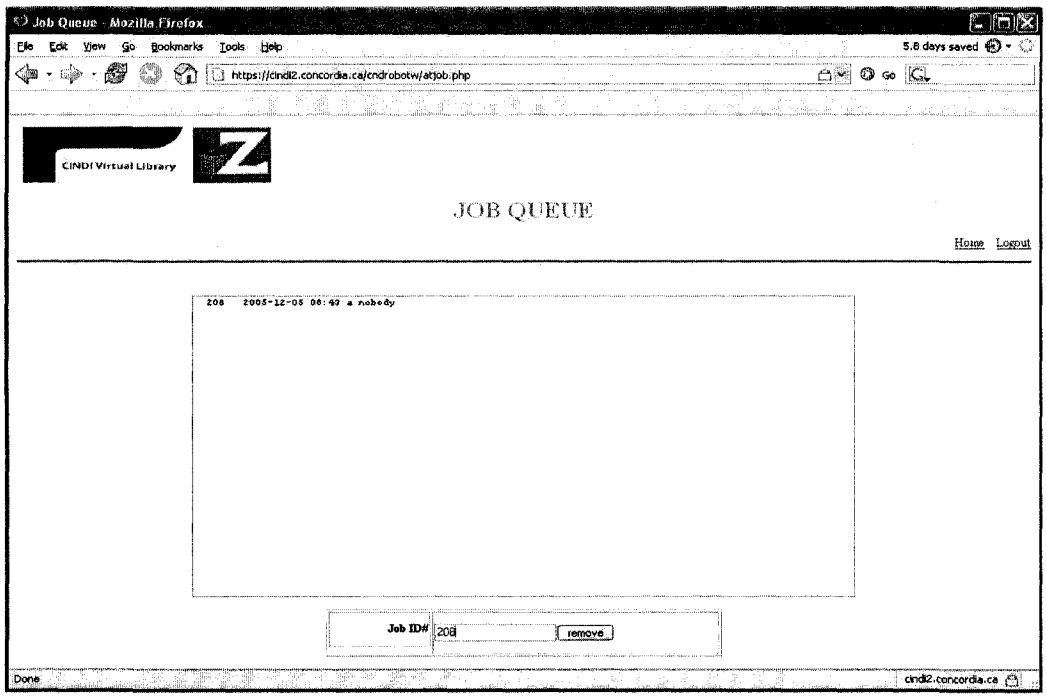

Figure 18 Job Queue Page

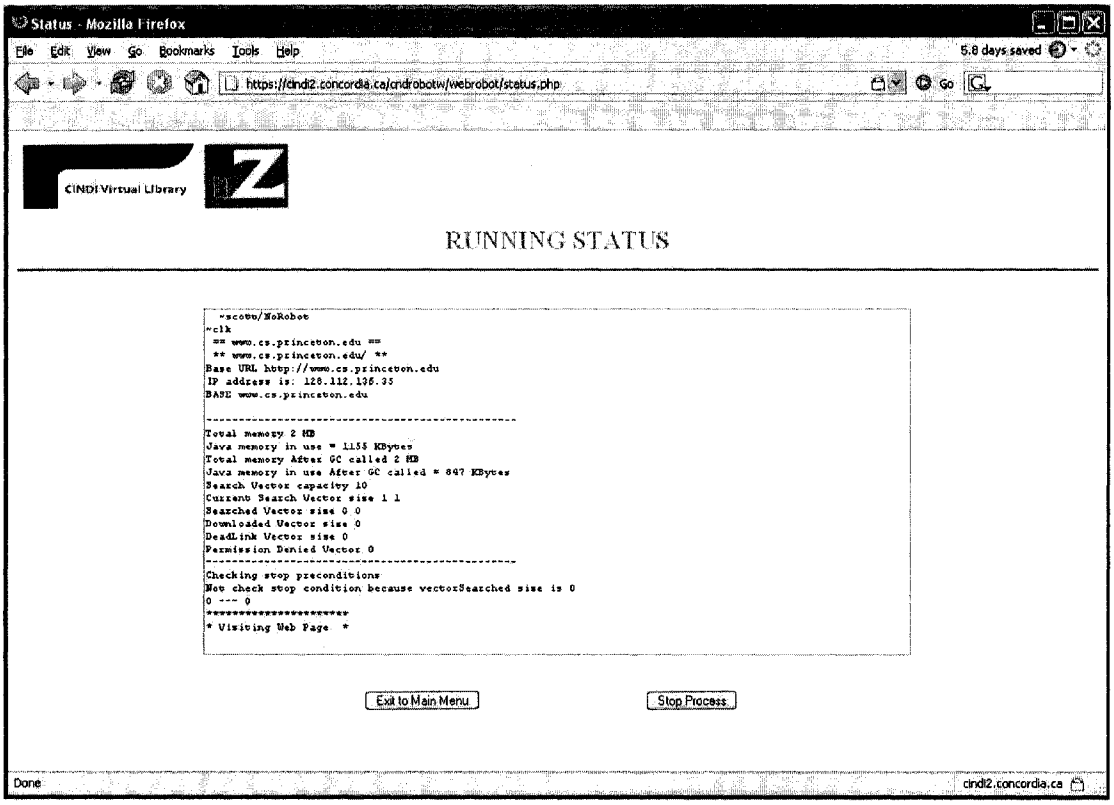

Figure 19 Running Status (Web Crawl) Page

Samples of running status pages for the Web crawl, file fetch, file filter and file restore are shown in Figure 19, 20, 21 and 22 respectively. Recall that the file fetcher and file filter start automatically and run sequentially after a crawling cycle finishes (section 3.3.1). However, the file fetcher and file filter can also be activated to run at any time as long as there are files to be fetched (section 3.2) and to be filtered. Users can run the file restore to check the stored documents against the records in DB when necessary. If any file were found missing, it would be re-fetched from its URL in our database.

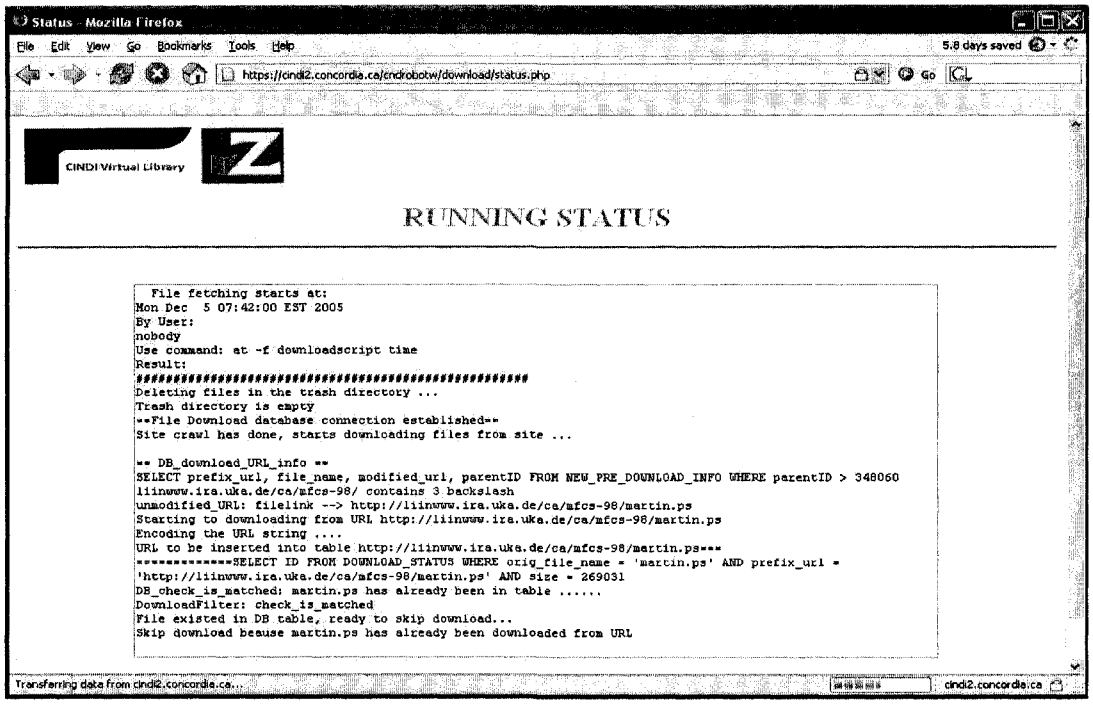

Figure 20 Running Status (File Fetch) Page

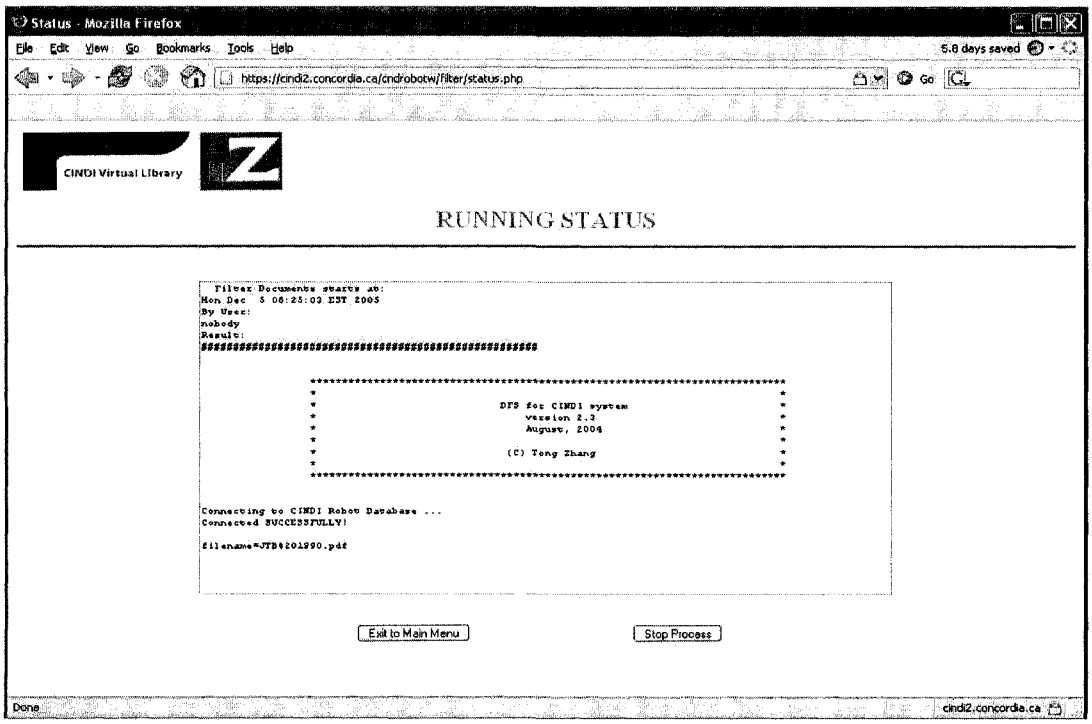

Figure 21 Running Status (File Filter) Page

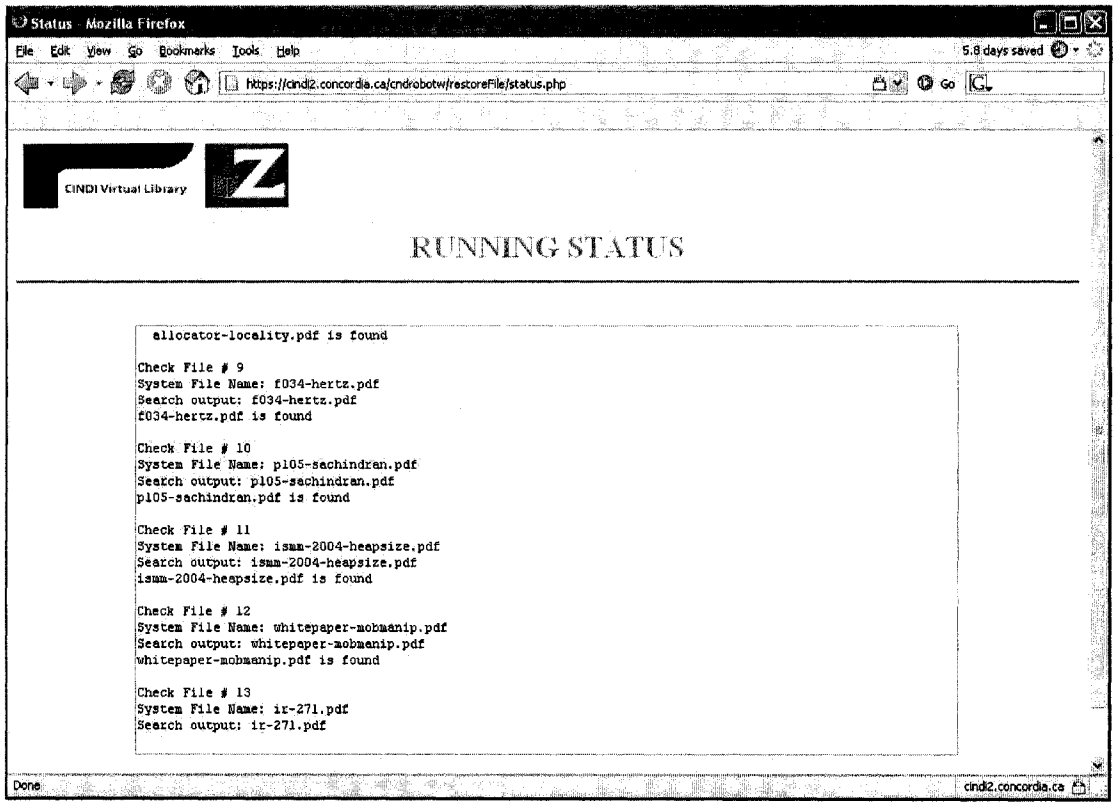

Figure 22 Running Status (File Restore) Page

To view the database tables of CNDROBOT, users can access to the database main page as shown in Figure 23. The table on the page contains all the DB tables that relates to the CNDROBOT. To view the records in a table, simply click the link corresponding to the table name. Figure 24 shows the records in the DOWNLOAD_STATUS table.

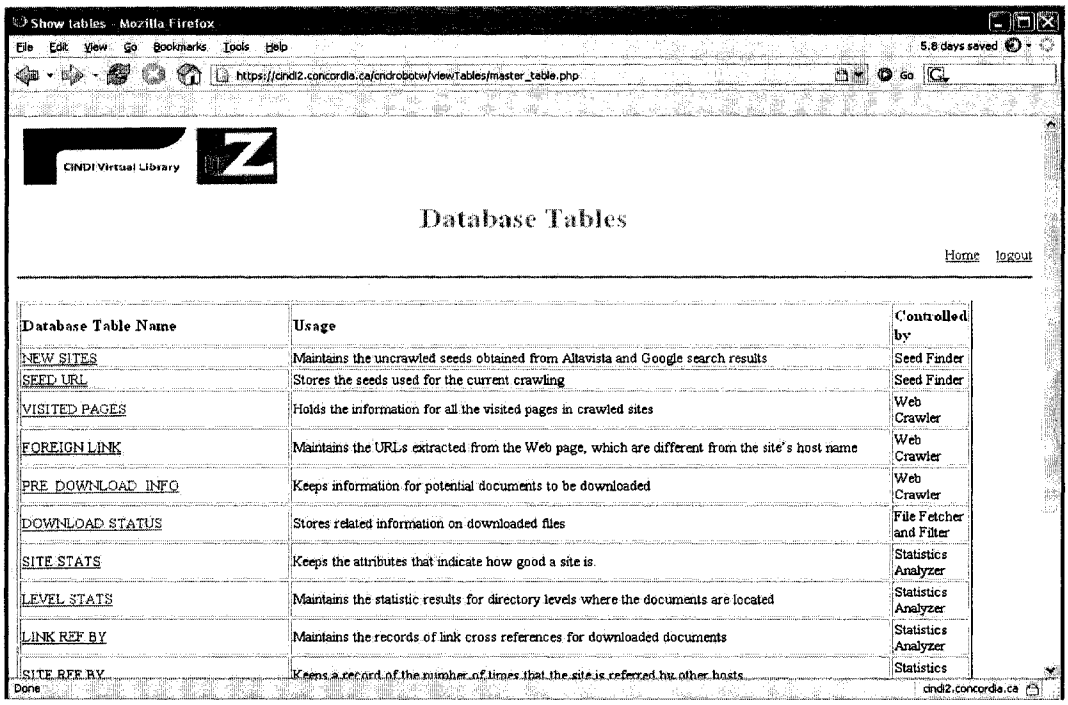

Figure 23 CINDI Database Main Page

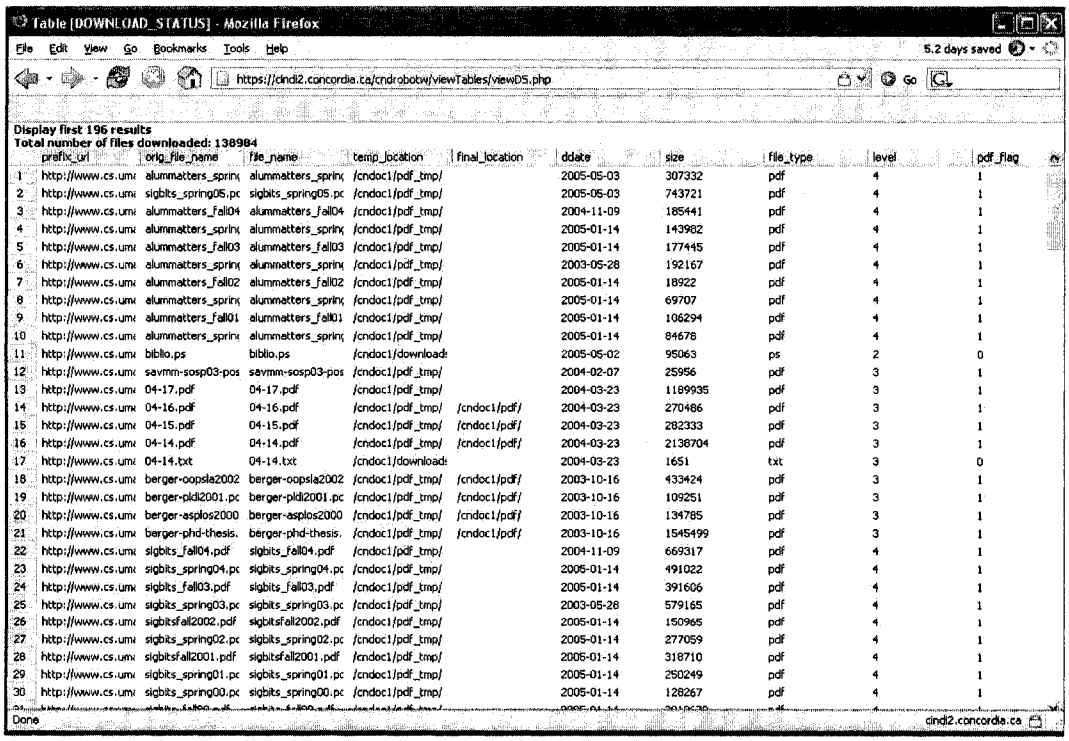

Figure 24 View Table DOWNLOAD_STATUS

Chapter 5

Experimental Results, Evaluation and Performance Improvement

In this chapter, we will present some preliminary test results and evaluations based on CNDROBOT's crawling experiences. Our experiments were performed on CINDI2 server (cindi2.concordia.ca) at Concordia University Computer Science Department. CNDROBOT ran an initial crawl and a subsequent crawl over a period of 35 days. The initial crawl took approximately 18 days. The robot randomly retrieved and crawled 30 sites from 3,130 sites acquired by the seed finder. In the initial crawl, the robot visited 348,173 Web pages and downloaded 106,416 documents, of which 20,348 were accepted by the DFS. After the DFS filtered the downloaded documents and the statistics analysis and link analysis were performed, the robot selected 15 crawled sites and retrieved another 15 new sites for the subsequent crawling. The subsequent crawl took approximately 17 days with 348,258 Web pages visited and 155,195 documents discovered. The third crawl is in process and it has crawled 19 sites with 283,265 Web pages visited and 118,263 potential documents discovered.

The crawling process was not continuous during this period. The robot stopped and resumed two times due to a system reboot and a network failure. In addition, there were several crashes since, at that time, our web crawling program could not perform adequate error protection and corrections while parsing the HTML Web pages that were not well written. After each crash, changes to the program were implemented and the robot was restarted at a resuming point. So far, the robot can run smoothly without interruption.

The information and data collected through the initial crawl are primarily used to test the performance of the Web crawler, file fetcher, statistic analyzer and link analyzer. The information gathered from the subsequent crawl is used to test the performance improvement over the initial crawl.

5.1 Experiments on Seed finder

The seed finder extracts 1,295 seeds using the phrase "computer science department" and obtains a total of 2,419 seeds after submitting "computer science publications" query to the Google and AltaVista search engines. Finally, a total of 3,130 unique seeds are acquired and stored into the NEW_SITES table after submitting the query "computer science technical reports".

As mentioned in section 3.3.3.1, seed finder parses the AltaVista's Web pages twice in order to discover real seeds from the mixtures. To test the accuracy of identifying real seeds, we ran the seed finder using the keyword phrase "computer science department". After the first parse, the seed finder extracted 1,035 links and stored them in the TEMP_SEED_URL table. After the second parse 840 seeds were spotted and 195 sponsored and redundant links were removed. We manually verified each seed to determine that none of them was sponsored link and there is no redundant link in the table.

5.2 Experiments on Web Crawl

From the NEW_SITES table, 30 new seed sites for the initial crawl are randomly retrieved. They are stored in the SEED_URL table as given in Table 25. Since the seeds for the initial crawl are the new sites, they have a value of 0 for the attribute "Is_new_seed" All other attributes in the table are set to either 0 or NULL before the robot starts. The crawling time for a site depends on the quantity and average size of Web pages in the site. On average, crawling a site takes approximately 15 hours for the initial crawl and 14 hours for the subsequent crawl.

ID	Link	Host_name	Is_new_seed	Resume_flag	SDATE	EDATE	Num_ref_by
1	www.cs.umass.edu	NULL	0	0	0000-00-00 00:00:00	NULL	0
2	www.cs.concordia.ca	NULL	0	0	0000-00-00 00:00:00	NULL	0
3	www.cs.indiana.edu	NULL	0	0	0000-00-00 00:00:00	NULL	0
4	www-cs.stanford.edu	NULL	0	0	0000-00-00 00:00:00	NULL	0
5	www.cs.cornell.edu	NULL	0	0	0000-00-00 00:00:00	NULL	0
6	www.cs.cmu.edu	NULL	0	0	0000-00-00 00:00:00	NULL	0
7	www.cs.umd.edu	NULL	0	0	0000-00-00 00:00:00	NULL	0
8	www.cs.uiuc.edu	NULL	0	0	0000-00-00 00:00:00	NULL	0
9	www.cis.upenn.edu	NULL	0	0	0000-00-00 00:00:00	NULL	0
10	www.cs.purdue.edu	NULL	0	0	0000-00-00 00:00:00	NULL	0
11	www.cs.unc.edu	NULL	0	0	0000-00-00 00:00:00	NULL	0
12	www.cs.toronto.edu/DCS/index.html	NULL	0	0	0000-00-00 00:00:00	NULL	0
13	www.cs.columbia.edu	NULL	0	0	0000-00-00 00:00:00	NULL	0
14	liinwww.ira.uka.de/bibliography	NULL	0	0	0000-00-00 00:00:00	NULL	0
15	dmoz.org/Computers/Computer_Science/Publications	NULL	0	0	0000-00-00 00:00:00	NULL	0
16	www.elsevier.com	NULL	0	0	0000-00-00 00:00:00	NULL	0
17	www.computer.org	NULL	0	0	0000-00-00 00:00:00	NULL	0
18	www.sciencedirect.com	NULL	0	0	0000-00-00 00:00:00	NULL	0
19	www.cs.virginia.edu/studpubs	NULL	0	0	0000-00-00 00:00:00	NULL	0
20	www.cs.kent.ac.uk	NULL	0	0	0000-00-00 00:00:00	NULL	0
21	cjtcs.cs.uchicago.edu	NULL	0	0	0000-00-00 00:00:00	NULL	0
22	www.dmtcs.org	NULL	0	0	0000-00-00 00:00:00	NULL	0
23	www.mcs.vuw.ac.nz/comp/Publications/	NULL	0	0	0000-00-00 00:00:00	NULL	0
24	reports-archive.adm.cs.cmu.edu/cs.html	NULL	0	0	0000-00-00 00:00:00	NULL	0
25	www.cs.bu.edu/techreports/	NULL	0	0	0000-00-00 00:00:00	NULL	0
26	cs.anu.edu.au/techreports/	NULL	0	0	0000-00-00 00:00:00	NULL	0
27	www.cs.arizona.edu/research/reports.html	NULL	0	0	0000-00-00 00:00:00	NULL	0
28	www.cs.rpi.edu/research/tr.html	NULL	0	0	0000-00-00 00:00:00	NULL	0
29	www.cs.umich.edu	NULL	0	0	0000-00-00 00:00:00	NULL	0
30	www.lib.utk.edu/refs/computersci	NULL	0	0	0000-00-00 00:00:00	NULL	0

Table 25 SEED_URL Table of Initial Crawl

Not all the URLs encountered could be downloaded and parsed due to various HTTP errors as shown in Table 26. Pages moved temporarily or permanently are the major types of HTTP errors. The permission denied error mostly occurs due to the access control of links. The following table illustrates the distributions of HTTP errors, which occurred in 61,125 invalid URLs and in the total number of URLs encountered in the initial crawl.

HTTP Errors	*Number of Invalid URLs*	*Percentage I Invalid URLs*	*Percentage II Total Visited Pages*
Page has no content	*795*	*1.30%*	*0.23%*
Dead link (page moved)	*38,564*	*63.10%*	*11.08%*
Permission denied	*21,759*	*35.60%*	*6.25%*
Other	*7*	*0.00%*	*0.00%*
Total	*61,125*	*100%*	*17.56%*

Table 26 Distributions for HTTP Errors

The column *percentage I* represents the percentage of the number of invalid URLs with one type of HTTP error out of 61,125 invalid URLs. The column *percentage II* demonstrates the percentage of the number of invalid URLs with one type of HTTP error out of the total of 348,173 URLs encountered in the initial crawl.

As discussed in section 3.2, the links extracted from the Web page can be classified as links in the domain and foreign links. CNDROBOT, as a focused crawler, visits and parses only the Web pages within the seed site domain. From the 30 sites, 117,875 foreign links were extracted and this accounts for 117,875 / 466,048 * 100% = 25.29% of the total hyperlinks. Since crawling a Web page takes half a second on average, visiting and parsing all those foreign links takes approximately 117,875* 0.5 = 58,937 seconds = 16 hours. To test the time that can be saved, we picked www.cs.concordia.ca as a sample seed and ran the robot on it on June 1 and June 6, 2005 respectively. In the first run, all the extracted links including the foreign links were visited. The total number of visited Web pages was 15,045 and it took about 8.5 hours to complete the crawling. In the second run, we tuned the robot to exclude crawling the foreign links and the crawling process took about 3 hours and 10 minutes for the total of 9,578 visited Web pages. From this test result, we can see that a significant amount of time can be saved by analyzing the crawl boundary and avoiding irrelevant regions of the Web.

5.3 Experiments on File Fetch

After the initial crawling, we downloaded a total of 106,416 documents from the 30 seed sites. The types of files that CNDROBOT accepted to download are pdf, ps, doc, txt, html, tex, latex, ppt, xml, and rtf. Almost half of the downloaded documents are in PDF format as given in Figure 25.

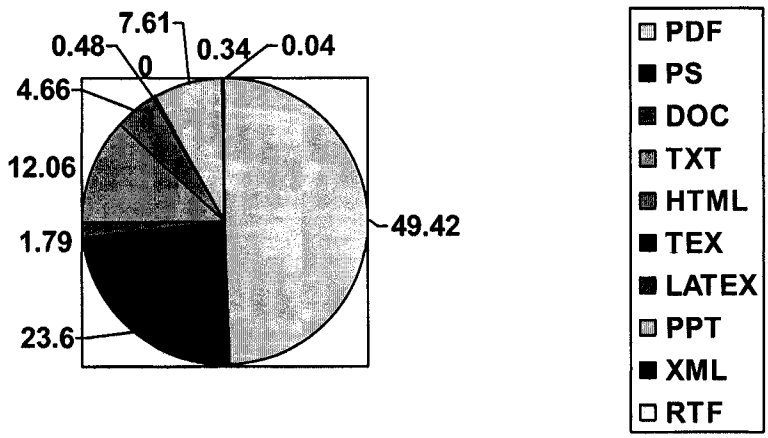

Figure 25 Distributions of Downloaded File Types

5.3.1 Functional Tests

To assess the effectiveness of the file fetcher, we examined the log file after the file fetching process was complete. First we randomly picked 16 downloaded documents in the DOWNLOAD_STATUS table and then found their detailed download descriptions in the log file. We verified the description with the information in the tables that were updated by the File Fetcher. We also ensured the existence of the documents and that they had correct file names under the right directories. If a digital signature check was performed and two files were identical, we verified that the duplicated file was removed from the system. Below is an example of how we performed the functional test for the file fetcher.

The document with ID 41047 is in the sample set. Its record in the DOWNLOAD_STATUS table was first retrieved and shown as follows.

ID	prefix_url	orig_file_name	file_name	temp_location	final_location	ddate	size	file_type	level	pdf_flag	ashg_flag	filter_flag	is_diff_format	is_renamed	parentID	num_ref_by
41047	http://www.cs.umd.edu/class/fall2003/cmsc434-0201/Handouts/	XeroxStar.pdf	XeroxStar0.pdf	/cndoc1/pdf_tmp/	NULL	2003-09-08	3382463	pdf	5	1	0	0	2	1	121210	-1

Table 277 Document ID# 41047 in DOWNLOAD_STATUS Table

```
Starting to downloading from URL http://www.cs.umd.edu/class/fall2003/cmsc434-0201/Handouts/XeroxStar.pdf
URL to be inserted into table http://www.cs.umd.edu/class/fall2003/cmsc434-0201/Handouts/XeroxStar.pdf===
=============SELECT ID FROM DOWNLOAD_STATUS WHERE orig_file_name = 'XeroxStar.pdf' AND prefix_url =
'http://www.cs.umd.edu/class/fall2003/cmsc434-0201/Handouts/' AND size = 3382463
DB_check_is_matched: XeroxStar.pdf This file not yet downloaded
Is not contained in the table
=============SELECT ID FROM DOWNLOAD_STATUS WHERE orig_file_name LIKE 'XeroxStar.%' AND prefix_url LIKE
'http://www.cs.umd.edu/class/fall2003/cmsc434-0201/Handouts/%'
DB_check_file_filter: XeroxStar There is no such file name in same page
=============SELECT ID FROM DOWNLOAD_STATUS WHERE file_name = 'XeroxStar.pdf' AND temp_location = '/cndoc1/pdf_tmp/'
DB_check_file_name: XeroxStar.pdf is already in table ..............................
trying to change file name to XeroxStar0.pdf
=============SELECT ID FROM DOWNLOAD_STATUS WHERE file_name = 'XeroxStar0.pdf' AND temp_location = '/cndoc1/pdf_tmp/'
DB_check_file_name: XeroxStar0.pdf There is no duplication of file name
Need to further check file signature
Set flag to 1
retrieving location for id 40000
retrieving temp_location
File in temp_location /cndoc1/pdf_tmp/
MD5 (/cndoc1/pdf_tmp/XeroxStar.pdf) = e03dc4d4f0e1de82abcaea2af8b1d844
MD5 (/cndoc1/pdf_tmp/XeroxStar0.pdf) = e03dc4d4f0e1de82abcaea2af8b1d844
output1  e03dc4d4f0e1de82abcaea2af8b1d844

output2  e03dc4d4f0e1de82abcaea2af8b1d844

Two files are the same
Database connection established
File Length: 30
http://www.cs.umd.edu/class/fall2003/cmsc434-0201/Handouts/XeroxStar.pdf:
  File Name:  XeroxStar0.pdf
  In the directory of : /cndoc1/pdf_tmp/XeroxStar0.pdf
  Content Type: application/pdf
  Content Length: 3382463
  Last Modified: Mon Sep 08 16:50:20 EDT 2003
  Expiration: 0
  Content Encoding: null

(1) Test if the document has been refered before
=============SELECT ID FROM DOWNLOAD_STATUS WHERE orig_file_name = 'XeroxStar.pdf' AND num_ref_by > 0
DB_check_is_referred: XeroxStar.pdf This file not referred yet
(2) Start to calculate the num_ref_by ...
===SELECT COUNT(*) FROM DOWNLOAD_STATUS WHERE orig_file_name = 'XeroxStar.pdf'===
2
(3) updating the Document_Ref_By table
===SELECT ID FROM DOWNLOAD_STATUS WHERE orig_file_name = 'XeroxStar.pdf' ORDER BY ddate ASC===
=============UPDATE DOWNLOAD_STATUS SET num_ref_by = -1 WHERE ID = 41047
num_ref_by has been updated to -1
40000 41047
DB_set_ref_num 2 documentID 40000
=============UPDATE DOWNLOAD_STATUS SET num_ref_by = 1 WHERE ID = 40000
num_ref_by has been updated to 1
Remove the file /cndoc1/pdf_tmp/hasofer_s0.pdf from the system
```

Figure 26 Sample of File Fetching Log File

The description for fetching this document was found in the log file. By looking through the description, we first determined that the file "XeroxStar.pdf" was a new document downloaded from the file link http://www.cs.umd.edu/class/fall2003/cmsc434-0201/Handouts/XeroxStar.pdf and that it had no other file formats under the directory of "Handouts". To verify the correctness of the first fact, we queried the DOWNLOAD_STATUS table to ensure that no record with such a prefix url, file name

and size were found. This also tested the effectiveness of our download policy, i.e. no files are downloaded twice from the same link. To verify that the downloaded file has no other file formats, we executed a query to make sure there was no record with the file name "XeroxStar", file types other than "pdf" and those with the prefix url http://www.cs.umd.edu/class/fall2003/cmsc434-0201/Handouts/.

The file was treated as new although there was a document with the same file name that had been downloaded before. The file name was changed from the original file name "XeroxStar" to "XeroxStar0" and downloaded to the "/cndoc1/pdf_tmp" directory. Because it has the same file name and file size as the one downloaded before, document ID number 40000 as shown in Table 28, the file digital signatures for both files were checked and their file signatures were matched.

ID	prefix_url	orig_file_name	file_name	temp_location	final_location	ddate	size	file_type	level	pdf_flag	ashg_flag	filter_flag	is_diff_format	is_renamed	parentID	num_ref_by
40000	http://www.cs.umd.edu/class/spring2005/cmsc434/Handouts/	XeroxStar.pdf	XeroxStar.pdf	/cndoc1/pdf_tmp/	NULL	2003-09-08	3382463	pdf	5	1	0	2	2	2	120691	1

Table 28 Document ID# 40000 in DOWNLOAD_STATUS Table

To verify that the document with ID 41047 had been deleted, we first found the directory in which it was stored. Since its filter flag had not been updated, the file with the changed name "XeroxStar0.pdf" would be in the directory of "/cndoc1/pdf_tmp/" if it had not been deleted. We could not find it and confirmed that this document was deleted.

Since the last modified dates for these two files are the same, the first downloaded document (ID # 40000) is deemed to be the referee and the document (ID # 41047) is considered as the referrer. We went to the DOCUMENT_REF_BY table to ensure the record for both documents was there.

```
mysql> SELECT * FROM DOCUMENT_REF_BY WHERE SID = 40000 AND FID = 41047;
+------+-------+-------+
| ID   | SID   | FID   |
+------+-------+-------+
| 1878 | 40000 | 41047 |
+------+-------+-------+
1 row in set (0.02 sec)
```

Table 29 Record in DOCUMENT_REF_BY Table

The overall test results for 16 sample documents are summarized in Table 30. The functions of file fetching are proven to be effective and achieve 100 percent accuracy for these test documents.

Test Document ID	*File Name*	*Tested for Document Redundancy& File Formats*	*Tested for Changed File Name & Location*	*Tested for Digital Signature & File Removal*	*Test Success (Y/N)*
308	3dkernel_pdbresults.pdf	Y	Y	N/A	Y
1219	voting-experts.pdf	Y	Y	N/A	Y
3092	icml-1999-unify.ps	Y	Y	Y	Y
4192	DL-99-HM.pdf	Y	Y	N/A	Y
4523	appoint-w98.txt	Y	Y	N/A	Y
6230	midrange.pdf	Y	Y	Y	Y
12435	419-sp05-10.2-transport-v3.pdf	Y	Y	N/A	Y
14240	CS414Section2.pdf	Y	Y	N/A	Y
24322	sophomoric6x9.pdf	Y	Y	N/A	Y
39747	31-3.pdf	Y	Y	N/A	Y
41047	XeroxStar.pdf	Y	Y	Y	Y
68943	Hirst-NearSynonyms-95.ps	Y	Y	Y	Y
72352	tsd2002bib.pdf	Y	Y	Y	Y
88270	TCAD_group_final.pdf	Y	Y	N/A	Y
101482	lec13print.ps	Y	Y	Y	Y
102465	search-tutorial.pdf	Y	Y	N/A	Y

Table 30 Sample of Test Documents and Testing Results

5.3.2 Correlation Analysis

Document Size and Document Quality

Since CINDI library only collects scientific documents such as research papers and technical reports and the size of those documents must be big enough to contain a certain amount of information, intuitively we know that there should be a relationship between the size and amount of information.

To find the relationship, we did an experiment on 52,552 downloaded PDF documents. We chose the PDF files as our test documents because they are more representative in terms of volume and diversity of document size. These test documents were processed by DFS, which has a filtering accuracy of 98% [TZ04], and the filtered information was updated in the DOWNLOAD_STATUS table. We counted the number of documents accepted and rejected by the filter and calculated the percentage of valid documents over the total number of downloaded documents for five different document sizes. The results are summarized in the table below.

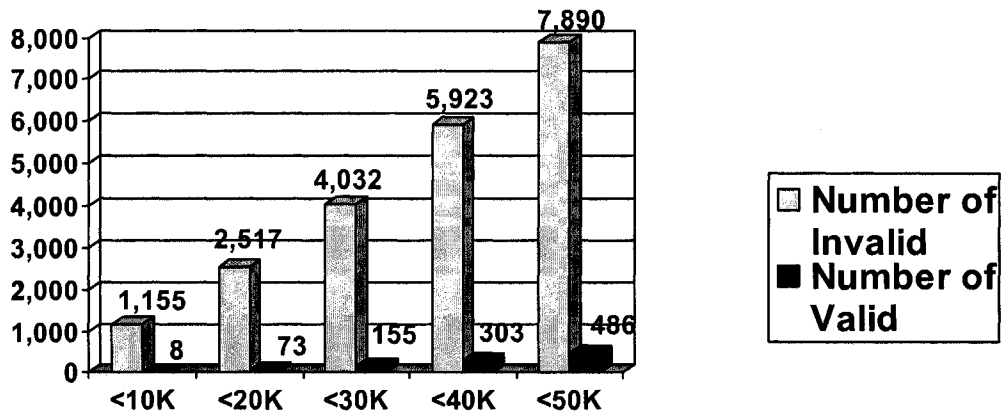

Figure 27 Document Size and Document Quality

We perceive that the number of accepted documents increases as the size of the documents increases. In addition, the experimental outcome has an implication for us when designing and implementing the file fetcher, i.e. not downloading a document if its

size is less than a certain number of bytes. A further test was made to determine the number of bytes below which all downloaded documents are invalid. We find out that there are 829 documents whose sizes are less than 8K bytes and all of them are invalid. Therefore, we set the 8k as the cut-off point of skip downloading. By doing this, we can save time not only for file fetching but also for file filtering.

Directory Levels and Number of Documents Found

As discussed in section 3.2 and shown in Table 11, there is a relationship between the level of directory where documents are located and the quantity of documents found at that level. For the initial crawl, 97% of the documents found and downloaded have URLs with directory levels of 2, 3, 4, and 5. No documents are found at URLs with a level greater than 7. Therefore, further search to deeper levels in a site is not cost effective and may even prove fruitless.

5.4 Experiments on Statistics Analyzer

After the initial crawl, 43,854 distinct hosts are found in the FOREIGN_LINK table. There are 43,322 seed candidates after crawled sites and existing seeds are removed from these hosts. A total of 9,125 candidates qualify as new seeds and are inserted into the NEW_SITES table

To test the effectiveness of seed selection by using the rules with the heuristics of "keyword in Web page" and "anchor text in Web page", we took the first 300 entries from the FOREIGN_LINK table. The first 37 entries of the sample set are given in the table below.

```
mysql> SELECT * from FOREIGN_LINK WHERE FID < 300;
+-----+-----------------------------------------------------------------+-----------------------------+-----+
| FID | url                                                             | host_name                   | PID |
+-----+-----------------------------------------------------------------+-----------------------------+-----+
|   1 | http://www.umass.edu                                            | www.umass.edu               |   1 |
|   2 | http://umass.edu                                                | umass.edu                   |   1 |
|   3 | http://umass.edu/umhome/policies/                               | umass.edu                   |   1 |
|   4 | http://www.cra.org/                                             | www.cra.org                 |   1 |
|   5 | http://www.factfinder.census.gov                                | www.factfinder.census.gov   |   5 |
|   6 | http://ciir.cs.umass.edu/personnel/croft.html                   | ciir.cs.umass.edu           |   6 |
|   7 | http://www-psl.acso.umass.edu/cgi-bin/inquiry/grading_display.pl | www-psl.acso.umass.edu      |   6 |
|   8 | http://www.umass.edu/admissions/                                | www.umass.edu               |   6 |
|   9 | http://www.amherstcommon.com/                                   | www.amherstcommon.com       |   7 |
|  10 | http://www.noho.com/                                            | www.noho.com                |   7 |
|  11 | http://www.peterpanbus.com/                                     | www.peterpanbus.com         |   7 |
|  12 | http://www.valleytransporter.com                                | www.valleytransporter.com   |   7 |
|  13 | http://www.amtrak.com                                           | www.amtrak.com              |   7 |
|  14 | http://manic.cs.umass.edu/                                      | manic.cs.umass.edu          |  10 |
|  15 | http://macdb.cs.umass.edu/cal/                                  | macdb.cs.umass.edu          |  12 |
|  16 | http://www.umassalumni.com/                                     | www.umassalumni.com         |  15 |
|  17 | http://www-edlab.cs.umass.edu/                                  | www-edlab.cs.umass.edu      |  17 |
|  18 | http://www.nsm.umass.edu                                        | www.nsm.umass.edu           |  18 |
|  19 | http://www.umass.edu/umhome/visit_campus                        | www.umass.edu               |  18 |
|  20 | http://www.umass.edu/umhome/events/index.php                    | www.umass.edu               |  18 |
|  21 | http://calendar.fivecolleges.edu/FiveCol/calendrome.cgi          | calendar.fivecolleges.edu   |  18 |
|  22 | http://www.amherstarea.com                                      | www.amherstarea.com         |  18 |
|  23 | http://northamptonuncommon.com                                  | northamptonuncommon.com     |  18 |
|  24 | http://www.valleyvisitor.com/                                   | www.valleyvisitor.com       |  18 |
|  25 | http://www.masslive.com/                                        | www.masslive.com            |  18 |
|  26 | http://www.gazettenet.com/                                      | www.gazettenet.com          |  18 |
|  27 | http://www-all.cs.umass.edu/                                    | www-all.cs.umass.edu        |  20 |
|  28 | http://www-anw.cs.umass.edu/                                    | www-anw.cs.umass.edu        |  20 |
|  29 | http://laser.cs.umass.edu/                                      | laser.cs.umass.edu          |  20 |
|  30 | http://www-robotics.cs.umass.edu                                | www-robotics.cs.umass.edu   |  20 |
|  31 | http://dis.cs.umass.edu/                                        | dis.cs.umass.edu            |  20 |
|  32 | http://anytime.cs.umass.edu/                                    | anytime.cs.umass.edu        |  20 |
|  33 | http://www-net.cs.umass.edu/                                    | www-net.cs.umass.edu        |  21 |
|  34 | http://ccbit.cs.umass.edu/ccbit                                 | ccbit.cs.umass.edu          |  21 |
|  35 | http://ccbit.cs.umass.edu/ckc                                   | ccbit.cs.umass.edu          |  21 |
|  36 | http://ripples.cs.umass.edu                                     | ripples.cs.umass.edu        |  21 |
|  37 | http://manic.cs.umass.edu/criccs                                | manic.cs.umass.edu          |  21 |
```

Table 31 Test Samples in FOREIGN_LINK Table

There are 184 distinct hosts. We found 19 of them had been already either crawled or discovered by the seed finder. The Web page contents of the remaining 165 hosts were compared with 174 domain keywords (see Appendix C) and 3 representative anchor texts in the RDVT table (see table 13). The initial selection rule can be written in the format below.

Number of Keywords: - K

Number of Anchor Texts: - A

Accepted: - $(K > 0 \wedge A > 0) \vee (K > 1 \vee A > 1)$

The statistics analyzer accepted 99 hosts as seeds and rejected 67 hosts. We manually validated the Web content of the accepted hosts and the results are summarized as follows.

1- Broad topic 2- Irrelevant content 3- Ambiguous topic

#	Host URL	Num of keyword match	Num of anchor text match	Is a valid seed (reason)
1	www.factfinder.census.gov	1	1	No (2)
2	www-edlab.cs.umass.edu	7	0	Yes
3	northamptonuncommon.com	2	0	No (2)
4	laser.cs.umass.edu	3	1	Yes
5	www-robotics.cs.umass.edu	2	0	Yes
6	dis.cs.umass.edu	7	1	Yes
7	ripples.cs.umass.edu	3	3	Yes
8	iesl.cs.umass.edu	5	2	Yes
9	kdl.cs.umass.edu	3	0	Yes
10	signl.cs.umass.edu	4	1	Yes
11	prisms.cs.umass.edu	4	1	Yes
12	www-ccsl.cs.umass.edu	3	0	Yes
13	sysbio.cs.umass.edu	2	0	Yes
14	www.apple.com	2	0	No (1)
15	research.microsoft.com	3	2	Yes
16	www.philly.com	5	0	No (2)
17	mas.cs.umass.edu	7	1	Yes
18	www.oit.umass.edu	3	0	Yes
19	www.analog.cx	3	0	No (3)
20	www.arenasoftware.com	2	0	No (2)
21	winscp.vse.cz	2	0	No (2)
22	www.w3.org	8	1	Yes
23	vip.oit.umass.edu	2	0	No (2)
24	www.ccsf.edu	5	0	No (1)
25	www.cs.wright.edu	3	0	Yes
26	cs.conncoll.edu	8	0	Yes
27	www.pvamu.edu	2	0	No (1)
28	www.adobe.com	5	0	Yes
29	www.clarkson.edu	2	0	No (1)
30	www.iis.sinica.edu.tw	2	0	Yes
31	homepages.inf.ed.ac.uk	1	2	Yes
32	www.hpl.hp.com	2	2	Yes
33	lass.cs.umass.edu	6	2	Yes
34	www.heaplayers.org	3	0	Yes
35	www.framingham.edu	4	0	No (1)
36	www.extension.harvard.edu	3	0	No (1)
37	www.umb.edu	1	1	No (1)
38	www.wpi.edu	4	0	No (1)
39	www.worcester.edu	3	0	No (1)
40	www.dcoss.org	5	0	Yes
41	www.comm.csl.uiuc.edu	3	0	Yes
42	www.podc.org	4	0	Yes
43	www.hipc.org	6	0	Yes
44	www.oracle.com	5	0	Yes
45	cise.nsf.gov	6	1	Yes
46	www.utexas.edu	2	0	No (1)
47	oregonstate.edu	2	0	No (1)
48	www.nsf.gov	4	1	No (1)

49	www.nist.gov	3	1	No (2)
50	www.georgetown.edu	2	0	No (1)
51	www.acm.org	5	1	Yes
52	patterns.projects.cis.ksu.edu	3	1	Yes
53	www.comlab.ox.ac.uk	0	2	Yes
54	www.kebe.com	2	0	No (2)
55	computationalcomplexity.org	4	0	Yes
56	boston.com	2	0	No (2)
57	j-bradford-delong.net	2	0	No (2)
58	atrios.blogspot.com	2	0	No (2)
59	headheeb.blogmosis.com	4	0	No (2)
60	fafblog.blogspot.com	4	0	No (2)
61	dailykos.com	3	0	No (2)
62	talkingpointsmemo.com	2	0	No (2)
63	andrewsullivan.com	5	0	No (2)
64	washingtonmonthly.com	4	0	No (2)
65	www.wiley.com	3	0	No (1)
66	www.bcs.rochester.edu	3	0	No (2)
67	www.seas.upenn.edu	3	0	Yes
68	suma.cs.umass.edu	3	0	Yes
69	www.research.ibm.com	5	0	Yes
70	wosp.zeesource.net	2	0	Yes
71	msdnaa.oit.umass.edu	4	0	Yes
72	www.mass.gov	2	0	No (2)
73	www.icra2005.org	2	0	No (3)
74	www.roboticsconference.org	4	0	Yes
75	www.ee.virginia.edu	2	0	Yes
76	www.virginia.edu	2	0	No (1)
77	www.sigmobile.org	3	0	Yes
78	mmcn05.cse.nd.edu	6	0	Yes
79	www.issnip.org	2	0	Yes
80	www.ece.wisc.edu	1	1	Yes
81	www.ieee-infocom.org	2	0	Yes
82	www.icar2005.org	3	0	No (3)
83	www-sensorimotor.cs.umass.edu	2	1	Yes
84	www.cornell.edu	1	1	No (1)
85	www.siam.org	3	0	No (3)
86	www.gf.org	2	0	No (2)
87	www.soe.ucsc.edu	5	2	Yes
88	www.ntu.edu	4	0	No (1)
89	mallet.cs.umass.edu	7	0	Yes
90	www-ccs.cs.umass.edu	6	0	Yes
91	www.dbai.tuwien.ac.at	3	1	Yes
92	sol.rutgers.edu	3	1	No (3)
93	www.ieee.org	3	0	Yes
94	computer.org	5	0	Yes
95	data.cs.washington.edu	4	0	Yes
96	www.asprs.org	3	0	No (3)
97	acmmm05.comp.nus.edu.sg	4	0	Yes
98	www.nossdav.org	2	0	Yes
99	www.ornl.gov	4	0	No (2)

Table 32 Validations of Accepted Hosts

We did not accept a host as a seed if the Web content of the host was too broad, which means the robot needs to search extensive numbers of irrelevant Web pages before reaching relevant ones. For example, the home page of a university, e.g. www.wpi.edu contains the URLs to Web pages of all the faculties and departments. To reach the Web pages of the computer science department, the Web pages of all other departments will be crawled as well. We do not accept this type of host as the seed because crawling over it is not cost effective. In addition, for the host whose Web content is irrelevant or ambiguous, e.g. the Web site for the imaging and geospatial information society at www.asprs.org might contains some computer science related information, but most of the information is unrelated and we do not accept it either.

	Valid	***Invalid***			***Total***
		Broad	*Irrelevant*	*Ambiguous*	
Number of Hosts	55	17	21	6	99
Percentage	55.55%	17.18%	21.21%	6.06%	100%

Table 33 Statistics on 1st Seed Selection Test

5.4.1 Improvement on the Selection Rule

To improve the selection rule, we first examined the first term of the rule, which is ($K > 0 \wedge A > 0$). We found that 25 hosts have at least one for both of the number of keyword matches and the number of anchor text matches. Of these, two of them were not accepted. The effective rate is 92% if we use only the first term to evaluate. The second term ($K > 1 \vee A > 1$) has an effective rate of only 32/74 * 100% = 43%. This also implies that using keywords or anchor texts alone cannot determine the relevance of a Web page. A further examination was made of the keywords found in the Web pages of rejected hosts. We found that the most frequent keywords in these pages were "general", "miscellaneous", "languages", and "learning". Although these keywords are classified as domain keywords by Inspec, they are ambiguous and result in faulty judgement as to the relevance of a

page. We removed them with some other keywords such as "social issues" and "public policy issues" and retested the same sample set. The statistical results for the second test are given below.

	Valid	*Invalid*			*Total*
		Broad	*Irrelevant*	*Ambiguous*	
Number of Hosts	54	3	1	0	58
Percentage	93.10%	5.17%	1.73%	0.00%	100%

Table 34 Statistics on 2nd Seed Selection Test

The total effective rate has been improved from 55.55% to 93.10%. One valid host www.ieee-infocom.org was lost due to the keyword removal. Incorrect selections were significantly reduced from 44.45% to 6.90%. The effective rate is basically satisfactory.

5.5 Experiments on Documents Discoverability

To test the ability of CNDROBOT to discover useful documents, we randomly selected 30 computer science related research papers out of 20,348 accepted documents in our database. We chose two well-known computer science digital libraries, CiteSeer and ACM as the benchmarks and manually search each document in them. The test result is shown in Figure 28.

#	Paper Title	Author	Year Published	Organization	CiteSeer	ACM
1	Memory Ordering: A Value-Based Approach	Harold W. & Mikko H.	2004	Univ. of Wisconsin -Madison	N	Y
2	Concise Descriptions of Subsets of Structured Sets	Alberto O. & Ken Q.	2003	Univ. of Toronto	N	Y
3	Linking Shared Segments	W.E. Garrett et al.	1993	Univ. Rochester	Y	N
4	Interfaces for Modular Feature Verification	Harry C., Shriram K., & Kathi F.	2002	Brown Univ.	Y	N
5	Computer-assisted kinematic Tolerance Analysis of a Gear Selector Mechanism with the Configuration Space Method	Elisha S., Leo J., & Ralf S.	1999	Purdue Univ., Ford werke AG	N	N

6	Safety in Automated Trust Negotiation	William H. & NingHui Li	2004	George Mason Univ.	N	N
7	Improvements to Graph Coloring Register Allocation	Preston B., Keith D. & Linda T.	1994	Rice Univ.	Y	Y
8	System E: Expansion Variables for Flexible Typing with Linear and Non-linear Types and Intersection Types	Sebastien C, Jeff P. et al.	2004	Boston Univ. & Heriot-Watt Univ.	Y	N
9	Generating the Envelope of a Swept Trivariate Solid	Claudia M. & Kenneth I.	1999	Univ. of California	Y	N
10	An Implicit Finite Element Method for Elastic Solids in Contact	Gentaro H., Susan F. et al.	2001	Univ. North Carolina	Y	N
11	InfoVisExplorer	Jaroslav T. & Grant P.	2004	Georgia Institute of Tech.	N	N
12	OSGAR: A Scene Graph with Uncertain Transformations	Enylton M., Blair M. & Simon J.	2004	Naval Research Lab	N	N
13	Teallach: a Model-Based User Interface Development Environment for Object Databases	Tony G., Peter J. et al.	1999	Univ. Manchester	Y	N
14	Shape Estimation from Support and Diameter Functions	Amyn P., Peyman M. & Richard J	2004	Univ. of California	N	N
15	Applying Metric-Trees to Belief-Point POMDPs	Joelle P. Geoffrey G. & Sebastian T.	2003	Carnegie Mellon Univ.	Y	N
16	Processor Power Reduction Via Single-ISA Heterogeneous Multi-Core Architectures	Rakesh K., Keith F. et al.	2003	HP Labs	Y	N
17	Low-Latency Music Software Using Off-The-Shelf Operating Systems	Eli B. & Roger B.	1998	Carnegie Mellon Univ.	N	N
18	Virtual Appliances for Deploying and Maintaining Software	Constantine S., David Brumley & Ramesh C.	2003	Stanford Univ.	Y	N
19	LP Decoding Corrects a Constant Fraction of Errors	Jon F., Tal M. & Rocco A.	2003	Columbia Univ.	Y	N
20	Data Collection and Language Technologies for Mapudungun	Lori L., Rodolfo V. et al.	2002	Carnegie Mellon Univ.	Y	N
21	WebView Materialization	Alexandros L. & Nick R.	2000	Univ. of Maryland	Y	Y
22	Automatic Code Placement Alternatives for Ad-Hoc and Sensor Networks	Emin G., Rimon B. et al.	2001	Cornell University	Y	N

23	From Discourse Structures to Text Summaries	Daniel M.	1997	Univ. of Toronto	Y	N
24	The XP Customer Role in Practice: Three Studies	Angela M., Robert B. & James N.	2004	Victoria Univ. of Wellington	N	N
25	Cache As Filters: A New Approach to Cache Analysis	Dee A., Sally A. & Wulf A.	1998	Univ. of Virginia	N	N
26	Workload Characterization in Web Caching Hierarchies	GuangWei B. & Carey W.	2002	Univ. of Calgary	N	N
27	Semantics-Based Concurrency Control: Beyond Commutativity	Badrinath B. & Krithi R.	1992	Univ. of Massachusetts	N	Y
28	A Methodology for Controlling the Size of a Test Suite	Mary J., Rajiv G. & Mary L.	1993	Clemson Univ.	N	Y
29	TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications	Philip L., Nelson L. et. al.	2003	Intel Corp. & Univ. of California	Y	Y
30	CAFIXD: A Case-Based Reasoning Fixture Design Method. Framework and Indexing Mechanisms	Iain M. & Kevin R.	2004	Worcester Polytechnic Institute	N	N

Figure 28 Search results of CiteSeer and ACM Digital Libraries on Selected Research Papers

The search results show that for these 30 research papers found by CNDROBOT, CiteSeer has only 16 of them and ACM has only 7. At this stage, we cannot state that CINDI library with the support of CNDROBOT would be better than the other two in terms of the size of collection because the crawling process has not completed yet and an in-depth test has not been made. However, based on the results, we are confident that CNDROBOT has the ability to discover a large number of desired documents that others have not.

Chapter 6
Conclusion and Future Works

6.1 Conclusion

The gigantic size and the dynamic nature of the World Wide Web pose the unprecedented challenge for Web robots to locate and retrieve Web documents to build digital libraries. CNDROBOT has been constructed as a focused crawler to effectively discover the computer scientific documents of type research papers, theses, FAQs, academic papers and technical reports for CINDI digital library. These documents are mainly disseminated in institution sites, research organization sites and researcher's home pages. The goal of CNDROBOT is to acquire maximum amount of relevant documents while crawling a relatively small scale of the Web. To achieve this goal, we designed and implemented the CNDROBOT in five major components: the seed finder locates the trusted and topic-related sites (seeds) that are likely to contain the computer science literature. It submits to the Google and AltaVista search engines queries; parses the returned Web pages and extracts the seed sites from the pages. The Web crawler explores the Web pages in the seed sites to discover documents and potential new seeds. The file fetcher downloads the discovered documents to a local storage. The statistics analyzer analyzes previous crawling data, produces statistical results, discovers new seeds and selects seeds for the next crawl. The link analyzer provides recommendation for the URL prediction in future crawling based upon past crawling experiences.

6.2 Contribution of This Thesis

In this thesis, we have described the architecture, approach and implementation of the CNDROBOT and presented some heuristics and preliminary experiments. The objective of this thesis project is to crawl the Web and discover and collect potential documents desired by CINDI library. The design and implementation of CNDROBOT as well as the integration with DFS subsystem are the main contributions to the overall CINDI system.

For the seed finder, multi search engines were used to discover trusted sites for the Web crawler to start with. For the Web crawler, the Naïve crawling algorithm and knowledge based crawling algorithm were developed. The seeds rescheduling algorithm was applied by the statistics analyzer to select crawled sites to be revisited. The link analyzer was built to accumulate knowledge from previous crawling experiences. The file fetcher was implemented to effectively download the files and gather relevant file information. Finally, a secure, user-friendly Web application was implemented to allow users to control the CNDROBOT remotely through the Web.

6.3 Future Work

There are several promising areas for future work with the CNDROBOT. The first is to decrease the time for Web crawling and file fetching. The average crawling and downloading time is 15 hours for a site. Currently, the CNDROBOT crawls the sites and downloads the files sequentially, i.e. only one site and one file is crawled and downloaded at one time. One option is to reduce the time of Web crawling and file fetching by running several copies of the Web crawler and file fetcher simultaneously. For example, several copies of the Web crawler can run on one machine or multiple machines. However, the synchronization and coordination between these crawlers are needed so that the same site will not be crawled more than one time. The strategy, running two copies of file fetcher to download the files, is being implemented and tested. Another area for future work lies in discovering new sites and new documents. Currently, we discover new sites from the hosts of foreign links in prior crawling. One possibility is to explore new sites in downloaded documents since many of them contain hyperlinks to relevant topics in the context and references section and these hyperlinks might link to new sites or new documents.

Other areas of research involve improving the accuracy of new seed selection. As shown in section 5.4, the effective rate of correct selection is 93.10%. Although the result is basically satisfactory, it is possible to make a further improvement by investigating more incorrect selections and refining the selection rule.

Finally, for the crawling strategy, we adopted the simple breadth-first algorithm. At the other end crawler algorithm might involve more complex algorithm such as best-first algorithm. That algorithm can be implemented, tested and compared with current one.

References

[ADL] The ACM Digital Library, available at http://portal.acm.org/dl.cfm

[AS92] Andrew S. Tanenbaum. "Modern Operating Systems". Prentice Hall, New Jersey, 1992.

[AVA] AltaVista Search Engine, available at http://www.altavista.com

Baujard, O., Baujard, V., Aurel, S., Boyer, C., and Appel, R.D. "Trends in Medical Information Retrieval on the Internet," Computers in Biology and Medicine, 28, 1998, pp. 589-601.

[BC94] B.C. Desai, "A System for Seamless Search of Distributed Information Sources", May 1994, available at http://www.cs.concordia.ca/~bcdesai/web-publ/w3-paper.html

[CDL] California Digital Library, available at http://www.cdlib.org/

[CITE] Scientific Literature Digital Library, available at http://citeseer.ist.psu.edu/

[DB00] Davison, B. D. "Topical Locality in the Web", in Proceedings of the 23rd Annual International Conference on Research and Development in Information Retrieval (SIGIR 2000), July 2000, ACM

[DS04] Danny, S, Search Engine Watch, July 2004, available at http://searchenginewatch.com/searchday/article.php/3376041

[ECT] Excite Search Engine, available at http://www.excite.com/

[FEIS] "Find and Evaluate Internet Sources", University of Houston Victoria, available at http://www.uhv.edu/ac/research/prewrite/findinternet.pdf

[GAPI] Google Web APIs, available at http://www.google.com/apis/api_faq.html#gen1

[GB] Google Blog, available at http://www.google.com/googleblog/

[GH00] Gisle, H. "Search Engine Survey An Overview of the Mapmakers of Cyberspace", July 2000, available at http://heim.ifi.uio.no/~gisle/overload/engines.html

[GP01] Gary D. Price, "Specialized Search Engine FAQs: More Questions, Answers and Issues", available at http://www.infotoday.com/searcher/oct02/price.htm

[GP04] Gautam, P., Padmini, S., Filippo, M. "Crawling the Web", 2004, available at http://dollar.biz.uiowa.edu/~pant/Papers/crawling.pdf

[HA99] Heydon, A., Najork, M. "Mercator: A Scalable, Extensible Web Crawler", World Wide Web, Dec 1999.

[IFSK] InfoSeek search engine, available at http://www.infoseek.com

[INSP] Inspec, The Database for Physics, Electronics and Computing, available at http://www.iee.org/publish/inspec/

[JC98] Cho, J., Gaucia-Monlina, H., and Page, L. "Efficient Crawling through URL Ordering", in Proceedings of the 7th International World Wide Web Conference, Brisbane, Australia, Apr 1998

[KJ98] Kleinburg, J. "Authoritative Sources in a Hyperlinked Environment", Proceedings of the ACM-SIAM Symposium of Discrete Algorithms, 1998.

[KM94] Koster, M. "A Standard for Robot Exclusion", available at http://nersp.nerdc.ufl.edu/~nemnm/infoseek/norobots.html

[LC] The Library of Congress, available at http://www.loc.gov/about/

[LCS] Lycos Search Engine, available at http://www.lycos.com

[LV03] Lyman, P. and Varian, H. R. "How much information" available at http://www.sims.berkeley.edu/how-much-info-2003/

[MA99] McCallum, A., Nigam, K., Rennie, J., and Seymore, K. "A Machine Learning Approach to Building Domain Specific Search Engines," in Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-99), 1999, pp. 662-667.

[MC03] Michael, C. "Spidering and Filtering Web Pages for Vertical Search Engines", 2003, available at http://www.business.hku.hk/~mchau/papers/SpideringAndFiltering.pdf

[MCHC03] Michael, C and Hsinchun, C. "Personalized and Focused Web Spiders", 2003, available at http://citeseer.ist.psu.edu/548327.html

[MH98] Michael, H., Michal, J. et al. "The Shark-Search Algorithm – An Application: Tailored Web Site Mapping." In Proceedings of the 7th International World Wide Web Conference, 1998.

[MK95] Martin, K. April 1995 "Robots in the Web: threat or treat?" available at http://www.robotstxt.org/wc/threat-or-treat.html

[MN01] Marc, N and Janet, L. W. "Breadth-First Search Crawling Yields High-Quality Pages." In Proceedings of the 10th International World Wide Web Conference, 2001.

[MSE] "Multiple Search Engines", available at http://www.searchengineshowdown.com/multi/

[MYSQL] MySQL Home Page, available at http://www.mysql.com

[NA02] Niran, A. and Arnon, R. "Learnable Crawling: An Efficient Approach to Topic-specific Web Resource Discovery", 2002, Available at http://citeseer.ist.psu.edu/angkawattanawit02learnable.html

[NM01] Najork, M., Heydon, A. "High Performance Web Crawling", Sep 2001, available at http://gatekeeper.research.compaq.com/pub/DEC/SRC/research-reports/abstracts/src-rr-173.html

[NSDL] National Science Digital Library, available at http://www.nsdl.org

[NL00] Neel, S., Jeonghee, L., Anital, H. "Using MetaData to Enhance a Web Information Gathering System", 2000, available at http://www.research.att.com/conf/webdb2000/PAPERS/1b.ps

[NL02] Ned, L. F., Lucy, K. "Search Engines HandBook", published by Jefferson, NC : Farland & Co., 2002

[OV] Overture's Home Page, available at http://www.content.overture.com/d/

[PR99] Page Rank "The PageRank Citation Ranking: Bringing Order to the Web", available at http://dbpubs.stanford.edu:8090/pub/1999-66

[RMC] Robert M. Colomb. "A Digital Library Needs Many Indexes", available at http://www.itee.uq.edu.au/~colomb/Papers/Phronesis.html

[RR92] R. Rivest. "RFC 1321 – The MD5 Message-Digest Algorithm", April 1992, available at http://www.faqs.org/rfcs/rfc1321.html

[SANDBOX] Sandbox MSN, available at http://sandbox.msn.com/

[SB99] Soumen, C., Martin, B., Bryon, D. "Focused crawling: a new approach to topic-specific Web resource discovery", in 8th International WWW Conference May 1999, pp. 545-562.

[SC99] Steve, L., C. Lee, G., Kurt, B. "Digital Libraries and Autonomous Citation Indexing", in IEEE Computer, Volume 32, November 6, pp. 67-71, 1999.

[SDL] Stanford Digital Library, available at http://www-diglib.stanford.edu/

[SHBC] S. Haddad, Bipin C. Desai. "ASHG: Automatic Semantic Header Generator", available at http://www.cs.concordia.ca/~bcdesai/grads/haddad-thesis.pdf

[SK99] Steve, L., Kurt, B., C. Lee, G. "Indexing and Retrieval of Scientific Literature", Eighth International Conference on Information and Knowledge Management, November 2-6, pp. 139-146, 1999.

[SLDL] Scientific Literature Digital Library, available at http://citeseer.ist.psu.edu

[SM99] Soumen, C., Martin, B., Bryon, D. "Distributed Hypertext Resource Discovery Through Examples", 1999, available at http://www.fxpal.com/people/vdberg/pubs/paper_vldb99_P37.pdf

[SSAV] Search Site Alta Vista, available at http://livinginternet.com/w/wu_sites_alta.htm

[TOP] Top 100 Sites with Details of HTTP Server and Operating System, available at http://homepages.tig.com.au/~jmsalvo/top100/top100sites.html

[TZ04] Tong, Z. "A Gleaning Subsystem for CINDI", Master Thesis, Dept. of Computer Science, Concordia University, 2004.

[WRD] The Web Robot Database, available at http://www.robotstxt.org/wc/active.html

[WT98] Wes, S. and Tim, M. "Guide to Search Engines", published by Wiley Computer Publishing, 1998, pp. 1.

[VG97] Venkat, G., Vijav, R., William, G., Rajesh, K. "Information Retrieval on the World Wide Web", 1997 available at http://www.cacs.louisiana.edu/~raghavan/internet97.pdf

[XUE03] Xue, F.R. "Enhancement of the CINDI System", Master Thesis, Dept. of Computer Science, Concordia University, 2003.

[ZZ02] Zhan Z. "Porting the Automatic Semantic Header Generator to the Web", Major Report, Dept. of Computer Science, Concordia University, 2002.

Appendix A

Search Results of AltaVista, MSN and Google Using Phrase: computer science department (Accessed on July 25, 2005)

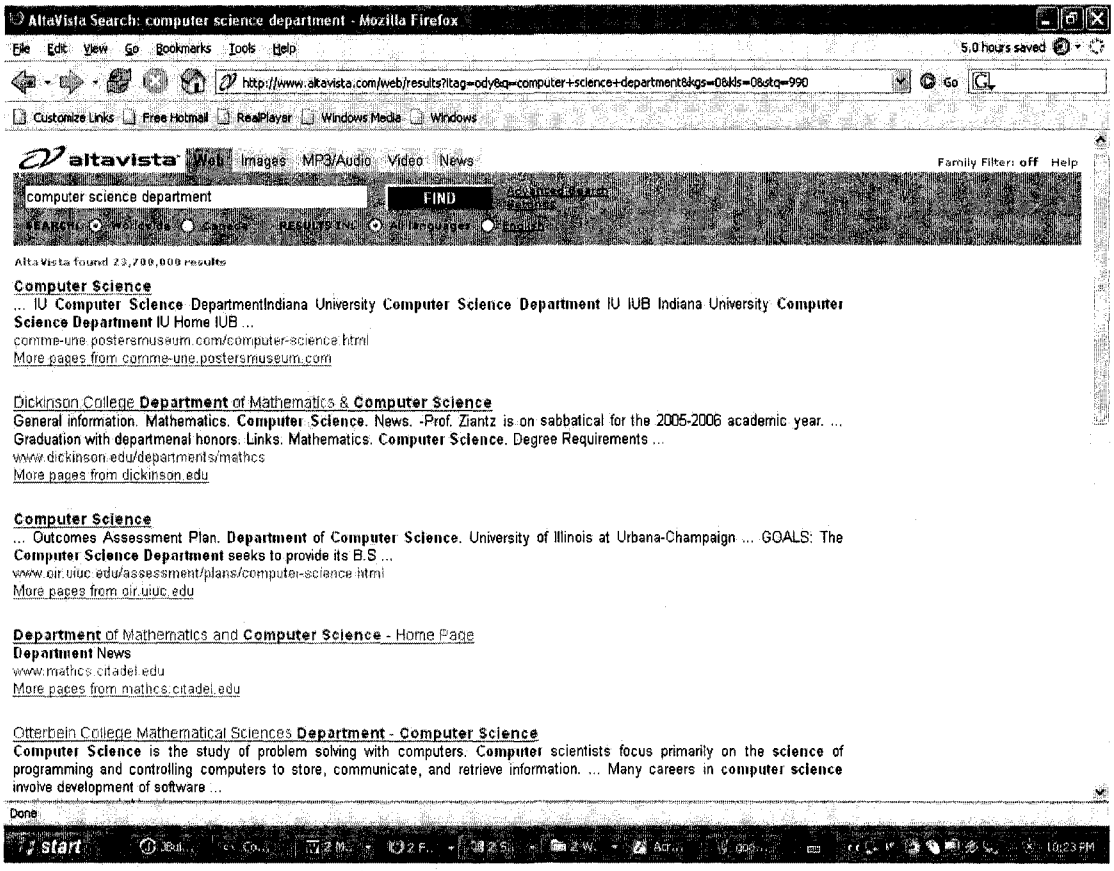

Fig A: AltaVista Web Page (Results 991 - 1000)

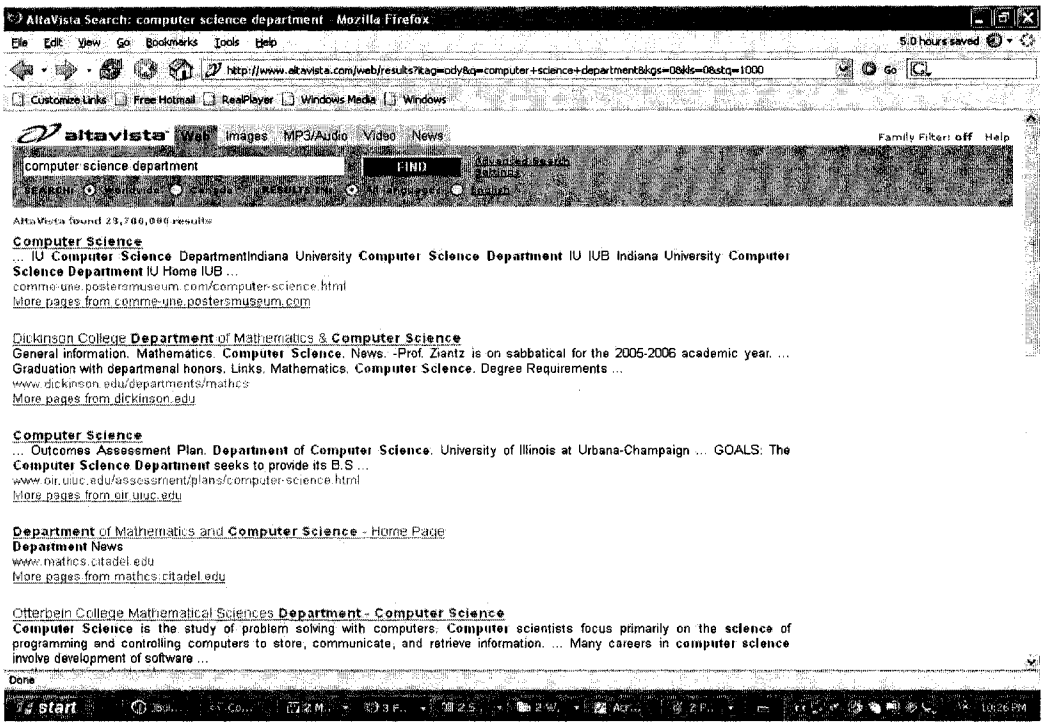

Fig B: AltaVista Web Page (Results > 1000)

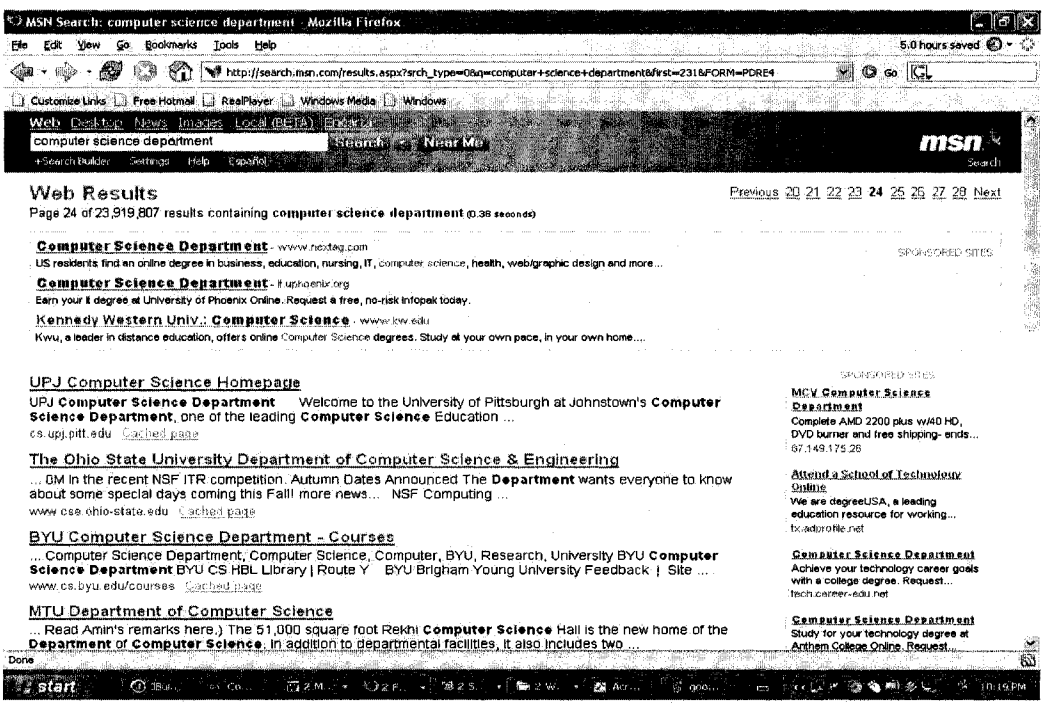

Fig C: MSN Web Page (Results 231 – 240)

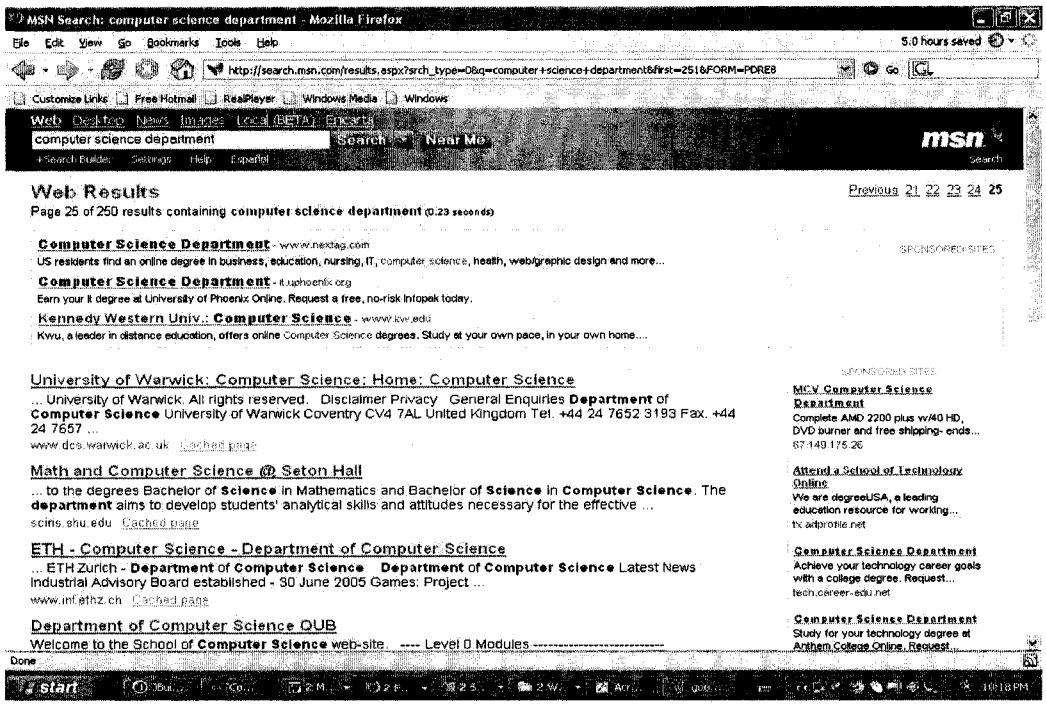

Fig D: MSN Web Page (Results 241 – 250)

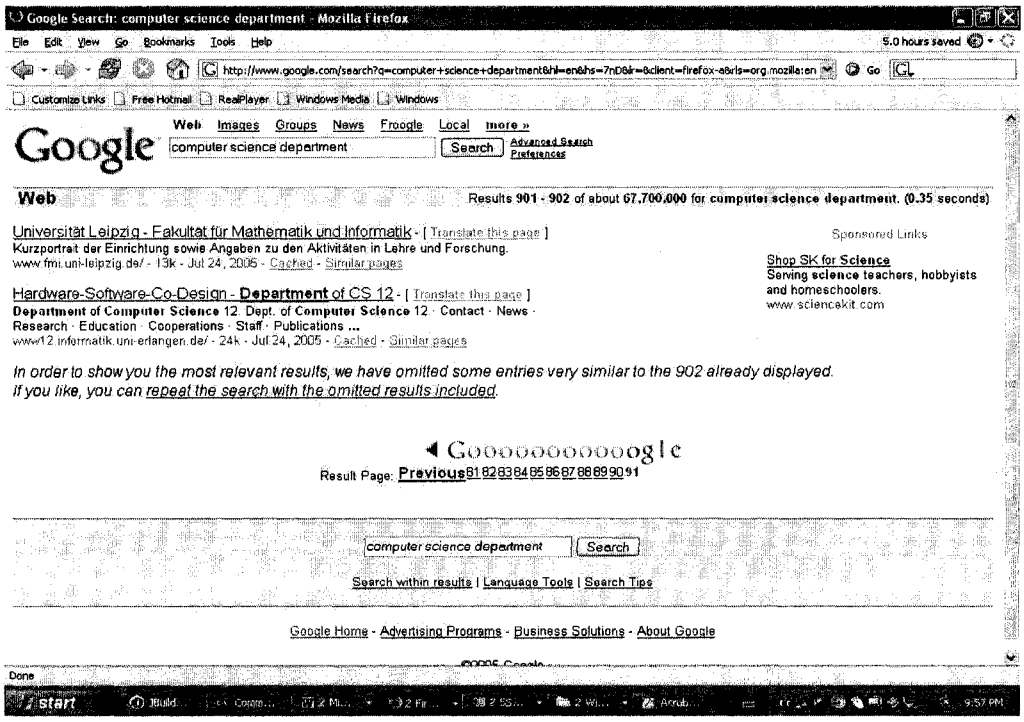

Fig E: Google Web Page (Results 901 – 902)

Search Results of AltaVista, MSN and Google Using Phrase: computer science publications (Accessed on July 25, 2005)

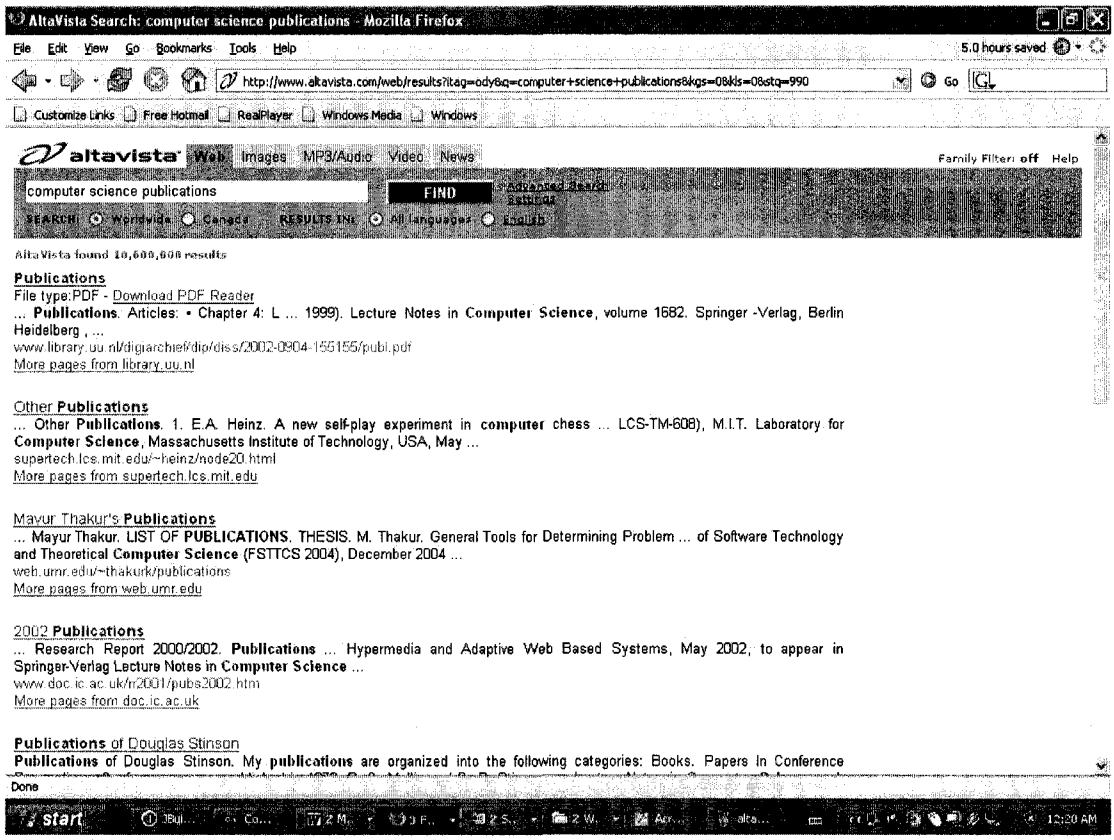

Fig F: AltaVista Web Page (Results 991 - 1000)

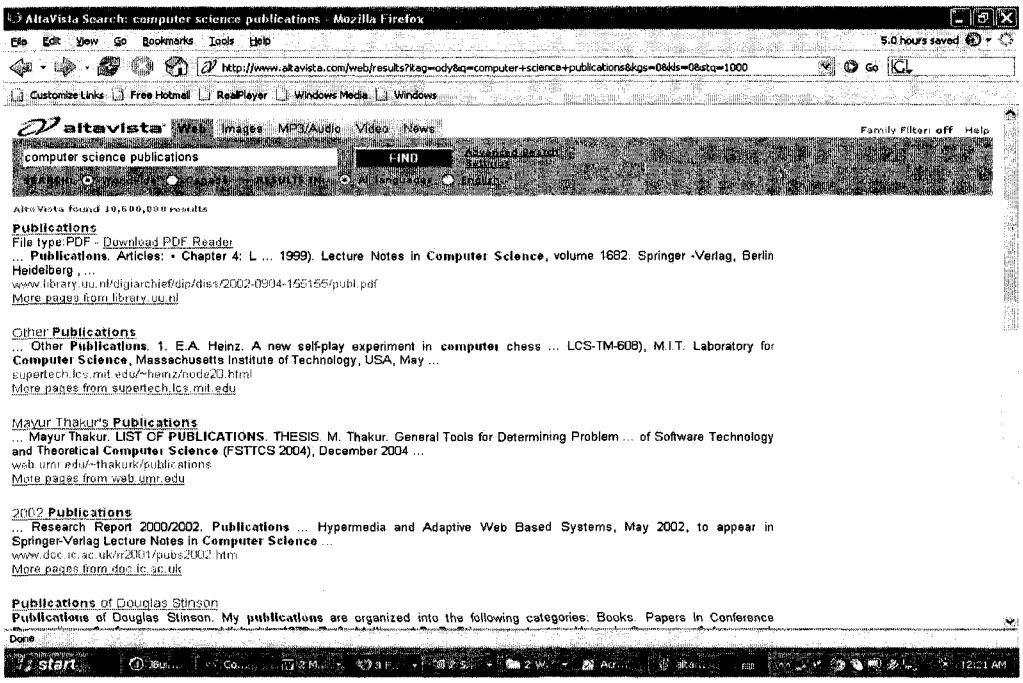

Fig G: AltaVista Web Page (Results > 1000)

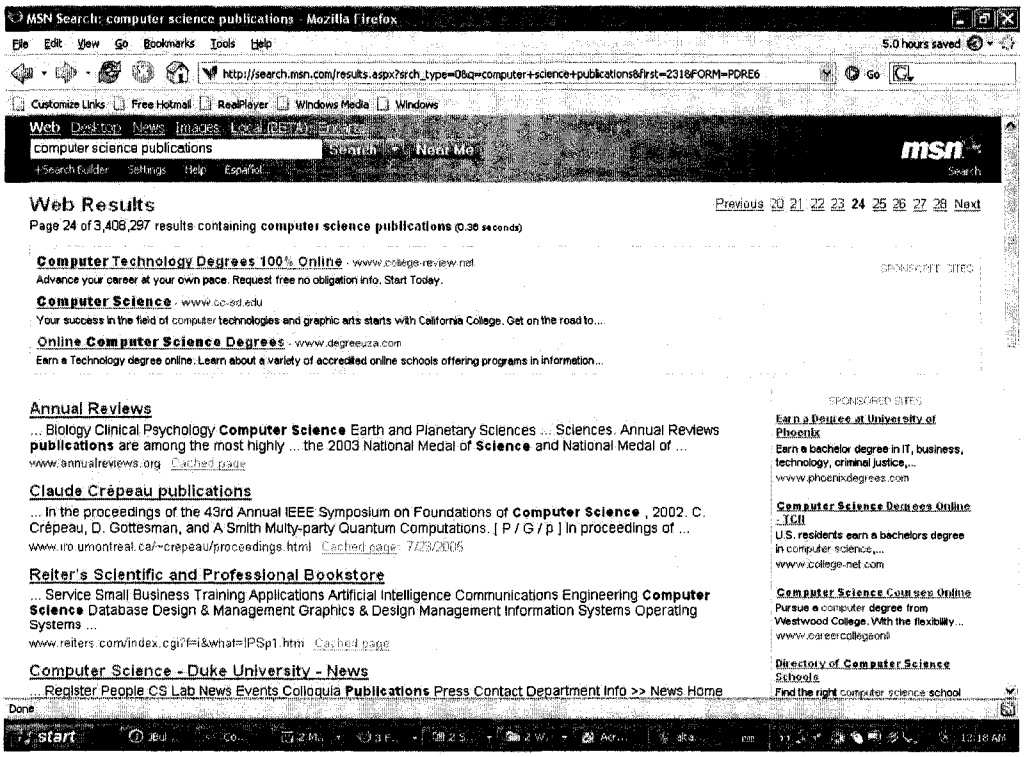

Fig H: MSN Web Page (Results 231 – 240)

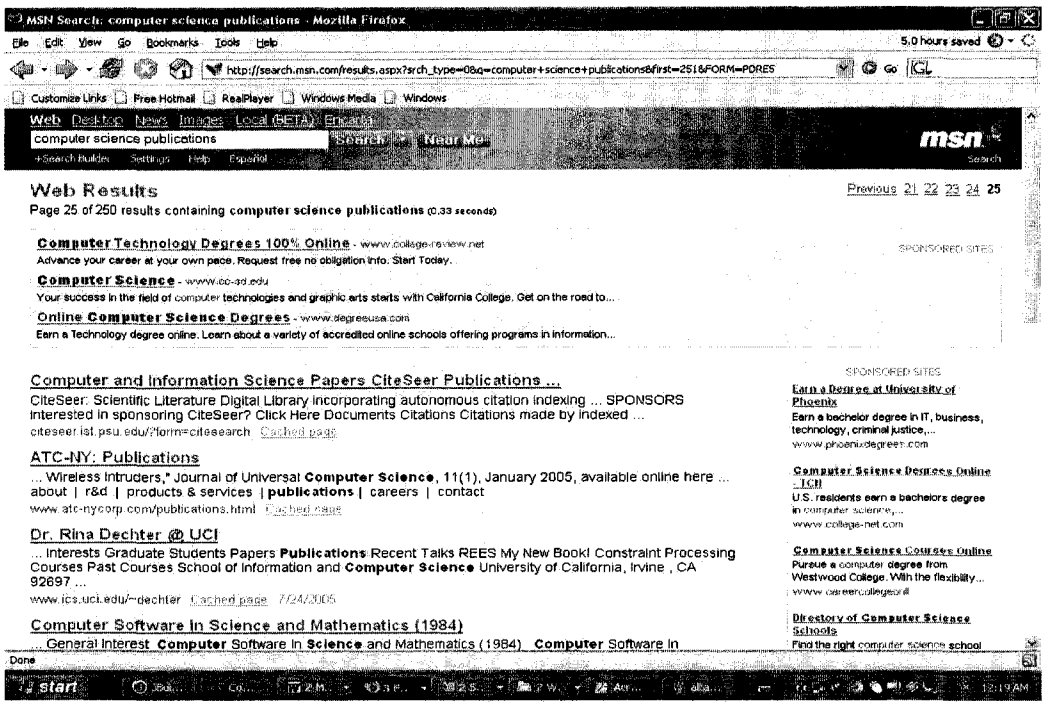

Fig I: MSN Web Page (Results 241 – 250)

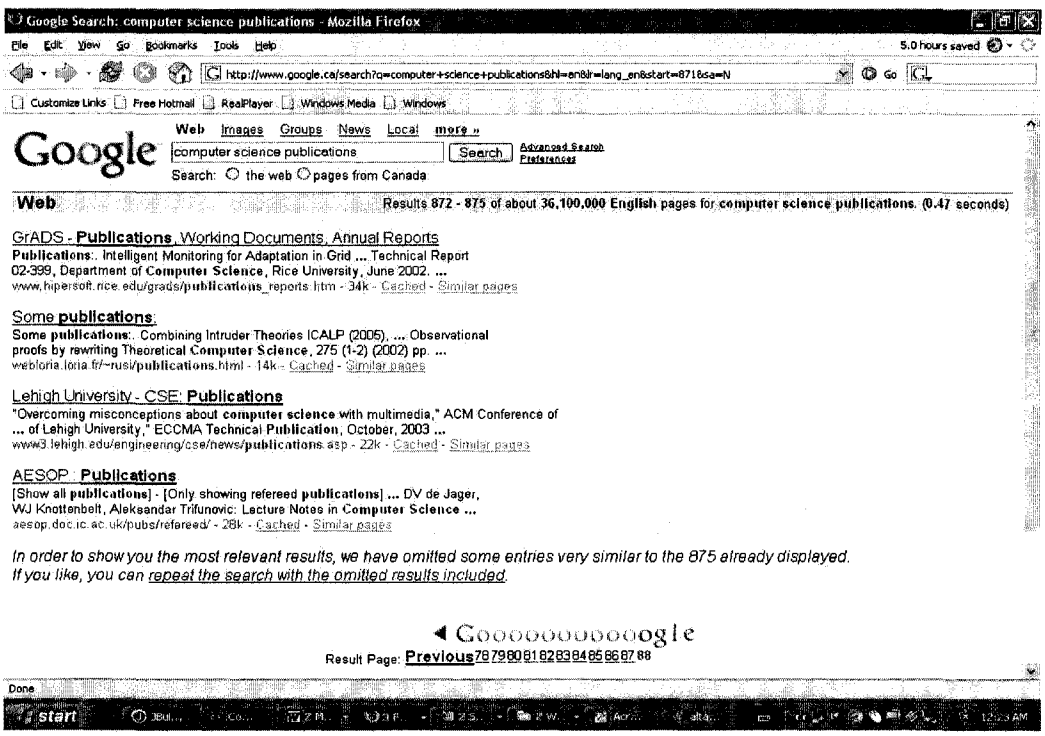

Fig J: Google Web Page (Results 872 – 875)

Appendix B

Source : IBM Official Web Site: http://www.ibm.com/robots.txt

```
# $Id: robots.txt,v 1.19 2004/11/21 16:33:07 krusch Exp $
#
# This is a file retrieved by webwalkers a.k.a. spiders that
# conform to a defacto standard.
# See <URL:http://www.robotstxt.org/wc/exclusion.html#robotstxt>
#
# Comments to the webmaster should be posted at
  <URL:http://www.ibm.com/contact>
#
# Format is:
#       User-agent: <name of spider>
#       Disallow: <nothing> | <path>
# Flag  Date      By      Reason
# $11-  19950130  epc     finally understood what the file was for!
# $L2=  19960909  epc     fixed url since mak moved to Webcrawler...
# $L3=  19970811  epc     drop /Stretch
# $L4=  19991102  krusch  fixed User-agent capitalization and contact
info
# $L5=  20010327  krusch  Updated disallow rules
# ---------------------------------------------------------------------
User-agent: *
Disallow: //
Disallow: /Admin
Disallow: /admin
Disallow: /zx
Disallow: /zz
Disallow: /common
Disallow: /cgi-bin
Disallow: /scripts
Disallow: /Scripts
Disallow: /i/
Disallow: /image
Disallow: /Search
Disallow: /search
Disallow: /link
Disallow: /perl
Disallow: /tmp
Disallow: /account/registration
Disallow: /webmaster
Disallow: /products/finder
Disallow: /products/learn/action

User-agent: Fast corporate crawler
Disallow: //
Disallow: /Admin
Disallow: /admin
Disallow: /zx
Disallow: /zz
Disallow: /common
Disallow: /cgi-bin
Disallow: /scripts
```

```
Disallow: /Scripts
Disallow: /i/
Disallow: /image
Disallow: /Search
Disallow: /search
Disallow: /link
Disallow: /perl
Disallow: /tmp
Disallow: /webmaster
Disallow: /products/finder
Disallow: /products/learn/action
#
```

Appendix C

DOMAIN KEYWORDS

ID	keyword
1	general
2	control design style
3	control structure performance analysis
4	control structure reliability and testing
5	microprogram design aids
6	microcode applications
7	miscellaneous
8	design styles
9	performance analysis and design aids
10	reliability, testing, and fault-tolerance
11	high-speed arithmetic
12	semiconductor memories
13	data communications devices
14	input/output devices
15	interconnections (subsystems)
16	design
17	design aids

18	reliability and testing
19	types and design styles
20	single data stream architectures
21	multiple data stream architectures
22	other architecture styles
23	parallel architectures
24	network architecture and design
25	network protocols
26	network operations
27	distributed systems
28	local and wide-area networks
29	internetworking
30	large and medium (mainframe) computers
31	minicomputers
32	microcomputers
33	vlsi systems
34	servers
35	applicative (functional) programming
36	automatic programming
37	concurrent programming
38	sequential programming
39	object-oriented programming
40	logic programming
41	visual programming
42	requirements/specifications
43	design tools and techniques
44	coding tools and techniques
45	software/program verification
46	testing and debugging
47	programming environments
48	distribution, maintenance, and enhancement
49	metrics
50	management
51	software architectures

52	interoperability
53	reusable software
54	formal definitions and theory
55	language classifications
56	language constructs and features
57	processors
58	process management
59	storage management
60	file systems management
61	communications management
62	reliability
63	security and protection
64	organization and design
65	performance
66	systems programs and utilities
67	models of computation
68	modes of computation
69	complexity measures and classes
70	numerical algorithms and problems
71	nonnumerical algorithms and problems
72	tradeoffs between complexity measures
73	specifying and verifying and reasoning
74	semantics of programming languages
75	studies of program constructs
76	mathematical logic
77	grammars and other rewriting systems
78	formal languages
79	interpolation
80	approximation
81	numerical linear algebra
82	quadrature and numerical differentiation
83	roots of nonlinear equations
84	optimization
85	ordinary differential equations

86	partial differential equations
87	integral equations
88	applications
89	combinatorics
90	graph theory
91	systems and information theory
92	user/machine systems
93	logical design
94	physical design
95	languages
96	systems
97	heterogeneous databases
98	database machines
99	database administration
100	database applications
101	content analysis and indexing
102	information storage
103	information search and retrieval
104	systems and software
105	online information services
106	library automation
107	digital libraries
108	office automation
109	types of systems
110	communications applications
111	multimedia information systems
112	user interfaces
113	group and organization interfaces
114	hypertext/hypermedia
115	sound and music computing
116	expressions and their representation
117	algorithms
118	languages and systems
119	applications and expert systems

120	deduction and theorem proving
121	knowledge representation formalisms and method
122	programming languages and software
123	learning
124	natural language processing
125	problem solving, control methods, and search
126	robotics
127	vision and scene understanding
128	hardware architecture
129	graphics systems
130	picture/image generation
131	graphics utilities
132	computational geometry and object modeling
133	methodology and techniques
134	three-dimensional graphics and realism
135	digitization and image capture
136	compression
137	enhancement
138	restoration
139	feature measurement
140	scene analysis
141	image representation
142	models
143	design methodology
144	clustering
145	simulation theory
146	simulation languages
147	model validation and analysis
148	model development
149	simulation output analysis
150	simulation support systems
151	types of simulation
152	document and text editing
153	document preparation

154	index generation
155	electronic publishing
156	computer uses in education
157	computer and information science education
158	public policy issues
159	social issues
160	organizational impacts
161	electronic commerce
162	hardware/software protection
163	governmental issues
164	project and people management
165	installation management
166	software management
167	system management
168	occupations
169	organizations
170	testing, certification, and licensing
171	professional ethics
172	application packages
173	hardware
174	management/maintenance

Appendix D

AltaVista Result Page HTML Source – Abridged

```
<html><head><title>AltaVista Search: computer science
department</title>
<meta name="description" content="AltaVista provides the most
comprehensive search experience on the Web!">
<meta name="keywords" content="search, searches, search engine,
directory, directories, category, categories, help, multi media, maps,
business finder, yellow pages, white pages,  people search, find
people, searching, searchers, advanced search, search help, search
guide, search tips, search tools">
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
 <LINK REL="SHORTCUT ICON" HREF="/favicon.ico">
<base target="_top">
<STYLE TYPE="text/css"><!--
-->
</STYLE>
 <script language=javascript>
<!--
//-->
</script></head>

</div><div id="results"><br class=lb>
<DIV class=xs style="PADDING-BOTTOM: 6px"><B><A class=lbl href="http:
//www.altavista.com/help/search/types_web%231" target=_blank
onMouseOver="this.style.color='#2249cc'"
onMouseOut="this.style.color='#999999'">Sponsored Matches</A></B> 
<a
href="http://www.content.overture.com/d/USm/ays/bjump/%3fo=U5213%26b=5%
26c=SearchEngineWatch" class=ltgy>Become a sponsor</a>
</DIV>                    <a class="res"
href="http://it.uphoenix.org"><b>Computer</b> <b>Science</b>
<b>Department</b></a><br>
<span class=s>Earn your It degree at University of Phoenix Online.
Request a free, no-risk Infopak today.</span><br>
<span class=ngrn>it.uphoenix.org</span><br>                    <br
class="smbr">                        <a class="res"
href="http://www.kw.edu">Kennedy Western Univ.: <b>Computer</b>
<b>Science</b></a><br>
                    <span class=s>Kwu, a leader in distance education,
offers online <b>Computer</b> <b>Science</b> degrees. Study at your own
pace, in your own home. Register for a free catalog. Must be 23 with 5+
years experience.</span><br>
                    <span class=ngrn>www.kw.edu</span><br>
<br class="smbr">                        <a class="res"
href="http://tech.career-edu.net"><b>Computer</b> <b>Science</b>
<b>Department</b></a><br>
                    <span class=s>Achieve your technology career goals
with a college degree. Request free school information and start
training for a better future.</span><br>
```

```
                    <span class=ngrn>tech.career-edu.net</span><br>
<br class="smbr">                    <a class="res"
href="http://www.nextag.com"><b>Computer</b> <b>Science</b>
<b>Department</b></a><br>
                    <span class=s>Find an online degree in business,
education, nursing, IT, <b>computer</b> <b>science</b>, health,
web/graphic design, criminal justice and more from a directory of
accredited universities.</span><br>
                    <span class=ngrn>www.nextag.com</span><br>
<br class="smbr"><br class=lb>
<DIV class=xs><B><A class=lbl
href="http://www.altavista.com/help/search/types_web%231"
onMouseOver="this.style.color='#2249cc'"
onMouseOut="this.style.color='#999999'">AltaVista found 27,600,000
results</A></B> 
</DIV>

<br class='lb'><a class='res' href='http://www.cs.indiana.edu/'>IU
<b>Computer</b> <b>Science</b> <b>Department</b></a><br><span class=s>
... Welcome to the Indiana University <b>Computer</b> <b>Science</b>
<b>Department</b> website. We are proud to present these ... Award for
faculty in the <b>Computer</b> <b>Science</b> <b>department</b> is
Professor Kent Dybvig ... <br></span>
<span class=ngrn>www.cs.indiana.edu</span> <span
class=rgy></span><br><a class=rgy href='http://cs.indiana.edu'>More
pages from cs.indiana.edu</a> <br><br><br class='lb'><a
class='res' href='http://www-cs.stanford.edu/'>Stanford <b>Computer</b>
<b>Science</b> <b>Department</b></a><br><span class=s> ... Founded in
1965, the <b>Department</b> of <b>Computer</b> <b>Science</b> is a
center for research ... of artificial intelligence, robotics,
foundations of <b>computer</b> <b>science</b>, scientific computing,
and systems ... <br></span><span class=ngrn>www-cs.stanford.edu </span>
<span class=rgy></span><br><a class=rgy href='http://www-
cs.stanford.edu'>More pages from www-
cs.stanford.edu</a> <br><br><br class='lb'><a class='res'
href='http://www.cs.umass.edu/'>UMass Amherst: <b>Department</b> of
<b>Computer</b> <b>Science</b></a><br><span class=s>Welcome to internet
home of the <b>Department</b> of <b>Computer</b> <b>Science</b> at the
University of Massachusetts Amherst. ... This site is maintained by the
<b>Department</b> of <b>Computer</b> <b>Science</b>. The Department's
<b>Computer</b> <b>Science</b> Research Center on the UMass Amherst
campus is home to over 30 research laboratories ... <br></span><span
class=ngrn>www.cs.umass.edu </span> <span class=rgy></span><br><a
class=rgy href='http://cs.umass.edu'>More pages from
cs.umass.edu</a> <br><br><br class='lb'><a class='res'
href='http://www.cs.man.ac.uk/'>School of <b>Computer</b>
<b>Science</b></a><br><span class=s>The University of Manchester,
School of <b>Computer</b> <b>Science</b> ... Study <b>Computer</b>
<b>Science</b>. News. Events. People. Contact <b>Computer</b>
<b>Science</b>. Search <b>Computer</b> <b>Science</b>. What's new ...
The first <b>Computer</b> <b>Science</b> <b>department</b> in the UK
and an internationally ... <br></span><span class=ngrn>www.cs.man.ac.uk
</span> <span class=rgy></span><br><a class=rgy
href='http://cs.man.ac.uk'>More pages from
cs.man.ac.uk</a> <br><br><br class='lb'><a class='res'
href='http://www.cs.cornell.edu/'><b>Department</b> of <b>Computer</b>
<b>Science</b>, Cornell University</a><br><span class=s> ...
```

Computer Science. The Department of
Computer Science offers undergraduate degrees in Arts and
Sciences and Engineering ... Strategic Plan. Women in Computer
Science. Ugrad Program. ACSU ...
<span
class=ngrn>www.cs.cornell.edu
<a
class=rgy href='http://cs.cornell.edu'>More pages from
cs.cornell.edu

<br class='lb'><a class='res'
href='http://www.cs.sunysb.edu/'>Computer Science
Department
 ... Welcome to the
Computer Science Department at Stony Brook
University. Established in 1969, the Computer Science
Department at Stony Brook University is ranked consistently
among the ...
www.cs.sunysb.edu

<a class=rgy
href='http://cs.sunysb.edu'>More pages from
cs.sunysb.edu

<br class='lb'><a class='res'
href='http://www.cs.tcd.ie/'>Computer Science - Trinity
College Dublin
Trinity College Dublin:
Computer Science Department ... Department
of Computer Science. Home ... Local. Department of
Computer Science ...
<span
class=ngrn>www.cs.tcd.ie
<a
class=rgy href='http://cs.tcd.ie'>More pages from
cs.tcd.ie

<br class='lb'><a class='res'
href='http://www.cs.ucsb.edu/'>Department of Computer
Science
 ... UNIVERSITY OF CALIFORNIA SANTA
BARBARA COMPUTER SCIENCE. PROGRAMS. COURSES. PEOPLE ...
February 2005. ECE Department is Inviting Faculty Applicants in
Computer Engineering - August 2004 ...
<span
class=ngrn>www.cs.ucsb.edu
<a
class=rgy href='http://cs.ucsb.edu'>More pages from
cs.ucsb.edu

<br class='lb'><a class='res'
href='http://www.csd.uwo.ca/'>UWO - Computer Science
Department - Welcome
 ... to the University
of Western Ontario, Computer Science department.
Western offers many options to obtain a degree in Computer
Science or in combination with another ...
<span
class=ngrn>www.csd.uwo.ca
<a
class=rgy href='http://csd.uwo.ca'>More pages from
csd.uwo.ca

<br class='lb'><a class='res'
href='http://www.cs.uiuc.edu/'>Department of Computer
Science | University of Illinois at Urbana-
Champaign
CS Changes Curriculum to Meet Future IT
Challenges. ... In the coming year, the Computer Science
Department will institute significant curriculum changes to ...

www.cs.uiuc.edu <span
class=rgy>
More pages
from cs.uiuc.edu

<DIV class=xs style="PADDING-BOTTOM:
6px"><B><A class=lbl href="/web/lsa" target=_blank
onMouseOver="this.style.color='#2249cc'"
onMouseOut="this.style.color='#999999'">**Sponsored Matches**</A></B>
<a
href="http://www.content.overture.com/d/USm/ays/bjump/%3fo=U5213%26b=5%
26c=SearchEngineWatch" class=ltgy>Become a sponsor
</DIV>
Attend a School of
Technology Online

```
<span class=s>We are degreeUSA, a leading education resource for
working professionals who want to advance their careers. Find the
college programs you want quickly and easily online.</span><br>
<span class=ngrn>tx.adprofile.net</span><br><br class="smbr">
<a class="res" href="http://www.college-review.net"><b>Computer</b>
Technology Degrees 100% Online</a><br>
<span class=s>Advance your career at your own pace. Request free no
obligation info. Start Today.</span><br>
<span class=ngrn>www.college-review.net</span><br><br class="smbr">
<a class="res" href="http://kwu.college-info.org"><b>Computer</b>
<b>Science</b> <b>Department</b></a><br>
<span class=s>Earn your technology degree at Kennedy-Western
University. Request free program information and start training
today.</span><br>
<span class=ngrn>kwu.college-info.org</span><br><br
class="smbr"></div><table border=0 cellspacing=0 cellpadding=2
width="100%">
<tr><td colspan=3><hr size=1 noshade color="#93B2DD"
align=center></td></tr><tr><td class=s nowrap valign=top>

<img
src="http://bsav.search.yahoo.com/serv?s=396566003&t=1120589590&_ylb=A9
ibyK8W18pCmoQAvAPzHaMX" width="1" height="1" border="0">
</body></html>
<!-- a20.search.dcn.yahoo.com compressed/chunked Tue Jul  5 11:53:10
PDT 2005 -->
```