Discovering Dependencies in Courseware Repositories

M. Tech. Dissertation

Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology

by

Nidhi Malik

Roll No: 06305002

under the guidance of

Prof. Sridhar Iyer

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai

Acknowledgements

I would like to thank Prof.Sridhar Iyer for his keen guidance, insights and constant support. It has been a good learning experience working with him.I thank him for always being there to discuss problems, solutions and new ideas.

I deeply thank my family for their constant encouragement and support.

I want to thank Chintan for solving my JAVA related queries and discussing critical points. I also thank Ramdasji for his valuable ideas. I also thank Janak, Pratabh, Manish for giving me inputs to evaluate the performance of the system.

Last but not the least, I want to express my heartfelt gratitude to my friends, colleagues and staff at CSE, IITB for making my stay here wonderful.

Nidhi Malik
Department of Computer Science and Engineering,
IIT Bombay

Abstract

Nowadays elearning has become popular, especially with the availability of large courseware repositories such as NPTEL and OCW. A variety of e-learning tools and systems are also available. However, suppose a user wants to learn about a particular subject, the search tools typically just return a large number of links to the user in response to his/her query. Many of these are not directly relevant, so the user does not know which links to follow in order to enhance his knowledge.

In this project we have built a system which not only provides the user with the most relevant learning module for his query, but also provides him/her with the relevant pre-requisite and follow-up modules also. This is done by creating a dependency graph of the courseware modules in the repository. We have implemented and tested our system on six courses taken from the NPTEL repository. For effective evaluation of our solution approach, we have not only compared the results for four different heuristics but also compared them with the dependencies determined by an expert in the subject area.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

eLearning is becoming quite important due to the emerging technologies. It is used as a means of learning by all categories of people from science, engineers,working professionals etc. A number of benefits can be obtained through the use of eLearning:

- Increases conceptual understanding through the use of interactivity and animation, and through the use of audio and video.
- Learners can learn at ones own pace and time. This is especially important for learners who cannot attend school physically during the regular hours.
- The system may retain records of discussion and allows for later reference through the use of threaded discussion on bulletin boards.
- eLearning permits instructors to develop materials using the world-wide resources of the Web.

For more details about elearning and its importance see [Sie04].

1.1 Availability of Repositories

There are huge amounts of data available on the World Wide Web (WWW). Different types of tools are available which help users in different ways from simply viewing the available content to creating lessons with the help of authoring tools. Many institutes/Universities have also made their courses open source and have published them on the web. However, the content is scattered over the WWW in the form of wikis, e-books, tutorials [GC05] etc. Some examples of content repositories are:
MIT OpenCourseWare [MIT] is an idea developed by MIT to advance knowledge and educate students in science, technology, and other areas also. There are over 1,800courses in 33 academic disciplines. This content is available for download freely in the form of MIT's Open Course Ware and there is a dedicated website for this dissemination [MIT].

The **National Programme on Technology Enhanced Learning (NPTEL)** [NPT], a project funded by the Ministry of Human Resource Development (MHRD), India, was initiated in 1999 to pave the way for introducing multimedia and web technology to enhance learning of basic science and engineering concepts. In order to facilitate the distribution of course material, two modes of operation have been suggested, namely, digital video lectures of courses and web based courses.

There are other repositories also including **CDEEP IIT Bombay**[CDE], **Stanford University's open education initiative**[Sta] and **utahs university e-learning program**[Uta] etc.

1.2 Motivation for MTP

Because of emerging technologies today, students have more options in choosing ways of learning[Kha05]. Many academic institutions/organizations have their own search facilities added to their content repository web sites [MIT], [NPT], [Sta]. However, suppose a user wants to learn about a particular subject, the search tools typically just return a large number of links to the user in response to his/her query. Many of these are not directly relevant, so the user does not know which links to follow in order to enhance his knowledge.

We looked upon the search facility provided by NPTEL, CDEEP, and MITs OCW. We acted as a user and gave queries to learn about topics such as TCP/IP, Ethernet etc. Although these topics are covered in courses available at these websites, the search facilities did not give the desired results. In this project we have built a system which not only provides the user with the most relevant learning module for his query, but also provides him/her with the relevant pre-requisite and follow-up modules also. There is no existing tool which actually serves this purpose.

1.3 Goal of MTP

Given a set of lecture files from a courseware repository, we aim to build a system that provides the user the most relevant answer (course module) to his query. We also create a dependency graph for the entire course so that the user can refer to the previous and advanced topics as required. We do not assume any prior knowledge of the ordering of the concepts covered in the modules.

We have divided our system into three parts: providing learning module to the user, feedback for the course co-ordinator and the proposed design of the overall system. We mainly focus on providing the user the most relevant learning module for the query entered. We also provide the prerequisites and follow ups for that topic. User can always refer to the prerequisites in case of any doubts and follow-ups to enhance his/her knowledge further. The user can judge himself by taking the self-evaluation quiz. The quiz will consist of multiple choice questions of that field.

1.4 Solutions Approach

Given the contents of a course from a repository (NPTEL in our case), we do the following:

1. We have used the Lucene, open source information retrieval library, in order to index the content available.

2. We use this index in order to not only retrieve the relevant documents but also to determine the dependencies among the modules.

3. Different strategies (heuristics) are used to order documents and then to arrange them in the order of pre-requisites and follow ups.

4. We have implemented and tested our system on several courses taken from NPTEL. For effective evaluation of our solution approach, we have not only compared the results for the various heuristics but also compared them with the dependencies as determined by an expert in the subject area.

5. Subsequently we have proposed a mechanism to present questions to the user in the form of a self-evaluation quiz. The response of the quiz may be used to provide feedback not only to the user but also to the content creator.

1.5 Organization of Thesis

In chapter 2, we look on different content repositories available emphasizing on NPTEL, CDEEP and MITs OCW in particular. In chapter 3 we describe the different solution approaches followed to implement our system. Chapter 4 gives the implementation details. In chapter 5 we show the different experiments conducted to evaluate the performance of the system. In chapter 6, we present the quiz and feedback module. In chapter 7, we present the conclusions and future extensions.

Chapter 2

Related Work

E-learning is used interchangeably in a wide variety of contexts. E-Learning can be used in academic settings as well as for corporate training [Kha05], in this report, we focus only on the use of eLearning in academic settings. The range of eLearning software spans the following extreme end points: using simple presentation tools such as Microsoft PowerPoint; the use of animation in tools such as JAWAA ; intelligent tutoring systems such as Andes etc.

2.1 Repositories surveyed

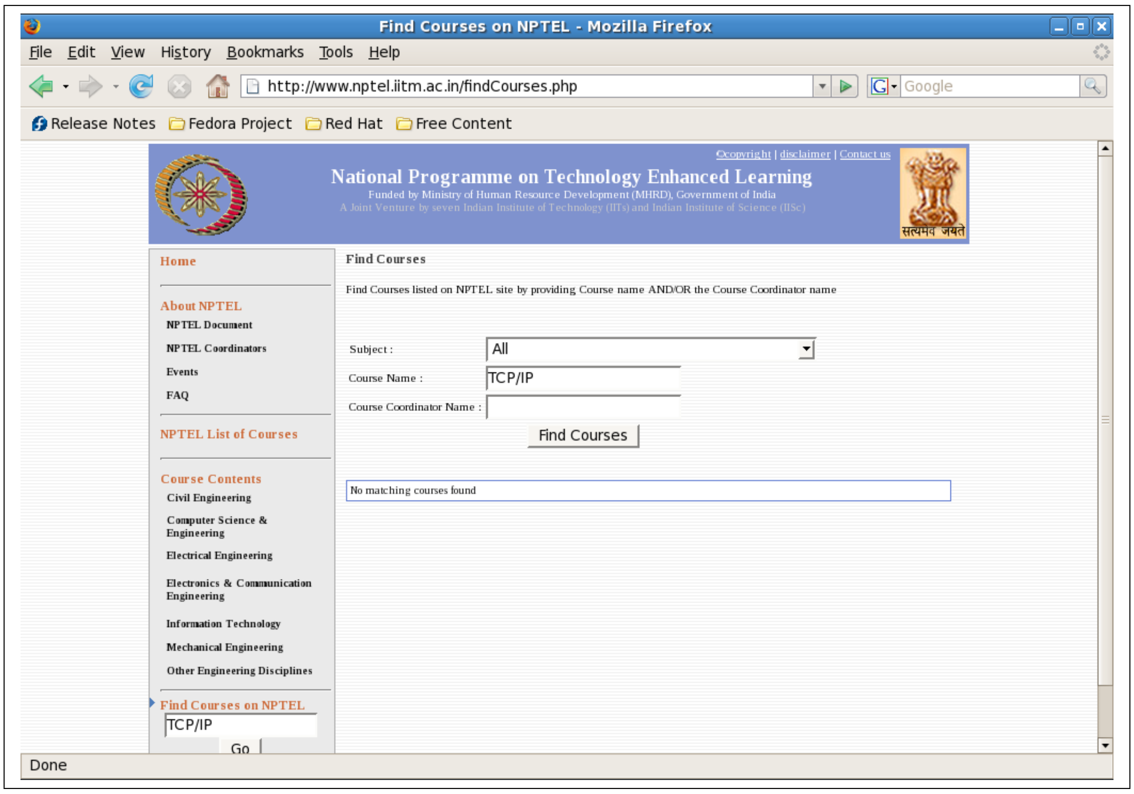
We have surveyed a number of content repositories available on different sites. These provide access to all of it's course's contents free of cost.

2.1.1 National Program on Technology Enhanced Learning (NPTEL)

The National Programme on Technology Enhanced Learning (NPTEL) [NPT], a project funded by the Ministry of Human Resource Development (MHRD) was initiated in 1999 to pave the way for introducing multimedia and web technology to enhance learning of basic science and engineering concepts. In order to facilitate the distribution of course material, two modes of operation have been suggested, namely, digital video lectures of courses and web based courses. E-Learning material are being created in such a form that it can be expanded and updated continuously. Simple course management packages that provide features like e-mail queries by students, bulletin board and Frequently Asked Questions (FAQ) are being incorporated.
There is no efficient search facility provided in NPTEL. The search option provided just works for the course names and one can't get any information about a particular query if it does not appear in the course name.
Sample screen shot for the query TCP/IP is shown below in the Figure 2.1

2.1.2 MIT's OCW

MIT OpenCourseWare is an idea developed by the MIT faculty to advance knowledge and educate students in science, technology, and other areas of scholarship to best serve the world. In 2000, MIT published the first proof- of-concept site in 2002, containing 50 courses. Now,



http://www.nptel.iitm.ac.in/findCourses.php

©copyright | disclaimer | Contact us

National Programme on Technology Enhanced Learning
Funded by Ministry of Human Resource Development (MHRD), Government of India
A Joint Venture by seven Indian Institute of Technology (IITs) and Indian Institute of Science (IISc)

Find Courses

Find Courses listed on NPTEL site by providing Course name AND/OR the Course Coordinator name

Subject : All
Course Name : TCP/IP
Course Coordinator Name :

No matching courses found

TCP/IP

Figure 2.1: Screen-shot of NPTEL

there are over 1,800 courses in 33 academic disciplines. This content is available for download freely in the form of MIT's Open Course Ware and there is a dedicated website for this.
Most of the content has been made available in the form of PDF documents. Some of these contain scans of material handwritten or drawn by the instructor and image captures of the blackboard contents.

The output returned 82 results for TCP/IP. This is really not helpful for a new user. It will be hard to decide out of the given 82 results which one he/she should follow and in which order.

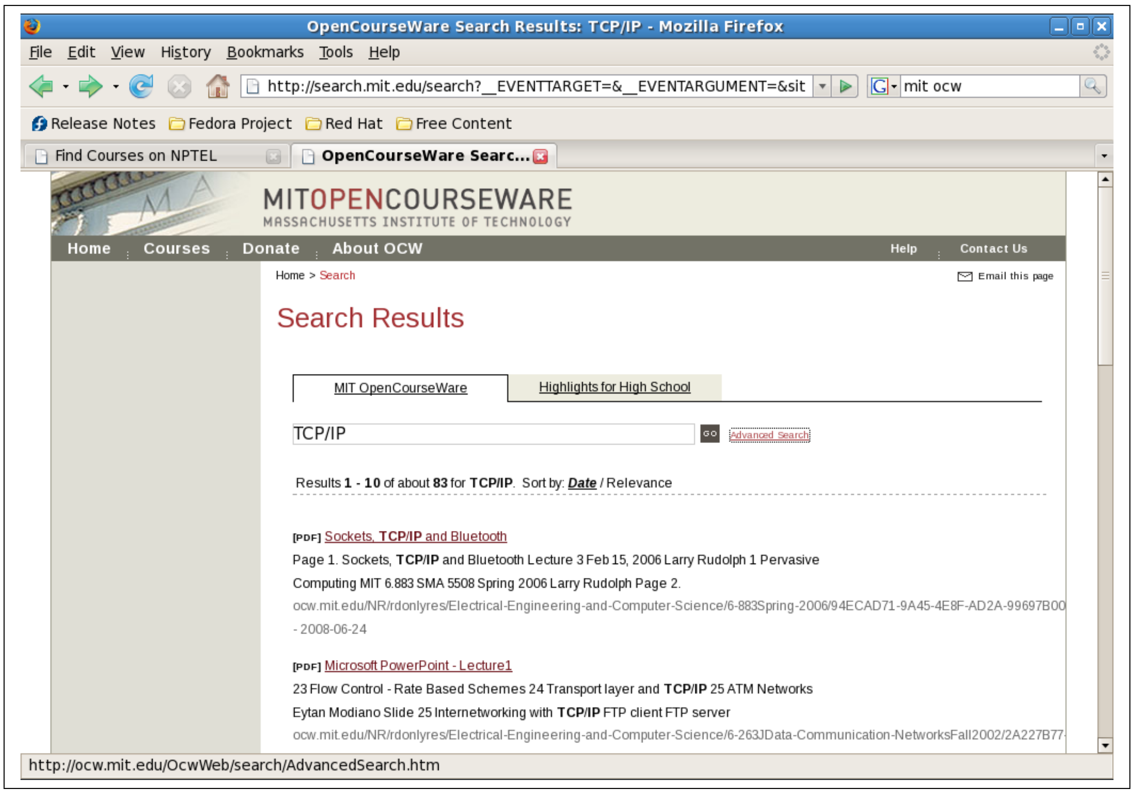
2.1.3 CDEEP IIT Bombay

The Centre for Distance Engineering Education Programme has been established to make IIT's courses available to the Student community at large. There are courses available in fundamental subjects, advance subjects. Different activities of CDEEP include laboratory demonstrations, transmitting classroom lectures live to the destination, develop web-based course material, tutorials, assignments, studio recording of lectures etc [CDE].

Currently, there is no search facility in CDEEP which can display for a particular query that whether it exist in any of the courses which are there in CDEEP curriculum or not.

2.2 Learning Management Systems

A Learning Management System (LMS) is a set of software tools designed to facilitate user learning interventions. Most LMSs we surveyed are web-based to facilitate ' 'anytime, any


OpenCourseWare Search Results: TCP/IP - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://search.mit.edu/search?__EVENTTARGET=&__EVENTARGUMENT=&sit
mit ocw
Release Notes
Fedora Project
Red Hat
Free Content
Find Courses on NPTEL
OpenCourseWare Searc...
MITOPENCOURSEWARE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Home
Courses
Donate
About OCW
Help
Contact Us
Home > Search
Email this page
MIT OpenCourseWare
Highlights for High School
GO
Advanced Search
http://ocw.mit.edu/OcwWeb/search/AdvancedSearch.htm

Search Results

TCP/IP

Results **1 - 10** of about **83** for **TCP/IP**. Sort by: ***Date*** / Relevance

[PDF] Sockets, TCP/IP and Bluetooth
Page 1. Sockets, **TCP/IP** and Bluetooth Lecture 3 Feb 15, 2006 Larry Rudolph 1 Pervasive
Computing MIT 6.883 SMA 5508 Spring 2006 Larry Rudolph Page 2.
ocw.mit.edu/NR/rdonlyres/Electrical-Engineering-and-Computer-Science/6-883Spring-2006/94ECAD71-9A45-4E8F-AD2A-99697B00
- 2008-06-24

[PDF] Microsoft PowerPoint - Lecture1
23 Flow Control - Rate Based Schemes 24 Transport layer and **TCP/IP** 25 ATM Networks
Eytan Modiano Slide 25 Internetworking with **TCP/IP** FTP client FTP server
ocw.mit.edu/NR/rdonlyres/Electrical-Engineering-and-Computer-Science/6-263JData-Communication-NetworksFall2002/2A227B77-

Figure 2.2: Screen-shot of MIT's OCW

place, any pace ' ' access to learning content and administration. LMSs are based on a variety of development platforms, from Java architectures to Microsoft.net and usually employ the use of a database back-end [Lea]. The learning environment used by universities and colleges allow instructors to manage their courses and exchange information with students for a course that in most cases will last several weeks and will meet several times during those weeks. While in case of the corporate world a course may be much shorter, completed in single instructor-led or online session.

2.2.1 ATutor

ATutor is an Open Source Web-based **Learning Management System** [ATu].
ATutor is used in various contexts, including online course management, continuing professional development for teachers, career development, and academic research. ATutor is used internationally and has been translated into over fifteen languages with support for over forty additional language modules currently under development.

- Two, of many, accessibility features in the system are text alternatives for all visual elements, and keyboard access to all elements of the program. With these features, a blind person can listen to the entire interface of the system with the help of a screen reader, and he or she can access the system without needing a mouse.

- ATutor is also designed for adaptability to any of several teaching and learning scenarios. There are four main areas that reflect this design principle: themes, privileges, tool modules, and groups.

The ATutor theme system to allow administrators to easily customize the look and layout of the system to their particular needs. Themes are used to give ATutor a new look, to give categories of courses their own look, or to provide multiple versions of ATutor on a single system, from which users could choose one as a preference setting.

2.2.2 OLAT

OLAT is the acronym for Online Learning and Training. It is a web application - a so called Learning Management System which supports any kind of online learning, teaching, and tutoring [OLA]. OLAT is free open source software OLAT has a lot of features for e-learning platforms such as content managing, forums, file discussions, quizzes with different kinds of questions, surveys, chat, submission modules etc.

There are other open source LMSs also available such as Moodle, SCORM, eFront etc.

Every type of LMS supports the following features:

- Manage users, roles, courses, instructors, and facilities and generate reports.
- Course calendar
- Learner messaging and notifications
- Assessment/testing capable of handling student pre/post testing
- Display scores and transcripts

2.3 Choice of Search Engine

Each search engine has multiple characteristics that differentiates it from the other engines. Each of the search engines can be characterized by the features they implement as well as the performance they have in different scenarios. There are many search engines available based on a number of factors such as open source, information type, business, metasearch engines etc. [sea]. Of these, we have chosen to use Lucene [HG04], for the following reasons:

1. Lucene is not a standalone search application. Instead it is an API [Luc]. It can be integrated in an application as a search and indexing API. Nutch is one popular search engine that uses Lucene. Nutch does the functions of crawling and parsing and uses Lucene for indexing. However, in our case, we do not need all the features of Nutch. The API provided by Lucene are sufficient.

2. Lucene has been completely written in Java. Our system is also built in Java, hence it is easy to integrate.

3. Lucene does not have any crawling and HTML parsing functionality. We do not require these as we are targetting specific repositories. Also, we do not require any front-end functionalities.

Chapter 3

Overview of Solutions

As mentioned earlier, we have currently taken NPTEL as our target content repository. In the examples described in this chapter we will take the course Computer Networks from NPTEL as a running example. This course has 40 pdf files in it. For every course what we have to do is:

1. Parse the pdf files to get textual data, using PDFbox.
2. Index the text files, using Lucene.
3. Analyze the keywords and their frequencies in the index, using LUKE.
4. Find pre-requisites and follow-ups for every file, using heuristics developed in this project.
5. Generate a DAG for the whole course, using DOT.

The step-by-step workflow of the processing of each step is shown in Figure 3.1.
Solutions for all the steps are described one by one in detail.

3.1 Parsing

Lucene requires the pdf files to be converted to text before indexing them. We made use of a Nutch utility called PDFbox to parse the pdf files. PDFBox is a Java library for extracting text contents from existing PDF documents. The implementation details of invoking PDFbox are described in the next chapter.

3.2 Indexing text data

We can embed Lucene easily into our applications and implement indexing and searching functionality. For specifically giving the right learning module we need to first index it. The basic code involving all the classes needed to index is given in the Appendix B. The main classes used for indexing are shown below:

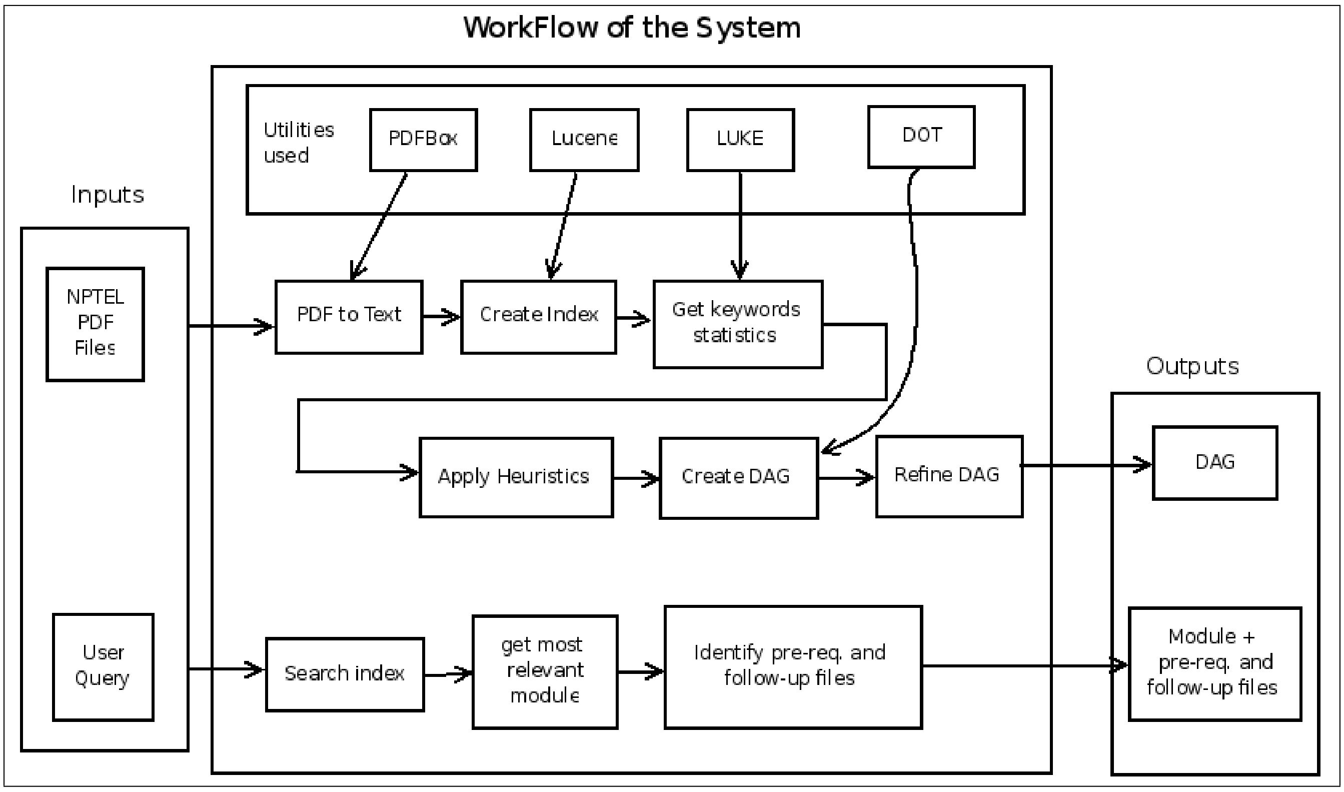


Figure 3.1: Screen-shot of the workflow

Listing 3.1: simple lucene index

```
Analyzer analyzer = new StopAnalyzer();
        StringBuffer contents = new StringBuffer();
        Directory directory = FSDirectory.getDirectory("/home/nidhi/Desktop/Computer_Networks/index");
        IndexWriter iwriter = new IndexWriter(directory, analyzer, true);
        iwriter.setMaxFieldLength(25000);
```

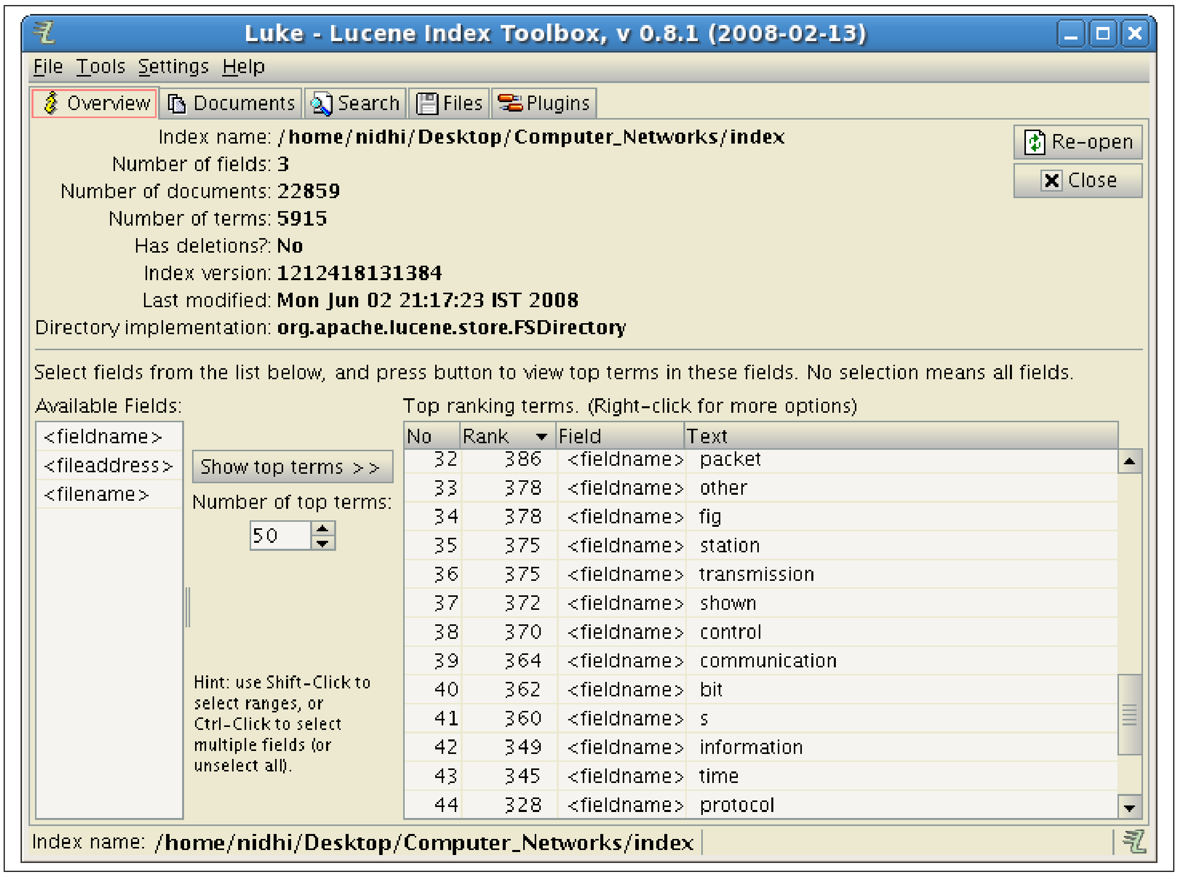
3.3 Analysing the Index

Once we have indexed our text data, we can print the index in different number of types. We can print the index according to the frequency of the terms, and even highlight the occurrence of the terms in the content. Lucene has its own default scoring mechanism on the basis of which it calculates the score of the documents and displays them in some order of preference. In our experiments we have taken the count of the index terms, i.e., the number of times each term occurs in each file. Then further processing is performed.

3.3.1 Refining counts

We give the 40 lecture files (after converting to text) of the course Computer Networks course to Lucene. Lucene indexes the text files and allows us to view and print the index with the help of LUKE (ref. Chapter 4 for more details). From there we get a collection of index terms (in the same way as it is in a book). We remove the non-technical English words, as Lucene indexes all the text data except some particular English words. The words which are removed from this course can also be removed from other courses so that we need not again and again remove them from other course's indexes.

A sample screenshot of LUKE output for the course Computer Networks is given below in Figure 3.2. Then we use the following strategies for refining the counts of the keywords.



Index name: /home/nidhi/Desktop/Computer_Networks/index
Number of fields: 3
Number of documents: 22859
Number of terms: 5915
Has deletions?: No
Index version: 1212418131384
Last modified: Mon Jun 02 21:17:23 IST 2008
Directory implementation: org.apache.lucene.store.FSDirectory

Select fields from the list below, and press button to view top terms in these fields. No selection means all fields.

Available Fields:
<fieldname>
<fileaddress>
<filename>

Hint: use Shift-Click to select ranges, or Ctrl-Click to select multiple fields (or unselect all).

Top ranking terms. (Right-click for more options)

No	Rank	Field	Text
32	386	<fieldname>	packet
33	378	<fieldname>	other
34	378	<fieldname>	fig
35	375	<fieldname>	station
36	375	<fieldname>	transmission
37	372	<fieldname>	shown
38	370	<fieldname>	control
39	364	<fieldname>	communication
40	362	<fieldname>	bit
41	360	<fieldname>	s
42	349	<fieldname>	information
43	345	<fieldname>	time
44	328	<fieldname>	protocol

Index name: /home/nidhi/Desktop/Computer_Networks/index

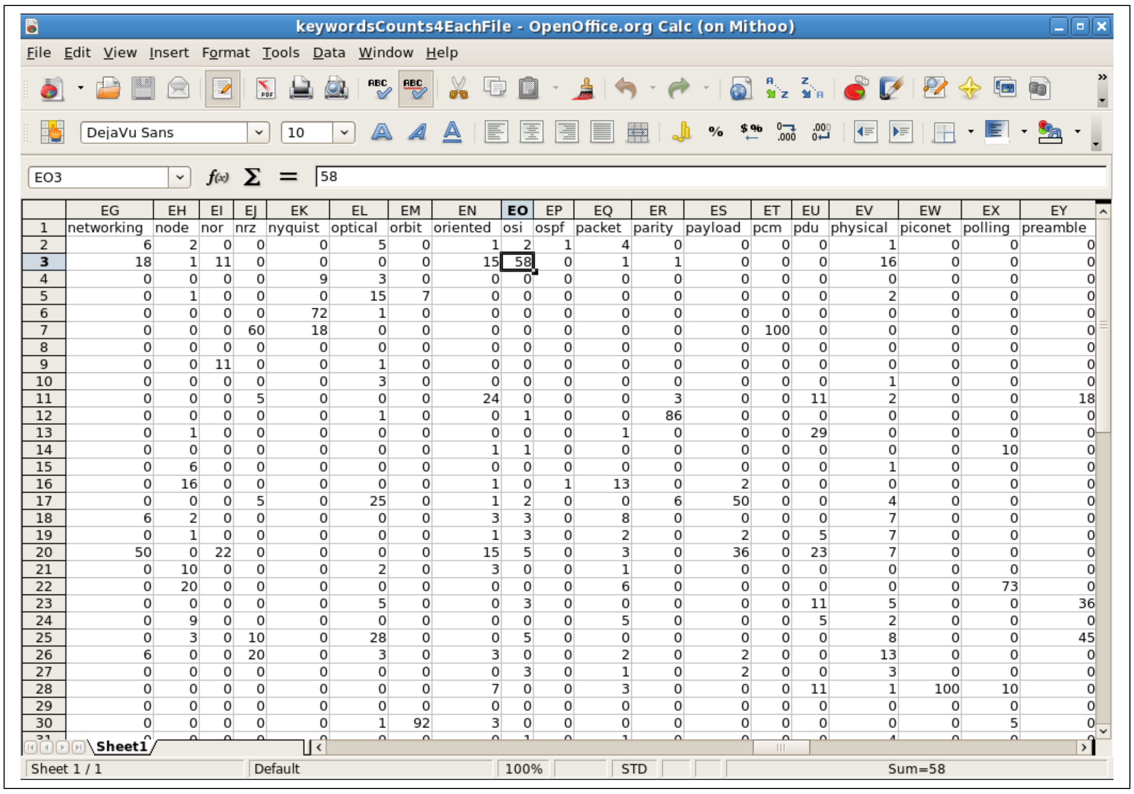
Figure 3.2: Screen-shot of Luke

3.3.2 Without any Threshold

In this strategy, we simply take the count of all keywords in each file. Note that we do not take the scores of the keywords in the document given by Lucene. Instead we have taken the counts of the keywords in the document as this gives more accurate results.

A sample screen shot for the table showing counts of all keywords is given in Figure 3.3.

It is not sufficient to simply use the absolute values of the counts, since these could be misleading. For example, in Figure 3.3, highest count of OSI is 58 and there are other smaller counts such as 2, 1 ,2,3 etc. These counts are not meaningful because there are other keywords in these files whose counts are much higher. So, we need not keep those valuse. So, we need to apply some kind of normalization in order to capture the relative importance of the keywords.


keywordsCounts4EachFile - OpenOffice.org Calc (on Mithoo)
File Edit View Insert Format Tools Data Window Help
DejaVu Sans
10
EO3
58
Sheet1
Sheet 1 / 1
Default
100%
STD
Sum=58

	EG	EH	EI	EJ	EK	EL	EM	EN	EO	EP	EQ	ER	ES	ET	EU	EV	EW	EX	EY
1	networking	node	nor	nrz	nyquist	optical	orbit	oriented	osi	ospf	packet	parity	payload	pcm	pdu	physical	piconet	polling	preamble
2	6	2	0	0	0	5	0	1	2	1	4	0	0	0	0	1	0	0	0
3	18	1	11	0	0	0	0	15	58	0	1	1	0	0	0	16	0	0	0
4	0	0	0	0	9	3	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	1	0	0	0	15	7	0	0	0	0	0	0	0	0	2	0	0	0
6	0	0	0	0	72	1	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	60	18	0	0	0	0	0	0	0	0	100	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	11	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	1	0	0	0
11	0	0	0	5	0	0	0	24	0	0	0	3	0	0	11	2	0	0	18
12	0	0	0	0	0	1	0	0	1	0	0	86	0	0	0	0	0	0	0
13	0	1	0	0	0	0	0	0	0	0	1	0	0	0	29	0	0	0	0
14	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	10	0
15	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
16	0	16	0	0	0	0	0	1	0	1	13	0	2	0	0	0	0	0	0
17	0	0	0	5	0	25	0	1	2	0	0	6	50	0	0	4	0	0	0
18	6	2	0	0	0	0	0	3	3	0	8	0	0	0	0	7	0	0	0
19	0	1	0	0	0	0	0	1	3	0	2	0	2	0	5	7	0	0	0
20	50	0	22	0	0	0	0	15	5	0	3	0	36	0	23	7	0	0	0
21	0	10	0	0	0	2	0	3	0	0	1	0	0	0	0	0	0	0	0
22	0	20	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	73	0
23	0	0	0	0	0	5	0	0	3	0	0	0	0	0	11	5	0	0	36
24	0	9	0	0	0	0	0	0	0	0	5	0	0	0	5	2	0	0	0
25	0	3	0	10	0	28	0	0	5	0	0	0	0	0	0	8	0	0	45
26	6	0	0	20	0	3	0	3	0	0	2	0	2	0	0	13	0	0	0
27	0	0	0	0	0	0	0	0	3	0	1	0	2	0	0	3	0	0	0
28	0	0	0	0	0	0	0	7	0	0	3	0	0	0	11	1	100	10	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	1	92	3	0	0	0	0	0	0	0	0	0	5	0

Figure 3.3: Screen-shot of the Keyword Count table

3.3.3 Mean Threshold

We initially tried a simple idea of taking the Mean of the counts for each keyword and ignoring values that are below the mean. Similarly, we can take the Mean of counts of the keywords in each file and ignore keywords that are below the mean.

For example, if we take a look at the counts of the term " csma ", it has the maximum count of 48 and minimum count is 1. Other counts for the keyword Frame are 1, 2 and 7. So we take the mean of all the values and ignore the values which are less than the mean. These values are simply assigned zero.

However, this idea does not work when there are some keywords which are not occurring in more than 2-3 files. Their counts in those files are also less.Those values will be discarded due to the mean threshhold. But we need to retain such values because there are no other modules for those keywords. Hence we do not use the Mean Threshold idea but modify it to the Percentage Threshold as described below.

3.3.4 Percentage ThreshHold

In this strategy, instead of taking the Mean of the counts for each keyword and ignoring values that are below the mean, we normalize the values. We take the sum of the counts of all the occurrences of that keyword. For each occurrence, we replace its absolute count with its percentage from the total.As a result of percentageThreshHold, pre-requisites for the query " OSI " are " Internet Protocol" , " Transport and Application Layer Protocols " and " Layered Network Architecture ".

Subsequently, the values whose percentage count is less than 10 are discarded. For more explanation refer to chapter 5.

3.4 Find out Pre-requisites and Follow-ups

The lecture modules are given in a seqeuential order. We have not assumed any other information about the dependencies of the concepts in the course. In order to facilitate the learning of user, we now have to give him pre-requisites and follow-up modules also for his query. We now describe a few heuristics used to identify these pre-requisites and follow-ups.

3.4.1 H1: Take count of words

For a given file we take its topKwords. For those topKwords we take the topKfiles. For each file in those topKfiles, we simply check for how many keywords is the given file is appearing in the topKfiles. After that we sort all the topKfiles (without duplicates) and assign the count. Then we arrange the files in ascending order of the counts. Those files whose module numbers are less than the module number of the file for which we are finding pre-requisites, are assigned as pre-requisites. Those whose module numbers are greater are assigned as the follow-ups.

Algorithm 1 Heuristic 1

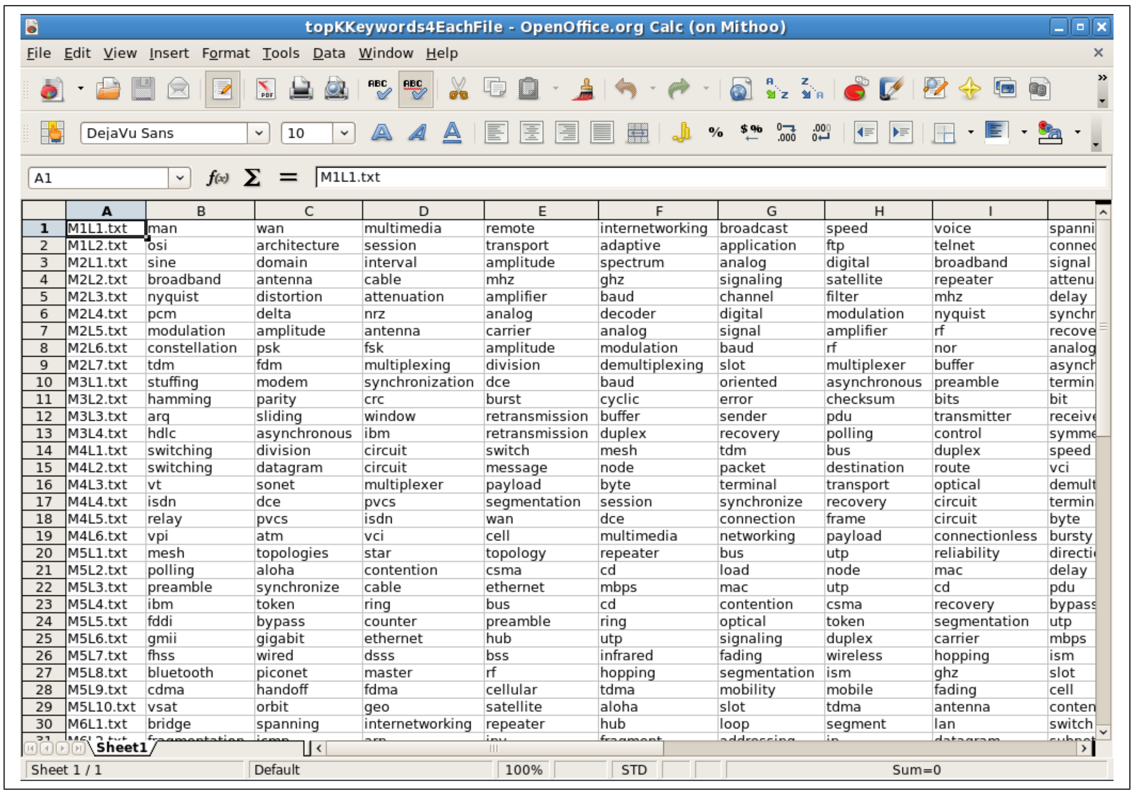
```
1: Take count of each Keyword in each file.
2: Store keywords as columns and files as rows.
3: For each file get topKkeywords
4: For each keyword
5: Sort the file entries and get unique files
6: Assign weight to each file based on sum of counts of all keywords appearing in it.
7: Order the files according to their weights.
8: For files whose index = 1 to $i - 1$; get the topK files according to weight.
9: For files whose index $>$ i;get the topK files according to weight.
```

The top 3 files whose counts are less than the serial number of the given file will be considered as the pre-requisites. In the same way 3 top files whose counts are greater than the given file will be given as the follow ups.

The Figure 3.4 below shows a view of topkWords for each file:

3.4.2 H2: Take position of the file

With the above approach it was the possibility that the file which was more important was not appearing at the correct position and in the correct order. So what we decided that instead of just taking the count, we shall also consider the position of the file in the topKfiles. We will follow the simple formula c = k-p+1; where c will denote the final count of the file. K denotes the total top files (in our case k = 5). p will denote the position of the file (varies from 1 to 5).



topKKeywords4EachFile - OpenOffice.org Calc (on Mithoo)

A1: M1L1.txt

	A	B	C	D	E	F	G	H	I	
1	M1L1.txt	man	wan	multimedia	remote	internetworking	broadcast	speed	voice	spanni
2	M1L2.txt	osi	architecture	session	transport	adaptive	application	ftp	telnet	connec
3	M2L1.txt	sine	domain	interval	amplitude	spectrum	analog	digital	broadband	signal
4	M2L2.txt	broadband	antenna	cable	mhz	ghz	signaling	satellite	repeater	attenu
5	M2L3.txt	nyquist	distortion	attenuation	amplifier	baud	channel	filter	mhz	delay
6	M2L4.txt	pcm	delta	nrz	analog	decoder	digital	modulation	nyquist	synchr
7	M2L5.txt	modulation	amplitude	antenna	carrier	analog	signal	amplifier	rf	recove
8	M2L6.txt	constellation	psk	fsk	amplitude	modulation	baud	rf	nor	analog
9	M2L7.txt	tdm	fdm	multiplexing	division	demultiplexing	slot	multiplexer	buffer	asynch
10	M3L1.txt	stuffing	modem	synchronization	dce	baud	oriented	asynchronous	preamble	termin
11	M3L2.txt	hamming	parity	crc	burst	cyclic	error	checksum	bits	bit
12	M3L3.txt	arq	sliding	window	retransmission	buffer	sender	pdu	transmitter	receive
13	M3L4.txt	hdlc	asynchronous	ibm	retransmission	duplex	recovery	polling	control	symme
14	M4L1.txt	switching	division	circuit	switch	mesh	tdm	bus	duplex	speed
15	M4L2.txt	switching	datagram	circuit	message	node	packet	destination	route	vci
16	M4L3.txt	vt	sonet	multiplexer	payload	byte	terminal	transport	optical	demult
17	M4L4.txt	isdn	dce	pvcs	segmentation	session	synchronize	recovery	circuit	termin
18	M4L5.txt	relay	pvcs	isdn	wan	dce	connection	frame	circuit	byte
19	M4L6.txt	vpi	atm	vci	cell	multimedia	networking	payload	connectionless	bursty
20	M5L1.txt	mesh	topologies	star	topology	repeater	bus	utp	reliability	directi
21	M5L2.txt	polling	aloha	contention	csma	cd	load	node	mac	delay
22	M5L3.txt	preamble	synchronize	cable	ethernet	mbps	mac	utp	cd	pdu
23	M5L4.txt	ibm	token	ring	bus	cd	contention	csma	recovery	bypass
24	M5L5.txt	fddi	bypass	counter	preamble	ring	optical	token	segmentation	utp
25	M5L6.txt	gmii	gigabit	ethernet	hub	utp	signaling	duplex	carrier	mbps
26	M5L7.txt	fhss	wired	dsss	bss	infrared	fading	wireless	hopping	ism
27	M5L8.txt	bluetooth	piconet	master	rf	hopping	segmentation	ism	ghz	slot
28	M5L9.txt	cdma	handoff	fdma	cellular	tdma	mobility	mobile	fading	cell
29	M5L10.txt	vsat	orbit	geo	satellite	aloha	slot	tdma	antenna	conten
30	M6L1.txt	bridge	spanning	internetworking	repeater	hub	loop	segment	lan	switch

Sheet 1 / 1 Default 100% STD Sum=0

Figure 3.4: Screen-shot of the TopKWords for each file table

Algorithm 2 Heuristic 2; Without any threshold,take keyword position; i is the index of the file for which we have to find requisites

```
1: Take count of each Keyword in each file.
2: Store keywords as columns and files as rows.
3: For each file get topKkeywords.
4: For each keyword get topKfiles.
5: Sort the file entries and get unique files.
6: For each file take position of the file for each keyword in topKfiles.
7: Assign weight as w = K-p+1.
8: For files whose index = 1 to $i - 1$; get the topK files.
9: For files whose index $>$ i;get the topK files.
```

After experiments we observed that it gave better ordering of the results with more relevant files on correct positions.

3.4.3 H3: Take the Average count

Another approach we tried for the correct ordering of the pre-requisites and follow-ups is to take the counts of the keywords appearing in that file. We took average counts of each file and then sorted according to the average count.

Algorithm 3 Heuristic 3; With Percentage threshold,take average count; i is the index of the file for which we have to find requisites

```
1: Take count of each Keyword in each file.
2: tore keywords as columns and files as rows.
3: For each file get topKkeywords
4: For each keyword get topKfiles
5: Sort the file entries and get unique files
6: Assign weight to each file based on the average of sum of counts of all keywords appearing
   in it.
7: Order the files according to their weights.
8: For files whose index = 1 to $i - 1$; get the topK files according to weight.
9: For files whose index $>$ i;get the topK files according to weight.
```

Another variation to this approach can be that we take the counts of the keywords appearing in that file and divide by K(K = 5 in our case).

3.4.4 H4: Find correlation

Another heuristic we applied orders the files based on their correlation with the file for which we are finding pre-requisites and follow-ups. Suppose i is the index of the file which we are currently processing. we have count of each keyword in this file. For all the other files in the array, we will correlate the corresponding entries with those of the ith file's entries. We multiply the corresponding entries and take summation of the resulting counts. In this way we get another approach to order the files. After this, we will sort the files according to the latest summation computed and separate out the pre-requisites and follow-ups.

Algorithm 4 Heuristic 4; With Percentage threshold,find correlationt; i is the index of the file for which we have to find requisites

```
1: Take count of each Keyword in each file.
2: Store keywords as columns and files as rows.
3: For each file get topKkeywords
4: For each keyword get topKfiles
5: Sort the file entries and get unique files
6: Multiply all keyword entries of the ith file to those of the others.
7: Take sum of the resulting counts.
8: For files whose index = 1 to $i - 1$; get the topK files according to weight.
9: For files whose index $>$ i;get the topK files according to weight.
```

By applying this heuristic the counts are increased accordingly due to multiplication and when applied with the percentage threshold gives promising results

3.5 Generate DAG

Now, we have found out pre-requisites and follow-up files for all the files of a course. Our next task is to generate dependency graph for the whole course. The purpose of creating dependency graph is to let the user decide what should he know before learning any specific topic. By viewing the dependency graph the user can easily identify which modules need to be learnt before any specific topic and what can be learnt after that[ref. chapter 4 for more detail].We have captured all the dependencies between the files and later we draw dependency graph with the help of those dependency.

3.5.1 Refining the Graph

Now we have set of pre-requisites for all the chapters of the Computer Networks course. We need to show them with the help of dependency graph. But while making the dependency graph

we observed that the graph is kind of messy and has asymmetric relations between the pre-requisites. So, we need to refine the dependencies for every file.
Let us say X and Y are two different files. When we make the dependency graph then we need to check out that we are showing only symmetric relations in our graph to make the graph more clean and readable. For every link between X and Y in the graph we need to ensure that there exists a link between X and Y iff X is a pre-requisite for Y and Y is a follow up of X. This way the graph will clearly depict the dependencies between the files in the course. The graph below in Figure 3.5 shows dependencies in “Computer Networks” course.

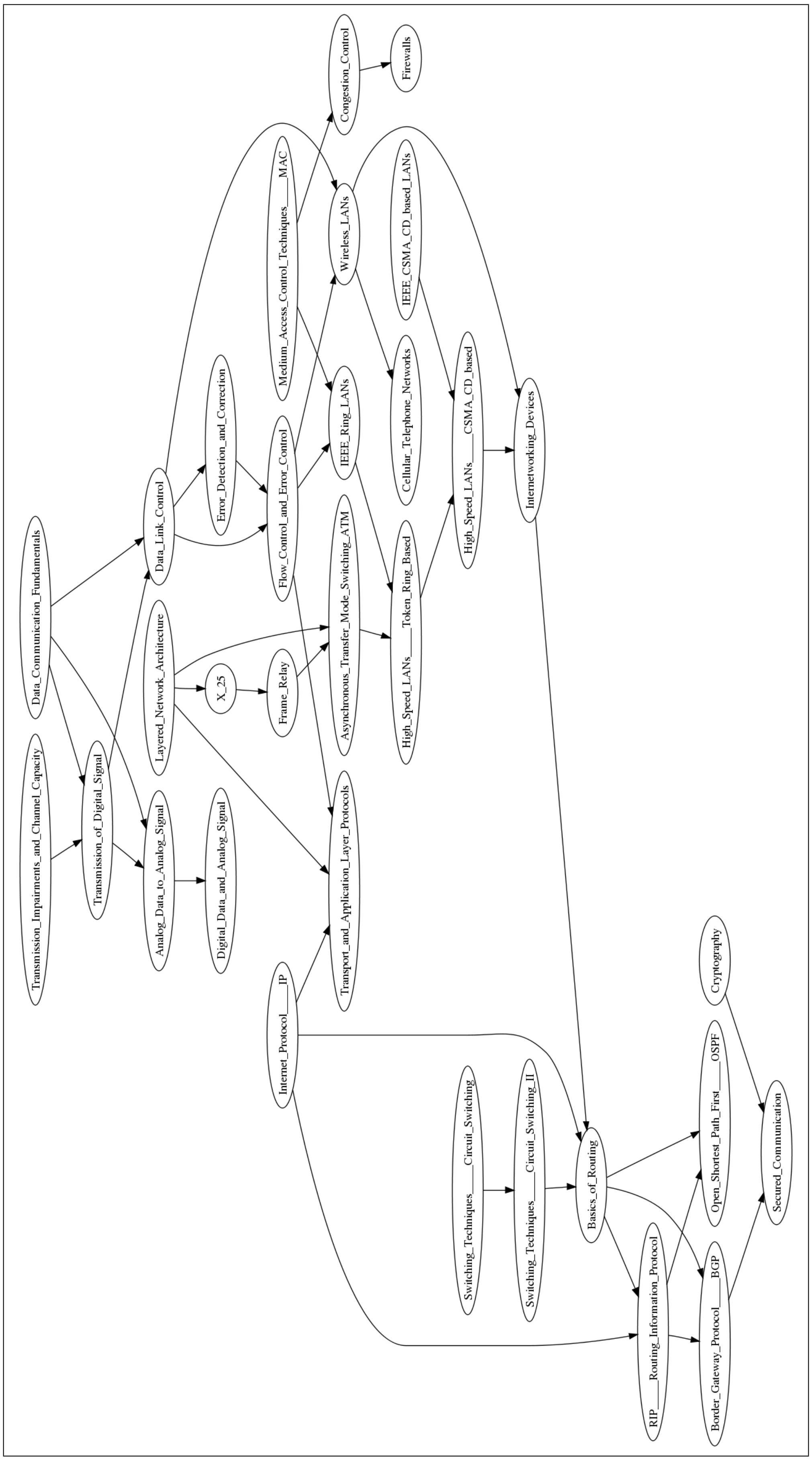

Data_Communication_Fundamentals
Transmission_Impairments_and_Channel_Capacity
Transmission_of_Digital_Signal
Analog_Data_to_Analog_Signal
Digital_Data_and_Analog_Signal
Data_Link_Control
Layered_Network_Architecture
Error_Detection_and_Correction
X_25
Frame_Relay
Flow_Control_and_Error_Control
Medium_Access_Control_Techniques____MAC
Asynchronous_Transfer_Mode_Switching_ATM
Transport_and_Application_Layer_Protocols
Internet_Protocol____IP
Congestion_Control
Firewalls
Wireless_LANs
IEEE_Ring_LANs
Cellular_Telephone_Networks
IEEE_CSMA_CD_based_LANs
High_Speed_LANs____Token_Ring_Based
High_Speed_LANs____CSMA_CD_based
Internetworking_Devices
Switching_Techniques____Circuit_Switching
Switching_Techniques____Circuit_Switching_II
Basics_of_Routing
RIP____Routing_Information_Protocol
Border_Gateway_Protocol____BGP
Open_Shortest_Path_First____OSPF
Cryptography
Secured_Communication

Figure 3.5: DAG for the Computer Networks course

Chapter 4

Implementation Details

In this chapter we will see how all the processing is being done and which are the components involved. We will one by one see the processing of each phase.

4.1 Parsing with PDFBox

As Lucene indexes only text information, we need to convert pdf files to text files. For that we used PDFBox which is an open source Java PDF library for working with PDF documents. It allows creation of new PDF documents, manipulation of existing documents and to extract content from documents. This functionality is encapsulated in the org.pdfbox.util.PDFTextStripper. PDFBox also includes several other command line utilities such as merger PDF documents, create images from pdf pages, print pdf etc. The code snippet is as shown below:

Listing 4.1: simple lucene index

```
File filename1 = new File("/home/nidhi/Desktop/SAD/pdf/M14L1" +".pdf");
  PDDocument doc1 = PDDocument.load(filename1);
  PDFTextStripper stripper = new PDFTextStripper();
  System.out.println(stripper.getText(doc1));
  String str1=stripper.getText(doc1);
  File outFile = new File("/home/nidhi/Desktop/SAD/TextFiles/M14L1" +".txt");
  FileWriter out = new FileWriter(outFile);
        out.write(str1);
        out.close();
```

.

4.1.1 Using Lucene to Index and Search

Lucene is a free open source information retrieval library, originally created in java.[HG04]. It lets the user add indexing and searching capabilities to their applications. Lucene doesn't care about the source of the data, its format, or even its language, as long as we can convert it to text [Lucene] This means we can use Lucene to index and search data stored in files:

web pages on remote web servers, documents stored in local file systems, simple text files, Microsoft Word documents, HTML or PDF files, or any other format from which we can extract textual information. There is no efficient search facility provided in NPTEL. The search option provided just works for the course names and one can't get any information about a particular query if it does not appear in the course name.

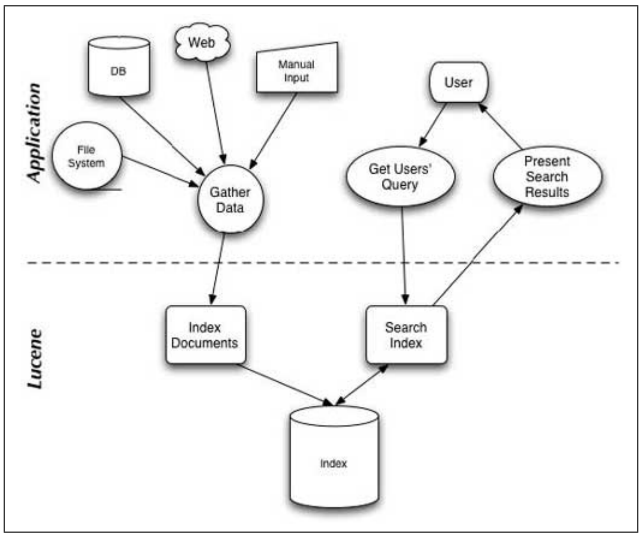


Figure 4.1: Architecture of Lucene [Lucene]

4.1.2 Indexing

Suppose we needed to search a large number of files, and we want to be able to find files that contained a certain word or a phrase. A naive approach to do this would be to sequentially scan each file for the given word or phrase. This approach has a number of flaws, the most obvious of which is that it does not scale to larger file sets or cases where files are very large. This is where we need indexing : To search large amounts of text quickly, we must first index that text and convert it into a format that will let us search it rapidly, eliminating the slow sequential scanning process. This conversion process is called indexing, and its output is called an index. Most of the content has been made available in the form of PDF documents. Some of these contain scans of material handwritten or drawn by the instructor and image captures of the blackboard contents.

Creating an index

In this section we will see how to use a single class called Indexer and its static methods together, they recursively traverse file system directories and index all files with a .txt extension. When Indexer completes execution it leaves behind a Lucene index for the Searcher to search upon. The core indexing classes and how we used them in our program is described as follows:

- **IndexWriter** is the central component of the indexing process. This class creates a new index and adds documents to an existing index. Index-Writer gives us write access to the index but does not let us read or search it. We can also run Indexer from the command-line. It takes two arguments: A path to a directory where we store the Lucene index and path to a directory that contains the files we want to index.

- **Directory** class represents the location of a Lucene index. It is an abstract class that allows its subclasses (two of which are included in Lucene) to store the index. If we have to store our index on the disk then we need to use FSDirectory, a Directory subclass that maintains a list of real files in the file system.

  The other implementation of Directory is a class called RAMDirectory. Although it exposes an interface identical to that of FSDirectory, RAMDirectory holds all its data in memory. This implementation is therefore useful for smaller indices that can be fully loaded in memory and can be destroyed upon the termination of an application.

- Analyzer

  Before text is indexed, it is passed through an Analyzer. The Analyzer, specified in the IndexWriter constructor, is in charge of extracting tokens out of text to be indexed and eliminating the rest. If the content to be indexed is not plain text, it should first be converted to it. Analyzer is an abstract class, but Lucene has with several implementations of it. Some of them deal with skipping stop words (frequently used words that do not help distinguish one document from the other, such as a, an, the, in, and on); some deal with conversion of tokens to lowercase letters, so that searches are not case-sensitive; and so on. Analyzers are an important part of Lucene and can be used for much more than simple input filtering. For a developer integrating Lucene into an application, the choice of analyzer(s) is a critical element of application design.

- Document

  A Document represents a collection of fields. It can be a web page, an email message, or a text file that we want to retrieve at a later time. Fields of a document represent the document or meta-data associated with that document. For each text file we find, we create a new instance of the Document class, populate it with Fields and add that Document to the index,effectively indexing the file.

- Field

  Lucene offers four different types of fields from which you can choose. Each field corresponds to a piece of data that is either queried against or retrieved from the index during search.

- Keyword

  It is not analyzed, but is indexed and stored.This type is suitable for fields whose original value should be preserved in its entirety, such as URLs, file system paths, dates, personal names, Social Security numbers, telephone numbers, and so on. For example, we used the file system path in Indexer as a Keyword field.

- Unindexed

  Is neither analyzed nor indexed, but its value is stored in the index as is. This type is suitable for fields that we need to display with search results (such as a URL or database primary key), but whose values we will never search directly.

- UnStored

  The opposite of UnIndexed. This field type is analyzed and indexed but is not stored in the index. It is suitable for indexing a large amount of text that does not need to be retrieved in its original form, such as bodies of web pages, or any other type of text document.

- Text

  Is analyzed, and is indexed. This implies that fields of this type can be searched against, but we have to be cautious about the field size.

4.1.3 Searching the Index

The Searcher program complements Indexer and it also provides command-line searching capability. It will take two arguments: The path to the index created with Indexer and a query to use to search the index.

The core searching classes are

- IndexSearcher

  It is a class that opens an index in a read-only mode. It offers a number of search methods, the simplest taking takes a single Query object as a parameter and returning a hits object.

- Term

  A Term is the basic unit for searching. Similar to the Field object, it consists of a pair of string elements: the name of the field and the value of that field. Because the TermQuery object is derived from the abstract parent class Query, it can be used with the Query type on the left side of the statement.

- Query

  Lucene comes with a number of concrete Query subclasses. The most basic Lucene Query is TermQuery. Other Query types are BooleanQuery, PhraseQuery, prefixQuery, PhrasePrefixQuery, RangeQuery, FilteredQuery, and SpanQuery.

- TermQuery

  TermQuery is the most basic type of query supported by Lucene, and it is one of the primitive query types. It is used for matching documents that contain fields with specific values. A term query accepts a single term such as:

- Hits

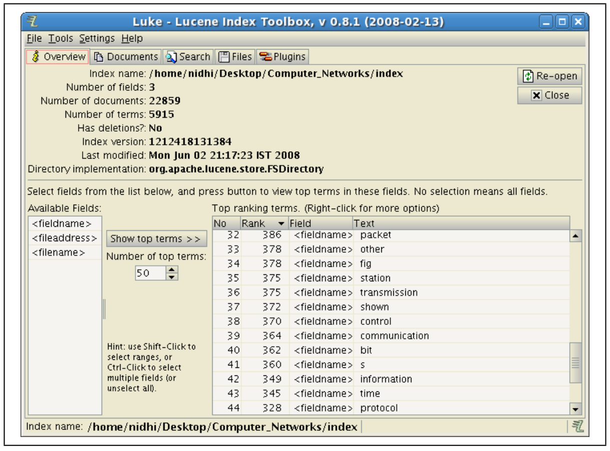
  This class simply points to the ranked search result documents that match a given query. The hits collection is ordered by score by default.

4.2 Using LUKE

The fundamental concepts in Lucene are index, document, field and term. An index contains a sequence of documents.

- A document is a sequence of fields.
- A field is a named sequence of terms.
- A term is a string.

Lucene's index falls into the family of indexes known as an inverted index. This is because it can list, for a term, the documents that contain it. This is the inverse of the natural relationship, in which documents list terms. When Lucene creates index it generates three files which are



Index name: /home/nidhi/Desktop/Computer_Networks/index
Number of fields: 3
Number of documents: 22859
Number of terms: 5915
Has deletions?: No
Index version: 1212418131384
Last modified: Mon Jun 02 21:17:23 IST 2008
Directory implementation: org.apache.lucene.store.FSDirectory

Select fields from the list below, and press button to view top terms in these fields. No selection means all fields.

Available Fields:
<fieldname>
<fileaddress>
<filename>

Hint: use Shift-Click to select ranges, or Ctrl-Click to select multiple fields (or unselect all).

Top ranking terms. (Right-click for more options)

No	Rank	Field	Text
32	386	<fieldname>	packet
33	378	<fieldname>	other
34	378	<fieldname>	fig
35	375	<fieldname>	station
36	375	<fieldname>	transmission
37	372	<fieldname>	shown
38	370	<fieldname>	control
39	364	<fieldname>	communication
40	362	<fieldname>	bit
41	360	<fieldname>	s
42	349	<fieldname>	information
43	345	<fieldname>	time
44	328	<fieldname>	protocol

Index name: /home/nidhi/Desktop/Computer_Networks/index

Figure 4.2: Screen-shot of Luke

not readable. For reading and printing the index we need to use LUKE. [Luk] Luke is a handy development and diagnostic tool, which accesses already existing Lucene indexes and allows us to display and modify their contents in several ways such as browse by document number, or by term, view documents and copy them, see the most frequent terms etc.

4.3 Generating graph with DOT

After we have identified the pre-requisites and follow ups for every chapter, we plan to prepare a dependency graph for the whole course. Having the dependency graph, the user can clearly make out which module he needs to learn in order to learn about a specific topic.
We make use of the DOT utility of the Graphviz package [DOT]. DOT is a plain-text graph description language. It is a simple way of describing graphs that both humans and computer programs can use. DOT graphs are files that end with the .dot extension.
We captured all the dependencies while finding out the pre-requisites and follow-up files for each chapter. Those dependencies are saved in a text file with the .dot extension.Various attributes can be applied to nodes and edges in DOT files. These attributes can control aspects such as color, shape, and line styles.Sample graph are shown in the next chapters.

Code snippet for generating DAG:

Listing 4.2: Generate DAG

```
public static void generateDAG(String filename)
  {
    int requisites[] ;
    int [] topKWords ;
    int [] topKFiles ;
    int [] noOfWordsHavingAFile;

    //String to be written in the file
    String str = "";

    //For each file
    for(int i=0;i<Helper.arrFileNames.length;i++)
    {

    System.out.println("Processing File : " + Helper.extractFileNameFromAbsolutePath(Helper.arrFileNames[i
        ]));
      //System.out.println("Processing File : " + Helper.arrFileNames[i]);
      System.out.println
System.out.println("Processing File : " + Helper.extractFileNameFromAbsolutePath(Helper.arrFileNames [i]) +
    " " + Helper.arrFileTitles[i]);
      //Fetch top K words of the file.
      topKWords = Helper.getTopKWords(Helper.arrFileNames[i]);

      requisites = new int[Helper.arrFileNames.length];
      noOfWordsHavingAFile = new int[requisites.length];
              //*************************************************************************//
      for(int j=0;j<requisites.length;j++)
      {
        requisites[j]=0;
        noOfWordsHavingAFile[j] = 0;
      }
      //Process each word
      for(int j=0;j<topKWords.length;j++)
      {
//  System.out.println("\n\nProcessing word no. : " + topKWords[j]);
        int currentWordIndex = topKWords[j];
        if(currentWordIndex==-1)
          break;
//  System.out.print(Helper.arrTechnicalWords[currentWordIndex] + " : ");
        //Fetch top K files of the word being processed
        topKFiles = Helper.getTopKFiles(Helper.arrTechnicalWords[currentWordIndex]);
        for(int k=0;k<topKFiles.length;k++)
        {
//   System.out.println(topKFiles[k] + " " );
          if(topKFiles[k]==-1)
            break;
//  System.out.print(Helper.extractFileNameFromAbsolutePath(Helper.arrFileNames[topKFiles[k]]) + " ");
          //requisites[topKFiles[k]] += 1;
```

```
                requisites[topKFiles[k]] += (Helper.K − k);
                //requisites[topKFiles[k]] += Helper.result[topKFiles[k]][currentWordIndex];
                //noOfWordsHavingAFile[topKFiles[k]]++;
//    System.out.println(Helper.result[k][j]);

            }}
```

Code snippet for refining DAG:

Listing 4.3: Refine DAG

```
private static void pruneDAG()
  {
    int[] tempArray1,tempArray2,tempArray;
    Vector tempVector;
    for(int i=0;i<Helper.arrFileNames.length;i++)
    {
      tempArray1 = Helper.arrFoundPrerequisites[i];
      tempVector=new Vector();
      for(int j=0;j<tempArray1.length;j++)
      {
        if(tempArray1[j]==-1)
          break;
        if(doesArrayContainValue(Helper.arrFoundFollowups[tempArray1[j]], i, 2))
          tempVector.add(new Integer(tempArray1[j]));
      }
      for(int j=0;j<Helper.MaxNoOfFollowups;j++)
        Helper.arrFoundPrerequisites[i][j]=-1;
      for(int j=0;j<tempVector.size();j++)
        Helper.arrFoundPrerequisites[i][j] = ((Integer)tempVector.get(j)).intValue();


      tempArray1 = Helper.arrFoundFollowups[i];
      tempVector=new Vector();
      for(int j=0;j<tempArray1.length;j++)
      {
        if(tempArray1[j]==-1)
          break;
        if(doesArrayContainValue(Helper.arrFoundPrerequisites[tempArray1[j]], i, 2))
          tempVector.add(new Integer(tempArray1[j]));
      }
      for(int j=0;j<Helper.MaxNoOfFollowups;j++)
        Helper.arrFoundFollowups[i][j]=-1;
      for(int j=0;j<tempVector.size();j++)
        Helper.arrFoundFollowups[i][j] = ((Integer)tempVector.get(j)).intValue();

    }

  }
```

Chapter 5

Experiments

We took NPTEL as the content repository. The examples given in the report are all in reference with the Computer Networks course. Experiments are done on other courses also. For every course, we have maintained specific files such as topkwords for each file, countofKeywords, fileTitles, file containing the pre-requisites and follow-ups for each file of the whole course etc. These files are kept for every course on which we did experiment. Finally a DAG is prepared for every course. The DAG will be presented to the user so that he/she can decide what to learn.

5.1 Testing

We have done experiments for 6 courses namely: Computer Networks, Artificial Intelligence, Software Engineering, embedded Systems, Operating System and Software Analysis and Design.

First of all let's take Computer Networks course. It has 40 pdf files. After parsing them with PDFBox, we got 40 text files. Those 40 text files are given to Lucene for which it will create index. At the time of indexing it creates segments file which are not readable by the user. We used Luke to print the index for the course. From that we got our index terms(keywords). For all those keywords, we took their count in each file in an excel file. We read all the counts in a string and through .csv format, directly wrote into an excel file. We maintained a separate file for topKwords of each file. We have maintained a dag.csv file which shows the pre-requisites and follow-ups for each file.

We have tried all the functions described in the third chapter with both the mean and percentage threshHolds. We observed that results after applying percentageThreshHold with the correlation function-Heuristic H4 are more promising.

Table 5.1: Requisites generated by our Program

File Name	Pre-requisite Files	Follow-up Files
Introduction and Course Otline (M1L1.txt)		• Wireless LANs (M5L7.txt) • Layered Network Architecture (M1L2.txt) • Flow Control and Error Control (M3L3.txt)
Layered Network Architecture (M1L2.txt)		• X 25 (M4L4.txt) • Wireless LANs (M5L7.txt) • Asynchronous Transfer Mode Switching ATM (M4L6.txt)
Data Communication Fundamentals (M2L1.txt)	• Layered Network Architecture (M1L2.txt)	• Transmission of Digital Signal (M2L4.txt) • Analog Data to Analog Signal (M2L5.txt) • Data Link Control (M3L1.txt)
Transmission Media (M2L2.txt)	• Data Communication Fundamentals (M2L1.txt) • Layered Network Architecture (M1L2.txt)	• Data Link Control (M3L1.txt) • Wireless LANs (M5L7.txt) • Synchronous Optical Network SONET (M4L3.txt)

Continued on next page...

Transmission Impairments and Channel Capacity (M2L3.txt)	• Data Communication Fundamentals (M2L1.txt) • Transmission Media (M2L2.txt) • Layered Network Architecture (M1L2.txt)	• Transmission of Digital Signal (M2L4.txt) • Data Link Control (M3L1.txt) • Digital Data and Analog Signal (M2L6.txt)
Transmission of Digital Signal (M2L4.txt)	• Data Communication Fundamentals (M2L1.txt) • Transmission Impairments and Channel Capacity (M2L3.txt) • Layered Network Architecture (M1L2.txt)	• Data Link Control (M3L1.txt) • Synchronous Optical Network SONET (M4L3.txt) • Analog Data to Analog Signal (M2L5.txt)
Analog Data to Analog Signal (M2L5.txt)	• Data Communication Fundamentals (M2L1.txt) • Transmission of Digital Signal (M2L4.txt) • Transmission Media (M2L2.txt)	• Wireless LANs (M5L7.txt) • Digital Data and Analog Signal (M2L6.txt) • Data Link Control (M3L1.txt)
Digital Data and Analog Signal (M2L6.txt)	• Transmission of Digital Signal (M2L4.txt) • Analog Data to Analog Signal (M2L5.txt) • Data Communication Fundamentals (M2L1.txt)	• Data Link Control (M3L1.txt) • Wireless LANs (M5L7.txt) • Synchronous Optical Network SONET (M4L3.txt)

Continued on next page...

Multiplexing of Digital Signals (M2L7.txt)	• Analog Data to Analog Signal (M2L5.txt) • Data Communication Fundamentals (M2L1.txt) • Transmission Impairments and Channel Capacity (M2L3.txt)	• Asynchronous Transfer Mode Switching ATM (M4L6.txt) • Data Link Control (M3L1.txt) • Synchronous Optical Network SONET (M4L3.txt)
Data Link Control (M3L1.txt)	• Transmission of Digital Signal (M2L4.txt) • Data Communication Fundamentals (M2L1.txt) • Layered Network Architecture (M1L2.txt)	• Wireless LANs (M5L7.txt) • Error Detection and Correction (M3L2.txt) • Asynchronous Transfer Mode Switching ATM (M4L6.txt)
Error Detection and Correction (M3L2.txt)	• Data Link Control (M3L1.txt) • Layered Network Architecture (M1L2.txt) • Transmission of Digital Signal (M2L4.txt)	• Wireless LANs (M5L7.txt) • Asynchronous Transfer Mode Switching ATM (M4L6.txt) • Flow Control and Error Control (M3L3.txt)
Flow Control and Error Control (M3L3.txt)	• Data Link Control (M3L1.txt) • Layered Network Architecture (M1L2.txt) • Error Detection and Correction (M3L2.txt)	• Wireless LANs (M5L7.txt) • Transport and Application Layer Protocols (M6L3.txt) • HDLC (M3L4.txt)

Continued on next page...

HDLC (M3L4.txt)	• Data Link Control (M3L1.txt) • Flow Control and Error Control (M3L3.txt) • Error Detection and Correction (M3L2.txt)	• Wireless LANs (M5L7.txt) • X 25 (M4L4.txt) • Asynchronous Transfer Mode Switching ATM (M4L6.txt)
Switching Techniques Circuit Switching (M4L1.txt)	• Layered Network Architecture (M1L2.txt) • Data Communication Fundamentals (M2L1.txt) • Data Link Control (M3L1.txt)	• Asynchronous Transfer Mode Switching ATM (M4L6.txt) • Switching Techniques Circuit Switching II (M4L2.txt) • Frame Relay (M4L5.txt)
Switching Techniques Circuit Switching II (M4L2.txt)	• Switching Techniques Circuit Switching (M4L1.txt) • Layered Network Architecture (M1L2.txt) • Transmission Media (M2L2.txt)	• Basics of Routing (M7L1.txt) • RIP Routing Information Protocol (M7L2.txt) • Border Gateway Protocol BGP (M7L4.txt)
Synchronous Optical Network SONET (M4L3.txt)	• Data Communication Fundamentals (M2L1.txt) • Transmission of Digital Signal (M2L4.txt) • Data Link Control (M3L1.txt)	• Asynchronous Transfer Mode Switching ATM (M4L6.txt) • Frame Relay (M4L5.txt) • High Speed LANs Token Ring Based (M5L5.txt)

Continued on next page...

X 25 (M4L4.txt)	• Layered Network Architecture (M1L2.txt) • Data Link Control (M3L1.txt) • Flow Control and Error Control (M3L3.txt)	• Frame Relay (M4L5.txt) • Asynchronous Transfer Mode Switching ATM (M4L6.txt) • Congestion Control (M7L5.txt)
Frame Relay (M4L5.txt)	• X 25 (M4L4.txt) • Layered Network Architecture (M1L2.txt) • Data Link Control (M3L1.txt)	• Asynchronous Transfer Mode Switching ATM (M4L6.txt) • IEEE Ring LANs (M5L4.txt) • High Speed LANs Token Ring Based (M5L5.txt)
Asynchronous Transfer Mode Switching ATM (M4L6.txt)	• Layered Network Architecture (M1L2.txt) • Frame Relay (M4L5.txt) • Data Link Control (M3L1.txt)	• High Speed LANs CSMA CD based (M5L6.txt) • Wireless LANs (M5L7.txt) • Internet Protocol IP (M6L2.txt)
Network Topology (M5L1.txt)	• Switching Techniques Circuit Switching (M4L1.txt) • Layered Network Architecture (M1L2.txt) • Asynchronous Transfer Mode Switching ATM (M4L6.txt)	• IEEE Ring LANs (M5L4.txt) • High Speed LANs Token Ring Based (M5L5.txt) • Satellite Networks (M5L10.txt)

Continued on next page...

Medium Access Control Techniques MAC (M5L2.txt)	• Switching Techniques Circuit Switching II (M4L2.txt) • X 25 (M4L4.txt) • Asynchronous Transfer Mode Switching ATM (M4L6.txt)	• IEEE Ring LANs (M5L4.txt) • Wireless LANs (M5L7.txt) • Congestion Control (M7L5.txt)
IEEE CSMA CD based LANs (M5L3.txt)	• Data Link Control (M3L1.txt) • Asynchronous Transfer Mode Switching ATM (M4L6.txt) • Transmission Media (M2L2.txt)	• Wireless LANs (M5L7.txt) • High Speed LANs CSMA CD based (M5L6.txt) • High Speed LANs Token Ring Based (M5L5.txt)
IEEE Ring LANs (M5L4.txt)	• Medium Access Control Techniques MAC (M5L2.txt) • Frame Relay (M4L5.txt) • Network Topology (M5L1.txt)	• High Speed LANs Token Ring Based (M5L5.txt) • Congestion Control (M7L5.txt) • Wireless LANs (M5L7.txt)
High Speed LANs Token Ring Based (M5L5.txt)	• IEEE Ring LANs (M5L4.txt) • Asynchronous Transfer Mode Switching ATM (M4L6.txt) • Frame Relay (M4L5.txt)	• High Speed LANs CSMA CD based (M5L6.txt) • Wireless LANs (M5L7.txt) • Internetworking Devices (M6L1.txt)

Continued on next page...

High Speed LANs CSMA CD based (M5L6.txt)	• High Speed LANs Token Ring Based (M5L5.txt) • IEEE CSMA CD based LANs (M5L3.txt) • Asynchronous Transfer Mode Switching ATM (M4L6.txt)	• Wireless LANs (M5L7.txt) • Internetworking Devices (M6L1.txt) • Satellite Networks (M5L10.txt)
Wireless LANs (M5L7.txt)	• Data Link Control (M3L1.txt) • Flow Control and Error Control (M3L3.txt) • Transmission Media (M2L2.txt)	• Bluetooth (M5L8.txt) • Cellular Telephone Networks (M5L9.txt) • Internetworking Devices (M6L1.txt)
Bluetooth (M5L8.txt)	• Wireless LANs (M5L7.txt) • Layered Network Architecture (M1L2.txt) • Asynchronous Transfer Mode Switching ATM (M4L6.txt)	• Satellite Networks (M5L10.txt) • Congestion Control (M7L5.txt) • Transport and Application Layer Protocols (M6L3.txt)
Cellular Telephone Networks (M5L9.txt)	• Wireless LANs (M5L7.txt) • Asynchronous Transfer Mode Switching ATM (M4L6.txt) • Bluetooth (M5L8.txt)	• Satellite Networks (M5L10.txt) • Congestion Control (M7L5.txt) • Transport and Application Layer Protocols (M6L3.txt)

Continued on next page...

Satellite Networks (M5L10.txt)	• Transmission Media (M2L2.txt) • Asynchronous Transfer Mode Switching ATM (M4L6.txt) • Wireless LANs (M5L7.txt)	• RIP Routing Information Protocol (M7L2.txt) • Internet Protocol IP (M6L2.txt) • Transport and Application Layer Protocols (M6L3.txt)
Internetworking Devices (M6L1.txt)	• High Speed LANs CSMA CD based (M5L6.txt) • Wireless LANs (M5L7.txt) • Layered Network Architecture (M1L2.txt)	• Basics of Routing (M7L1.txt) • RIP Routing Information Protocol (M7L2.txt) • Open Shortest Path First OSPF (M7L3.txt)
Internet Protocol IP (M6L2.txt)	• Layered Network Architecture (M1L2.txt) • Switching Techniques Circuit Switching II (M4L2.txt) • X 25 (M4L4.txt)	• Transport and Application Layer Protocols (M6L3.txt) • Basics of Routing (M7L1.txt) • RIP Routing Information Protocol (M7L2.txt)
Transport and Application Layer Protocols (M6L3.txt)	• Layered Network Architecture (M1L2.txt) • Internet Protocol IP (M6L2.txt) • Flow Control and Error Control (M3L3.txt)	• Congestion Control (M7L5.txt) • Secured Communication (M8L2.txt) • Firewalls (M8L3.txt)

Continued on next page...

Basics of Routing (M7L1.txt)	• Internetworking Devices (M6L1.txt) • Switching Techniques Circuit Switching II (M4L2.txt) • Internet Protocol IP (M6L2.txt)	• RIP Routing Information Protocol (M7L2.txt) • Open Shortest Path First OSPF (M7L3.txt) • Border Gateway Protocol BGP (M7L4.txt)
RIP Routing Information Protocol (M7L2.txt)	• Basics of Routing (M7L1.txt) • Internetworking Devices (M6L1.txt) • Layered Network Architecture (M1L2.txt)	• Border Gateway Protocol BGP (M7L4.txt) • Open Shortest Path First OSPF (M7L3.txt) • Congestion Control (M7L5.txt)
Open Shortest Path First OSPF (M7L3.txt)	• RIP Routing Information Protocol (M7L2.txt) • Basics of Routing (M7L1.txt) • Layered Network Architecture (M1L2.txt)	• Border Gateway Protocol BGP (M7L4.txt) • Congestion Control (M7L5.txt) • Cryptography (M8L1.txt)
Border Gateway Protocol BGP (M7L4.txt)	• RIP Routing Information Protocol (M7L2.txt) • Basics of Routing (M7L1.txt) • Open Shortest Path First OSPF (M7L3.txt)	• Congestion Control (M7L5.txt) • Secured Communication (M8L2.txt) • Cryptography (M8L1.txt)

Continued on next page...

Congestion Control (M7L5.txt)	• Basics of Routing (M7L1.txt) • RIP Routing Information Protocol (M7L2.txt) • Open Shortest Path First OSPF (M7L3.txt)	• Cryptography (M8L1.txt) • Firewalls (M8L3.txt) • Secured Communication (M8L2.txt)
Cryptography (M8L1.txt)	• Wireless LANs (M5L7.txt) • Error Detection and Correction (M3L2.txt) • Open Shortest Path First OSPF (M7L3.txt)	• Secured Communication (M8L2.txt) • Firewalls (M8L3.txt)
Secured Communication (M8L2.txt)	• Cryptography (M8L1.txt) • Border Gateway Protocol BGP (M7L4.txt) • RIP Routing Information Protocol (M7L2.txt)	• Firewalls (M8L3.txt)
Firewalls (M8L3.txt)	• Layered Network Architecture (M1L2.txt) • RIP Routing Information Protocol (M7L2.txt) • Congestion Control (M7L5.txt)	

After generating the requisites for every file, we also draw the DAG as specified in the third chapter. In the DAG, we need to ensure that there exists a link between X and Y iff X is a pre-requisite for Y and Y is a follow up of X. This way the graph will clearly depict the dependencies between the files in the course.

The graph for the same course with percentage Threshold and counts - Heuristic H3 is as shown below in Figure 5.1.

The graph for the same course with percentage Threshold and counts - Heuristic H4 is as shown below in Figure 5.2.

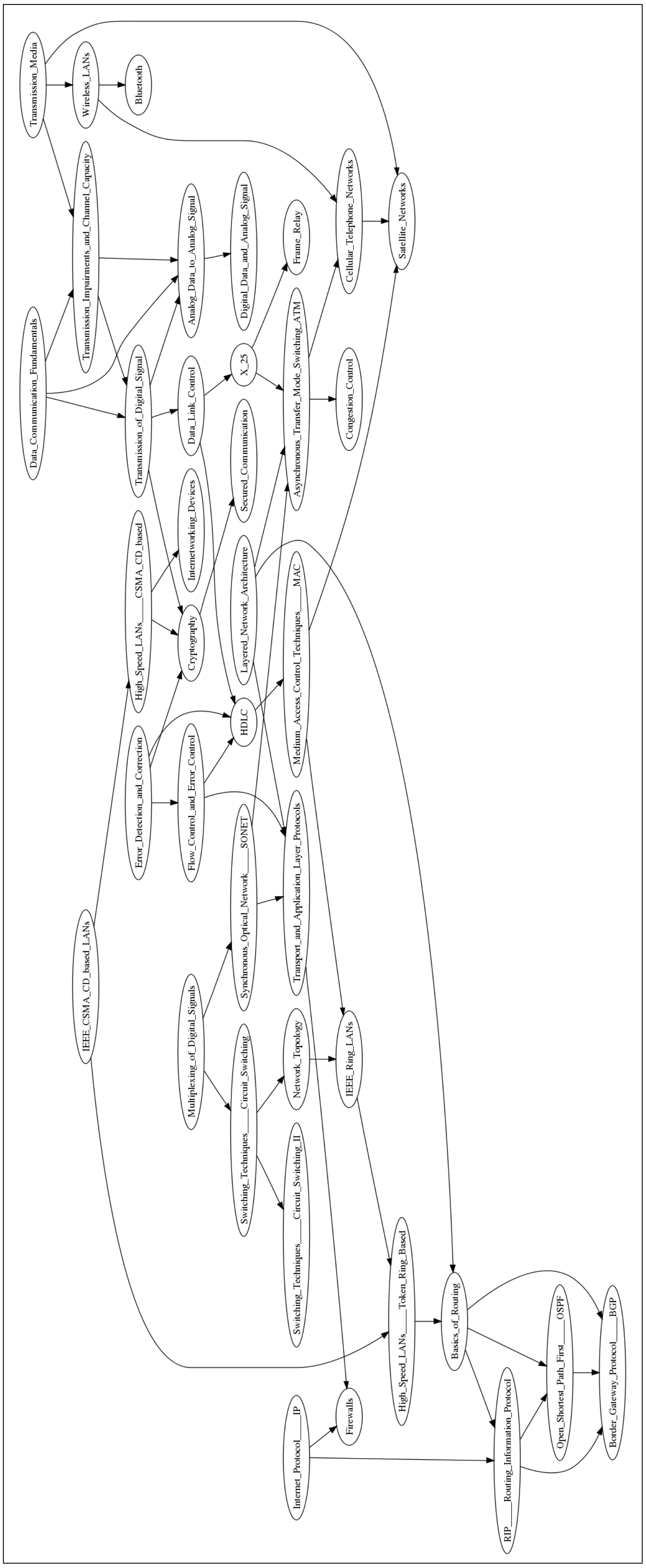

Transmission_Media
Wireless_LANs
Bluetooth
Transmission_Impairments_and_Channel_Capacity
Analog_Data_to_Analog_Signal
Digital_Data_and_Analog_Signal
Frame_Relay
Cellular_Telephone_Networks
Satellite_Networks
Data_Communication_Fundamentals
Transmission_of_Digital_Signal
Data_Link_Control
X_25
Asynchronous_Transfer_Mode_Switching_ATM
Congestion_Control
Secured_Communication
Internetworking_Devices
High_Speed_LANs____CSMA_CD_based
Cryptography
Layered_Network_Architecture
Medium_Access_Control_Techniques____MAC
HDLC
Error_Detection_and_Correction
Flow_Control_and_Error_Control
Synchronous_Optical_Network____SONET
Transport_and_Application_Layer_Protocols
IEEE_CSMA_CD_based_LANs
Multiplexing_of_Digital_Signals
Switching_Techniques____Circuit_Switching
Network_Topology
IEEE_Ring_LANs
Switching_Techniques____Circuit_Switching_II
High_Speed_LANs____Token_Ring_Based
Basics_of_Routing
Open_Shortest_Path_First____OSPF
Border_Gateway_Protocol____BGP
Internet_Protocol____IP
Firewalls
RIP____Routing_Information_Protocol

Figure 5.1: DAG for the Computer Networks course- Heuristic H3

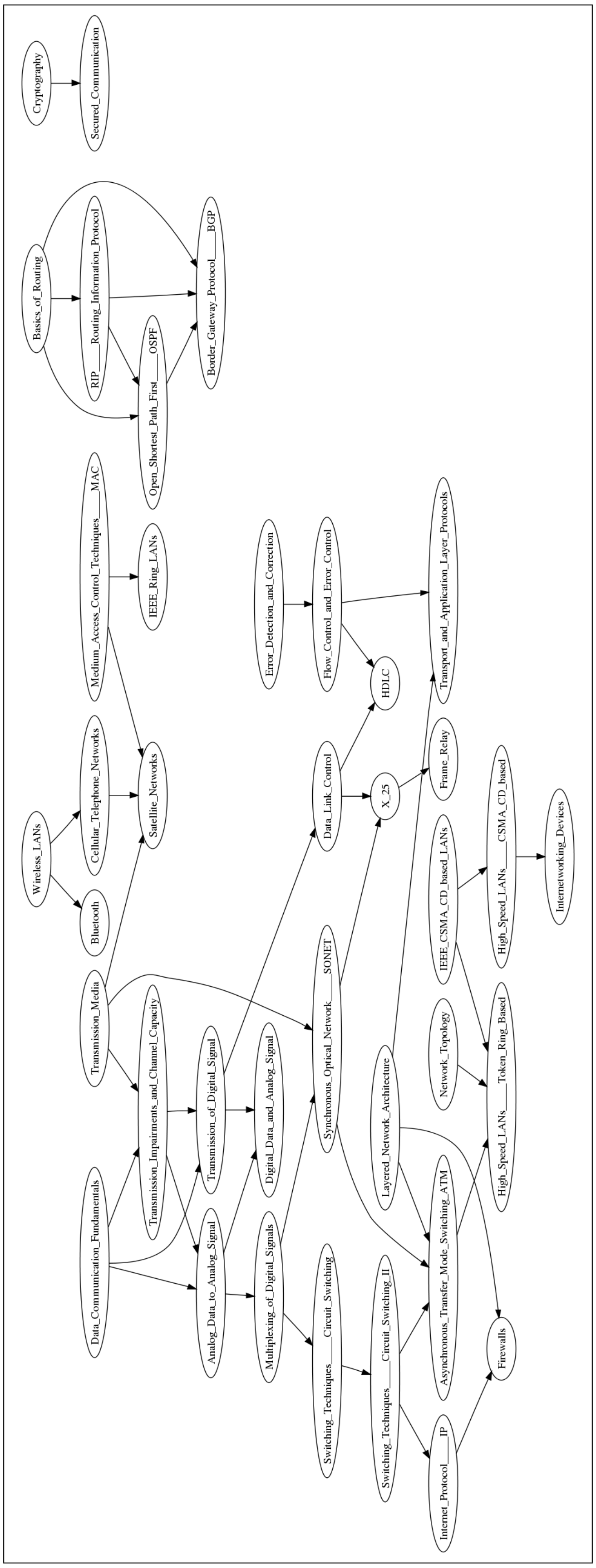

Cryptography
Secured_Communication
Basics_of_Routing
RIP____Routing_Information_Protocol
Open_Shortest_Path_First____OSPF
Border_Gateway_Protocol____BGP
Medium_Access_Control_Techniques____MAC
IEEE_Ring_LANs
Error_Detection_and_Correction
Flow_Control_and_Error_Control
HDLC
Transport_and_Application_Layer_Protocols
Wireless_LANs
Cellular_Telephone_Networks
Satellite_Networks
Bluetooth
Data_Link_Control
X_25
Frame_Relay
High_Speed_LANs____CSMA_CD_based
IEEE_CSMA_CD_based_LANs
Internetworking_Devices
Transmission_Media
Transmission_Impairments_and_Channel_Capacity
Transmission_of_Digital_Signal
Digital_Data_and_Analog_Signal
Synchronous_Optical_Network____SONET
Layered_Network_Architecture
Network_Topology
High_Speed_LANs____Token_Ring_Based
Data_Communication_Fundamentals
Analog_Data_to_Analog_Signal
Multiplexing_of_Digital_Signals
Switching_Techniques____Circuit_Switching
Switching_Techniques____Circuit_Switching_II
Asynchronous_Transfer_Mode_Switching_ATM
Firewalls
Internet_Protocol____IP

Figure 5.2: DAG for the Computer Networks course - Heuristic H4

5.2 Evaluation of the System

To know about the performance of the system, we have compared results generated by our program with those of the expert results. We created goodness metric for all of the 6 courses. We have created goodness metric separately for pre-requisites and follow-ups.

In the following description, P_i denotes the number of prerequisites suggested by expert for the i^{th} lecture. F_i denotes the number of followups suggested by expert for the i^{th} lecture. X_i denotes the number of correct prerequisites found by the program. Y_i denotes the number of correct followups found by the program. G_P denotes the goodness metric (performance of the program) for prerequisites. G_F denotes the goodness metric for followups. G denotes the overall goodness metric.

$$G_P = \sum_{i=1}^{N} \frac{P_i}{X_i} \tag{5.1}$$

$$G_F = \sum_{i=1}^{N} \frac{F_i}{Y_i} \tag{5.2}$$

$$G = \frac{G_P + G_F}{2} \tag{5.3}$$

Table 5.2: Evaluation results

Course	T0 - F0	T0 - F1	H1	H2	H3	H4
Networks	76.87	77.49	78.95	78.54	79.16	81.25
AI	60.56	69.91	73.57	72.76	73.17	80.89
SE	87.1	90.67	88.69	83.92	85.11	86.5
Embedded	81.15	77.97	85.11	76.38	78.96	81.74
OS	80.15	81.74	86	77.77	78.57	82.19
SAD	92.85	92.85	90.85	91.85	92	92.85

where T0 - F0 = Without any threshold

T0 - F1 = Without any threshold just take position

Chapter 6

Self-Assessment for the User

After we have found pre-requisites and follow-ups for each lecture of our course, we can concentrate on how to help the user to let him know how much he has learnt. In this section, we explain how the “Quiz” module proposed by the system can be used by the learner for self-evaluation. In addition, this module can also be used by the subject matter expert for getting feedback about the learning material.

6.1 Evaluation for the learner

After the learner finishes reading the recommended material for the topic, a quiz is offered related to that topic with the intention that the learner can judge whether the material was learnt well enough. When the learner takes the quiz, the scores of the learner are shown immediately along with the answers for those questions that the learner had got wrong. The learner is shown the scores of the quiz along with the average scores for that topic. The intention of this is that by looking at the absolute or average scores, the learner can get an idea of how much material was grasped. If the learner feels that the absolute score is not good enough, the learner can go through the pre-requisite material for the topic.

The quiz is objective in nature with the learner having to choose from multiple alternatives. The quiz questions, alternative answers, and the right choice are prepared by the subject matter expert per topic. These are stored in the “Quiz Question Bank” in a database.

6.2 Functioning of the System

The functioning of the system from the user perspective will be as follows:

1. User will enter the query or broadly speaking the area which he wants to learn.

2. User will get the most relevant link corresponding to the entered query.

3. User will be given the dependency graph for the course.

4. Quiz will be given to the student for self-evaluation.

5. Based on the outcomes of the results of quiz feedback will be provided to the user.

6.3 Feedback for the subject matter expert

The quiz results can also serve as a feedback for the subject matter expert as explained in this section.

Periodically, the subject matter expert can view various statistics about the topics handled by him / her. These include the following:

- Number of learners that used the search along with the keyword
- Number of learners that used the pre-requisite material, the actual topic material, and those that viewed the follow-up material
- The quiz scores per topic per learner, and the average score per topic

Based on this information, the subject matter expert can decide if the material being offered by the search engine is adequate or not. The subject matter expert can then decide to use some other material for the topic that may be better than the one currently narrowed down by the search engine If the subject matter expert find some problem related to the relevance of the topics such that he feels some topic is not given much weightage and the other is not that inmportant,then he can think about the default scoring mechanism of Lucene and decide the topics which are more relevant should be given more boost value.

Chapter 7

Conclusion and Future Work

We have tried out all the heuristics for 6 different courses. We can say that the heuristic with percentage Threshold and correlation function gives promising results as compared to the other heuristics.We have observed some conflicts in the system and the expert answers in the sense that for some files our program is generating requisites and in the expert views there are no requisites in that domain for the particular file. The things which need to be focused on for future work are: At this stage we had assumed the simple evaluation of the quiz given to the student. We can simply set some manual range to evaluate the quiz and inform the student about his/her progress. We have not taken into account the design and level of questions for the quiz. There can be many variations for the structure of the quiz such as fixing the number of questions of different difficulty levels and how to incorporate the level of difficulty with each question. We can observe the previous quizzes of the students. If a particular question is answered correctly by most of the students then that can be considered as easy and so on.

Appendix A

Lucene Scoring Formula

$$score(q,d) = coord(q,d) \,.\, queryNorm(q) \,.\, \sum_{\text{t in q}} (tf(\text{t in d}) \,.\, idf(t)^2 \,.\, t \,.\, getBoost() \,.\, norm(t,d))$$

tf(t in d) defines the number of times the term t appears in the sored document d. Documents with more occurrences of a given term have high score. The dafault value in Similarity class is:

$$\text{tf(t in d)} = \text{frequency}^{\frac{1}{2}}$$

idf(t) denotes the Inverse Document Frequency and correlates to the inverse of docFreq. The dafault computation is:

$$\text{idf}(t) = 1 + log(\frac{numDocs}{docFreq + 1})$$

coord(q,d) calculates how many of the query terms are found in the specified document. A document having more query terms will receive a higher score other than the document with fewer query terms.

queryNorm(q) is a normalizing factor which is used to make scores between queries comparable. This factor does not affect document ranking (since all ranked documents are multiplied by the same factor), but rather just attempts to make scores from different queries (or even different indexes) comparable. This is a search time factor computed by the Similarity in effect at search time. The default computation in DefaultSimilarity is:

$$queryNorm(q) = queryNorm(sumOfSquaredWeights) = \frac{1}{sumOfSquaredWeights^{\frac{1}{2}}}$$

The sum of squared weights (of the query terms) is computed by the query Weight object. For example, a boolean query computes this value as:

t.getBoost() is a search time boost of term t in the query q.

$$\text{sumOfSquaredWeights} = \text{q} \,.\, \text{getBoost()}^2 \,.\, \sum_{\text{t in q}} (\text{idf}(t) \,.\, \text{t.getBoost()})^2$$

norm(t,d) takes a few (indexing time) boost and length factors:

- Document boost - set by calling doc.setBoost() before adding the document to the index.

- Field boost - set by calling field.setBoost() before adding the field to a document.
- lengthNorm(field) - computed when the document is added to the index in accordance with the number of tokens of this field in the document, so that shorter fields contribute more to the score. LengthNorm is computed by the Similarity class in effect at indexing.

When a document is added to the index, all the above factors are multiplied. If the document has multiple fields with the same name, all their boosts are multiplied together:

$$\text{norm}(t, d) = \text{doc.getBoost()} \cdot \text{lengthNorm(field)} \cdot \prod_{\text{field f in d named as t}} \text{f.getBoost()}$$

Sample code for LUcene Indexing is shown below:

Listing A.1: Sample code Lucene

```
 * This code was originally written for
 * Erik's Lucene intro java.net article

public class Indexer {
  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      throw new Exception("Usage: java " + Indexer.class.getName()
        + " <index dir> <data dir>");
    }

    File indexDir = new File(args[0]);
    File dataDir = new File(args[1]);

    long start = new Date().getTime();
    int numIndexed = index(indexDir, dataDir);
    long end = new Date().getTime();
    System.out.println("Indexing " + numIndexed + " files took "
      + (end - start) + " milliseconds");
  }
  // open an index and start file directory traversal
  public static int index(File indexDir, File dataDir)
    throws IOException {

    if (!dataDir.exists() || !dataDir.isDirectory()) {
      throw new IOException(dataDir
        + " does not exist or is not a directory");
    }

    IndexWriter writer = new IndexWriter(indexDir,

      new StandardAnalyzer(), true);
    writer.setUseCompoundFile(false);
    indexDirectory(writer, dataDir);
    int numIndexed = writer.docCount();
    writer.optimize();
    writer.close();

    return numIndexed;
  }
  // recursive method that calls itself when it finds a directory
  private static void indexDirectory(IndexWriter writer, File dir)
    throws IOException {
    File[] files = dir.listFiles();
    for (int i = 0; i < files.length; i++) {
      File f = files[i];
      if (f.isDirectory()) {

        indexDirectory(writer, f);
      } else if (f.getName().endsWith(".txt")) {
```

```
                indexFile(writer, f);
            }
        }
    }
    // method to actually index a file using Lucene
    private static void indexFile(IndexWriter writer, File f)
        throws IOException {
        if (f.isHidden() || !f.exists() || !f.canRead()) {
            return;
        }
        System.out.println("Indexing " + f.getCanonicalPath());
        Document doc = new Document();

        doc.add(Field.Text("contents", new FileReader(f)));



        doc.add(Field.Keyword("filename", f.getCanonicalPath()));

        writer.addDocument(doc);

    }
}
```

Listing A.2: Sample code Lucene

```
 ∗ This code was originally written for
 ∗ Erik’s Lucene intro java.net article
 ∗/
public class Searcher {
  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      throw new Exception(”Usage: java ” + Searcher.class.getName()
        + ” <index dir> <query>”);
    }
         File indexDir = new File(args[0]);
         String q = args[1]; Query string
         if (!indexDir.exists() || !indexDir.isDirectory()) {
           throw new Exception(indexDir +
             ” does not exist or is not a directory.”);
         }
         search(indexDir, q);
      }
      public static void search(File indexDir, String q)
         throws Exception {
         Directory fsDir = FSDirectory.getDirectory(indexDir, false);

         IndexSearcher is = new IndexSearcher(fsDir);

         Query query = QueryParser.parse(q, ”contents”,
           new StandardAnalyzer());
         long start = new Date().getTime();

         Hits hits = is.search(query);
         long end = new Date().getTime();
         System.err.println(”Found ” + hits.length() +

           ” document(s) (in ” + (end − start) +
           ” milliseconds) that matched query ’” +
             q + ”’:”);
         for (int i = 0; i < hits.length(); i++) {

           Document doc = hits.doc(i);
           System.out.println(doc.get(”filename”));

         }
      }
    }
```

Appendix B

More Experimental Results

In the Experiments chapter, we have shown the experiments for "Computer Networks" course only. The other courses for which we have also done experiments for are "Artificial Intelligence", "Embedded Systems", "Software Engineering", "Operating System", "System Analysis and Design". The table for Embedded Systems is shown here:

Table B.1: Requisites generated by our Program

File Name	Pre-requisite Files	Follow-up Files
Introduction to Real Time Embedded Systems Part I (M1L1.txt)		• Wireless Communication (M5L27.txt) • Testing Embedded Systems (M8L38.txt) • Introduction to Real Time Systems (M6L28.txt)
Introduction to Real Time Embedded Systems Part II (M1L2.txt)		• Testing Embedded Systems (M8L38.txt) • AD and DA Converters (M3L18.txt) • Embedded Systems Components Part II (M1L4.txt)

Continued on next page...

Embedded Systems Components Part I (M1L3.txt)	• Introduction to Real Time Embedded Systems Part II (M1L2.txt)	• AD and DA Converters (M3L18.txt) • Embedded Systems Components Part II (M1L4.txt) • Embedded Processors and Memory I (M2L5.txt)
Embedded Systems Components Part II (M1L4.txt)	• Embedded Systems Components Part I (M1L3.txt) • Introduction to Real Time Embedded Systems Part II (M1L2.txt)	• AD and DA Converters (M3L18.txt) • Embedded Processors and Memory I (M2L5.txt) • Embedded Processors I (M2L10.txt)
Embedded Processors and Memory I (M2L5.txt)	• Embedded Systems Components Part II (M1L4.txt) • Embedded Systems Components Part I (M1L3.txt) • Introduction to Real Time Embedded Systems Part II (M1L2.txt)	• Embedded Processors and Memory II (M2L6.txt) • Field Programmable Gate Arrays and Applications (M4L20.txt) • General Purpose Processors I (M2L8.txt)

Continued on next page...

Embedded Processors and Memory II (M2L6.txt)	• Embedded Processors and Memory I (M2L5.txt) • Embedded Systems Components Part II (M1L4.txt) • Embedded Systems Components Part I (M1L3.txt)	• General Purpose Processors I (M2L8.txt) • Interfacing bus Protocols ISA bus etc (M3L13.txt) • Testing Embedded Systems (M8L38.txt)
Digital Signal Processors (M2L7.txt)	• Embedded Systems Components Part II (M1L4.txt) • Introduction to Real Time Embedded Systems Part II (M1L2.txt) • Embedded Processors and Memory I (M2L5.txt)	• AD and DA Converters (M3L18.txt) • Introduction to Real Time Systems (M6L28.txt) • Analog Interfacing (M3L19.txt)
General Purpose Processors I (M2L8.txt)	• Embedded Processors and Memory II (M2L6.txt) • Embedded Processors and Memory I (M2L5.txt) • Embedded Systems Components Part II (M1L4.txt)	• Embedded Processors I (M2L10.txt) • Concepts in Real Time Operating Systems (M6L31.txt) • Introduction to Hardware Description Languages I (M4L21.txt)

Continued on next page...

General Purpose Processors II (M2L9.txt)	• General Purpose Processors I (M2L8.txt) • Embedded Processors and Memory II (M2L6.txt) • Embedded Systems Components Part I (M1L3.txt)	• Interfacing bus Protocols ISA bus etc (M3L13.txt) • Concepts in Real Time Operating Systems (M6L31.txt) • Serial Data Communication (M5L25.txt)
Embedded Processors I (M2L10.txt)	• Embedded Systems Components Part II (M1L4.txt) • General Purpose Processors I (M2L8.txt) • Embedded Processors and Memory I (M2L5.txt)	• Embedded Processors II (M2L11.txt) • Serial Data Communication (M5L25.txt) • Concepts in Real Time Operating Systems (M6L31.txt)
Embedded Processors II (M2L11.txt)	• Embedded Processors I (M2L10.txt) • Embedded Systems Components Part II (M1L4.txt) • Embedded Processors and Memory I (M2L5.txt)	• Boundary Scan Methods and Standards (M8L41.txt) • Interfacing bus Protocols ISA bus etc (M3L13.txt) • Serial Data Communication (M5L25.txt)

Continued on next page...

Memory Interfacing (M2L12.txt)	• Embedded Systems Components Part II (M1L4.txt) • Embedded Processors I (M2L10.txt) • Embedded Processors and Memory I (M2L5.txt)	• Interfacing bus Protocols ISA bus etc (M3L13.txt) • Boundary Scan Methods and Standards (M8L41.txt) • AD and DA Converters (M3L18.txt)
Interfacing bus Protocols ISA bus etc (M3L13.txt)	• General Purpose Processors II (M2L9.txt) • Embedded Systems Components Part II (M1L4.txt) • Introduction to Real Time Embedded Systems Part II (M1L2.txt)	• Serial Data Communication (M5L25.txt) • DMA (M3L16.txt) • Interrupts (M3L15.txt)
Timers (M3L14.txt)	• Interfacing bus Protocols ISA bus etc (M3L13.txt) • Embedded Processors I (M2L10.txt) • Embedded Systems Components Part II (M1L4.txt)	• Interrupts (M3L15.txt) • Introduction to Hardware Description Languages II (M4L22.txt) • Introduction to Hardware Description Languages I (M4L21.txt)
Interrupts (M3L15.txt)	• Interfacing bus Protocols ISA bus etc (M3L13.txt) • Timers (M3L14.txt) • Embedded Processors I (M2L10.txt)	• DMA (M3L16.txt) • Concepts in Real Time Operating Systems (M6L31.txt) • Commercial Real Time Operating Systems (M6L32.txt)

Continued on next page...

DMA (M3L16.txt)	• Interfacing bus Protocols ISA bus etc (M3L13.txt) • Interrupts (M3L15.txt) • General Purpose Processors I (M2L8.txt)	• Analog Interfacing (M3L19.txt) • Software Design Part 2 (M7L37.txt) • Serial Data Communication (M5L25.txt)
USB and IrDA (M3L17.txt)	• Interfacing bus Protocols ISA bus etc (M3L13.txt) • Interrupts (M3L15.txt) • Embedded Systems Components Part I (M1L3.txt)	• Wireless Communication (M5L27.txt) • Serial Data Communication (M5L25.txt) • Introduction to Real Time Systems (M6L28.txt)
AD and DA Converters (M3L18.txt)	• Digital Signal Processors (M2L7.txt) • Embedded Systems Components Part II (M1L4.txt) • Introduction to Real Time Embedded Systems Part II (M1L2.txt)	• Analog Interfacing (M3L19.txt) • Introduction to Real Time Systems (M6L28.txt) • Introduction to Hardware Description Languages I (M4L21.txt)
Analog Interfacing (M3L19.txt)	• AD and DA Converters (M3L18.txt) • Interfacing bus Protocols ISA bus etc (M3L13.txt) • DMA (M3L16.txt)	• Introduction to Real Time Systems (M6L28.txt) • Introduction to Hardware Description Languages I (M4L21.txt) • Design for Testability (M8L39.txt)

Continued on next page...

Field Programmable Gate Arrays and Applications (M4L20.txt)	• Embedded Processors and Memory I (M2L5.txt) • Embedded Systems Components Part II (M1L4.txt) • General Purpose Processors I (M2L8.txt)	• Introduction to Hardware Description Languages I (M4L21.txt) • Design for Testability (M8L39.txt) • Introduction to Hardware Description Languages III (M4L23.txt)
Introduction to Hardware Description Languages I (M4L21.txt)	• Field Programmable Gate Arrays and Applications (M4L20.txt) • AD and DA Converters (M3L18.txt) • Timers (M3L14.txt)	• Introduction to Hardware Description Languages III (M4L23.txt) • Introduction to Hardware Description Languages II (M4L22.txt) • Modelling Timing Constraints (M7L35.txt)
Introduction to Hardware Description Languages II (M4L22.txt)	• Timers (M3L14.txt) • Introduction to Hardware Description Languages I (M4L21.txt) • Interfacing bus Protocols ISA bus etc (M3L13.txt)	• Introduction to Hardware Description Languages III (M4L23.txt) • Software Design Part 2 (M7L37.txt) • Design for Testability (M8L39.txt)

Continued on next page...

Introduction to Hardware Description Languages III (M4L23.txt)	• Introduction to Hardware Description Languages I (M4L21.txt) • Field Programmable Gate Arrays and Applications (M4L20.txt) • Introduction to Hardware Description Languages II (M4L22.txt)	• Modelling Timing Constraints (M7L35.txt) • Built In Self Test BIST for Embedded Systems (M8L40.txt) • Introduction to Software Engineering (M7L33.txt)
Parallel Data Communication (M5L24.txt)	• Interfacing bus Protocols ISA bus etc (M3L13.txt) • DMA (M3L16.txt) • Interrupts (M3L15.txt)	• Introduction to Real Time Systems (M6L28.txt) • Serial Data Communication (M5L25.txt) • Commercial Real Time Operating Systems (M6L32.txt)
Serial Data Communication (M5L25.txt)	• Interfacing bus Protocols ISA bus etc (M3L13.txt) • USB and IrDA (M3L17.txt) • Embedded Processors I (M2L10.txt)	• Wireless Communication (M5L27.txt) • Introduction to Real Time Systems (M6L28.txt) • Network Communication (M5L26.txt)

Continued on next page...

Network Communication (M5L26.txt)	• Serial Data Communication (M5L25.txt) • Interfacing bus Protocols ISA bus etc (M3L13.txt) • DMA (M3L16.txt)	• Requirements Analysis and Specification (M7L34.txt) • Introduction to Real Time Systems (M6L28.txt) • Modelling Timing Constraints (M7L35.txt)
Wireless Communication (M5L27.txt)	• Serial Data Communication (M5L25.txt) • USB and IrDA (M3L17.txt) • Interfacing bus Protocols ISA bus etc (M3L13.txt)	• Introduction to Real Time Systems (M6L28.txt) • Real Time Task Scheduling Part 2 (M6L30.txt) • Software Design Part 2 (M7L37.txt)
Introduction to Real Time Systems (M6L28.txt)	• AD and DA Converters (M3L18.txt) • Serial Data Communication (M5L25.txt) • Field Programmable Gate Arrays and Applications (M4L20.txt)	• Modelling Timing Constraints (M7L35.txt) • Testing Embedded Systems (M8L38.txt) • On line Testing of Embedded Systems (M8L42.txt)

Continued on next page...

Real Time Task Scheduling Part 1 (M6L29.txt)	• Introduction to Real Time Systems (M6L28.txt) • Embedded Systems Components Part II (M1L4.txt) • Introduction to Real Time Embedded Systems Part II (M1L2.txt)	• Real Time Task Scheduling Part 2 (M6L30.txt) • Concepts in Real Time Operating Systems (M6L31.txt) • Testing Embedded Systems (M8L38.txt)
Real Time Task Scheduling Part 2 (M6L30.txt)	• Real Time Task Scheduling Part 1 (M6L29.txt) • Wireless Communication (M5L27.txt) • Embedded Systems Components Part II (M1L4.txt)	• Concepts in Real Time Operating Systems (M6L31.txt) • Commercial Real Time Operating Systems (M6L32.txt) • Built In Self Test BIST for Embedded Systems (M8L40.txt)
Concepts in Real Time Operating Systems (M6L31.txt)	• Real Time Task Scheduling Part 2 (M6L30.txt) • Real Time Task Scheduling Part 1 (M6L29.txt) • Introduction to Real Time Systems (M6L28.txt)	• Commercial Real Time Operating Systems (M6L32.txt) • Software Design Part 2 (M7L37.txt) • Built In Self Test BIST for Embedded Systems (M8L40.txt)

Continued on next page...

Commercial Real Time Operating Systems (M6L32.txt)	• Concepts in Real Time Operating Systems (M6L31.txt) • Introduction to Real Time Systems (M6L28.txt) • Real Time Task Scheduling Part 2 (M6L30.txt)	• Software Design Part 2 (M7L37.txt) • Testing Embedded Systems (M8L38.txt) • Design for Testability (M8L39.txt)
Introduction to Software Engineering (M7L33.txt)	• Introduction to Real Time Systems (M6L28.txt) • Introduction to Hardware Description Languages I (M4L21.txt) • Introduction to Hardware Description Languages III (M4L23.txt)	• Software Design Part 1 (M7L36.txt) • Requirements Analysis and Specification (M7L34.txt) • Software Design Part 2 (M7L37.txt)
Requirements Analysis and Specification (M7L34.txt)	• Introduction to Software Engineering (M7L33.txt) • Introduction to Real Time Systems (M6L28.txt) • Introduction to Hardware Description Languages I (M4L21.txt)	• Software Design Part 1 (M7L36.txt) • Software Design Part 2 (M7L37.txt) • Testing Embedded Systems (M8L38.txt)

Continued on next page...

Modelling Timing Constraints (M7L35.txt)	• Introduction to Real Time Systems (M6L28.txt) • Introduction to Hardware Description Languages I (M4L21.txt) • Introduction to Hardware Description Languages III (M4L23.txt)	• Testing Embedded Systems (M8L38.txt) • Software Design Part 2 (M7L37.txt) • Design for Testability (M8L39.txt)
Software Design Part 1 (M7L36.txt)	• Introduction to Software Engineering (M7L33.txt) • Requirements Analysis and Specification (M7L34.txt) • Introduction to Real Time Systems (M6L28.txt)	• Software Design Part 2 (M7L37.txt) • Testing Embedded Systems (M8L38.txt) • Design for Testability (M8L39.txt)
Software Design Part 2 (M7L37.txt)	• Software Design Part 1 (M7L36.txt) • Introduction to Software Engineering (M7L33.txt) • Concepts in Real Time Operating Systems (M6L31.txt)	• Boundary Scan Methods and Standards (M8L41.txt) • Testing Embedded Systems (M8L38.txt) • Design for Testability (M8L39.txt)

Continued on next page...

Testing Embedded Systems (M8L38.txt)	• Introduction to Real Time Systems (M6L28.txt) • Introduction to Software Engineering (M7L33.txt) • Introduction to Real Time Embedded Systems Part II (M1L2.txt)	• Built In Self Test BIST for Embedded Systems (M8L40.txt) • On line Testing of Embedded Systems (M8L42.txt) • Design for Testability (M8L39.txt)
Design for Testability (M8L39.txt)	• Field Programmable Gate Arrays and Applications (M4L20.txt) • Introduction to Hardware Description Languages I (M4L21.txt) • Testing Embedded Systems (M8L38.txt)	• Boundary Scan Methods and Standards (M8L41.txt) • Built In Self Test BIST for Embedded Systems (M8L40.txt) • On line Testing of Embedded Systems (M8L42.txt)
Built In Self Test BIST for Embedded Systems (M8L40.txt)	• Testing Embedded Systems (M8L38.txt) • Design for Testability (M8L39.txt) • Embedded Systems Components Part II (M1L4.txt)	• On line Testing of Embedded Systems (M8L42.txt) • Boundary Scan Methods and Standards (M8L41.txt)

Continued on next page...

Boundary Scan Methods and Standards (M8L41.txt)	• Software Design Part 2 (M7L37.txt) • Design for Testability (M8L39.txt) • Built In Self Test BIST for Embedded Systems (M8L40.txt)	• On line Testing of Embedded Systems (M8L42.txt)
On line Testing of Embedded Systems (M8L42.txt)	• Built In Self Test BIST for Embedded Systems (M8L40.txt) • Introduction to Real Time Systems (M6L28.txt) • Testing Embedded Systems (M8L38.txt)	

DAG for the course wth Heuristic percentage Threshold and correlation H4 is as shown below in Figure B.1.

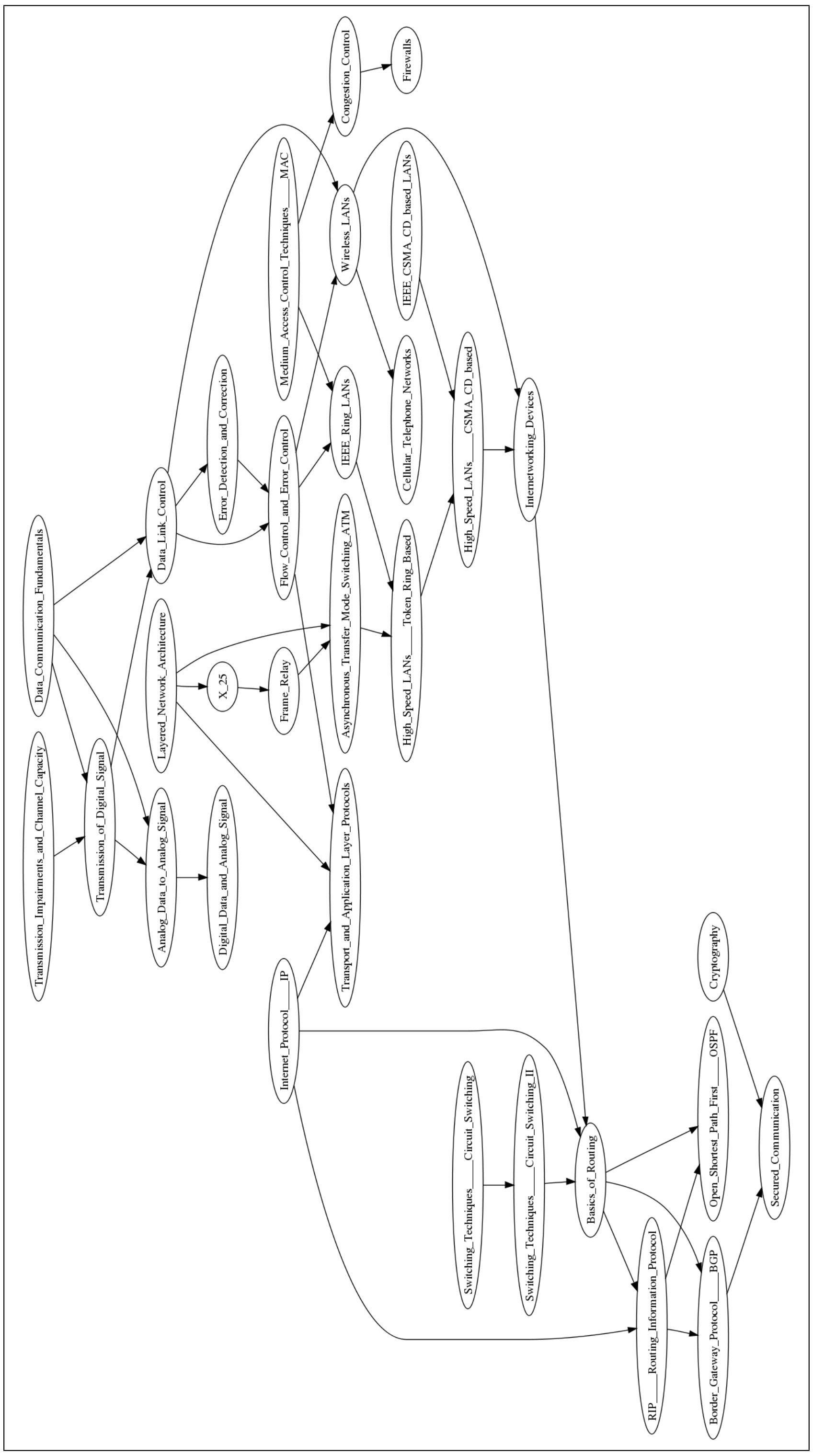

Transmission_Impairments_and_Channel_Capacity
Data_Communication_Fundamentals
Transmission_of_Digital_Signal
Data_Link_Control
Layered_Network_Architecture
Analog_Data_to_Analog_Signal
Digital_Data_and_Analog_Signal
Error_Detection_and_Correction
X_25
Frame_Relay
Flow_Control_and_Error_Control
Medium_Access_Control_Techniques____MAC
Internet_Protocol____IP
Transport_and_Application_Layer_Protocols
Asynchronous_Transfer_Mode_Switching_ATM
IEEE_Ring_LANs
Congestion_Control
Wireless_LANs
Firewalls
Cellular_Telephone_Networks
IEEE_CSMA_CD_based_LANs
High_Speed_LANs____Token_Ring_Based
High_Speed_LANs____CSMA_CD_based
Internetworking_Devices
Switching_Techniques____Circuit_Switching
Switching_Techniques____Circuit_Switching_II
Basics_of_Routing
RIP____Routing_Information_Protocol
Border_Gateway_Protocol____BGP
Open_Shortest_Path_First____OSPF
Cryptography
Secured_Communication

Figure B.1: DAG for the Embedded Systems course - Heuristic H4

DAG for the course Artificial Intelligence wth Heuristic percentage Threshold and correlation H4 is as shown below in FigureB.2

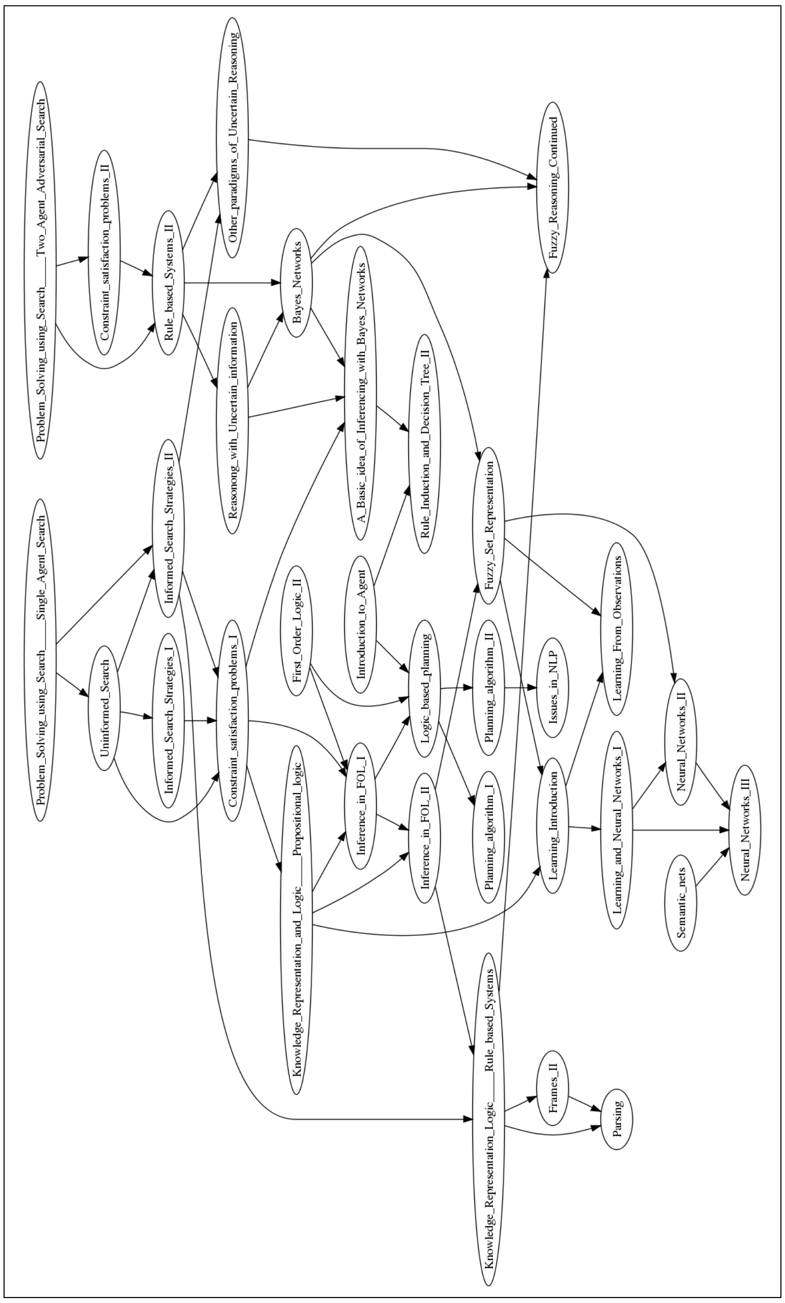

Problem_Solving_using_Search___Two_Agent_Adversarial_Search
Constraint_satisfaction_problems_II
Rule_based_Systems_II
Other_paradigms_of_Uncertain_Reasoning
Bayes_Networks
Reasonong_with_Uncertain_information
A_Basic_idea_of_Inferencing_with_Bayes_Networks
Rule_Induction_and_Decision_Tree_II
Fuzzy_Reasoning_Continued
Problem_Solving_using_Search___Single_Agent_Search
Informed_Search_Strategies_II
Uninformed_Search
Informed_Search_Strategies_I
Constraint_satisfaction_problems_I
First_Order_Logic_II
Introduction_to_Agent
Logic_based_planning
Fuzzy_Set_Representation
Planning_algorithm_II
Issues_in_NLP
Learning_From_Observations
Knowledge_Representation_and_Logic___Propositional_logic
Inference_in_FOL_I
Inference_in_FOL_II
Planning_algorithm_I
Learning_Introduction
Learning_and_Neural_Networks_I
Neural_Networks_II
Semantic_nets
Neural_Networks_III
Knowledge_Representation_Logic___Rule_based_Systems
Frames_II
Parsing

Figure B.2: DAG for the Artificial Intelligence course - Heuristic H4

DAG for the course Software Engineering wth Heuristic percentage Threshold and correlation H4 is as shown below in FigureB.3

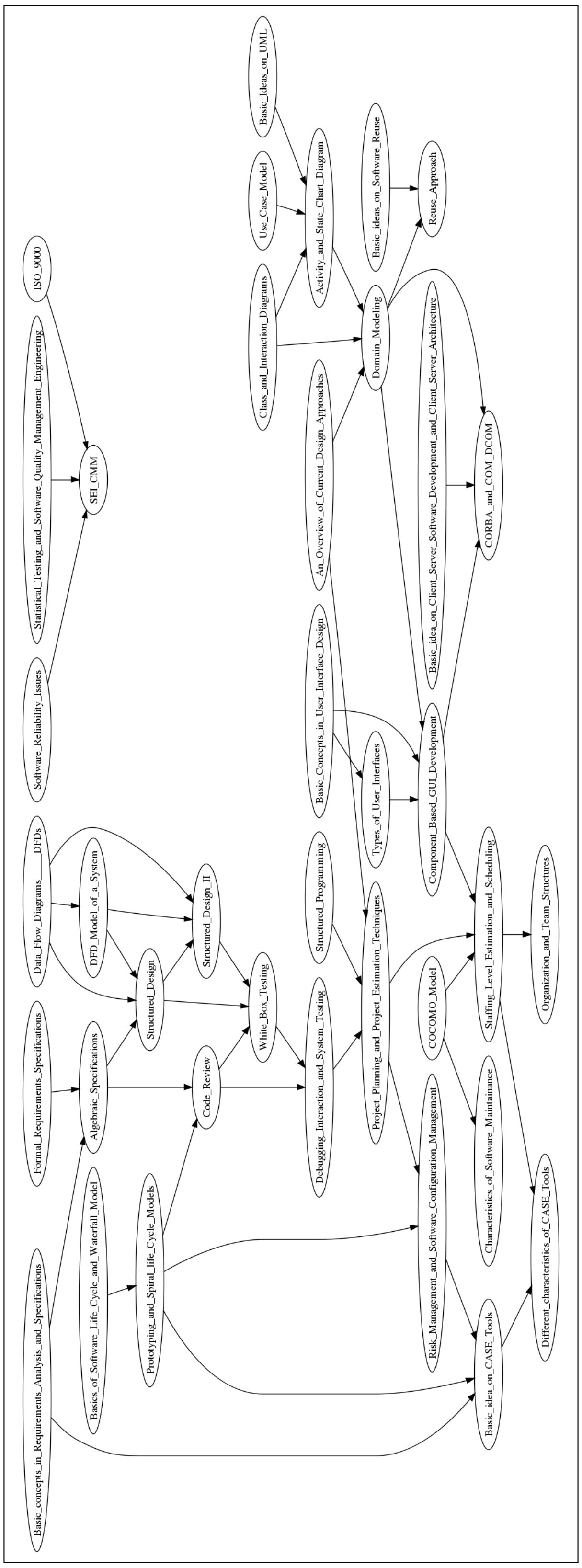

ISO_9000
Statistical_Testing_and_Software_Quality_Management_Engineering
SEI_CMM
Software_Reliability_Issues
Data_Flow_Diagrams___DFDs
DFD_Model_of_a_System
Structured_Design_II
Structured_Design
Formal_Requirements_Specifications
Algebraic_Specifications
Code_Review
White_Box_Testing
Basics_of_Software_Life_Cycle_and_Waterfall_Model
Prototyping_and_Spiral_life_Cycle_Models
Basic_concepts_in_Requirements_Analysis_and_Specifications
Basic_Ideas_on_UML
Use_Case_Model
Activity_and_State_Chart_Diagram
Class_and_Interaction_Diagrams
An_Overview_of_Current_Design_Approaches
Domain_Modeling
Basic_ideas_on_Software_Reuse
Reuse_Approach
Basic_idea_on_Client_Server_Software_Development_and_Client_Server_Architecture
CORBA_and_COM_DCOM
Basic_Concepts_in_User_Interface_Design
Types_of_User_Interfaces
Component_Based_GUI_Development
Structured_Programming
Debugging_Interaction_and_System_Testing
Project_Planning_and_Project_Estimation_Techniques
COCOMO_Model
Staffing_Level_Estimation_and_Scheduling
Organization_and_Team_Structures
Characteristics_of_Software_Maintainance
Risk_Management_and_Software_Configuration_Management
Basic_idea_on_CASE_Tools
Different_characteristics_of_CASE_Tools

Figure B.3: DAG for the Software Engineering course - Heuristic H4

DAG for the course Operating System wth Heuristic percentage Threshold and correlation H4 is as shown below in FigureB.4

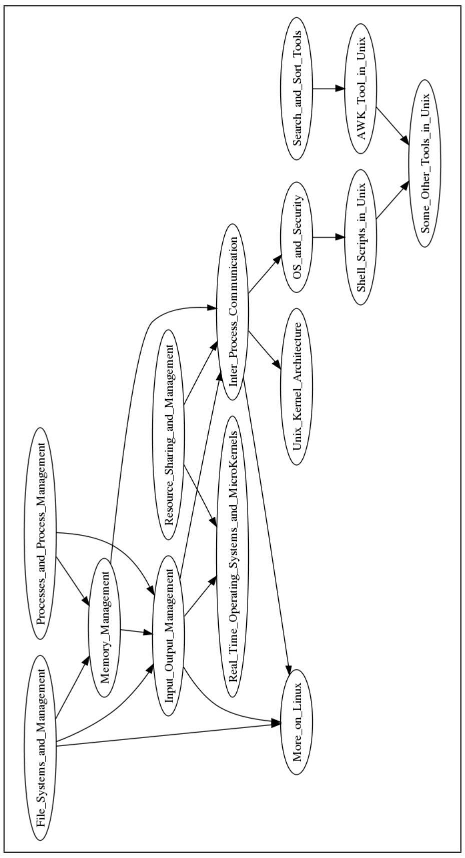


Figure B.4: DAG for the Operating Systems course - Heuristic H4

DAG for the course Systems Analysis and Design wth Heuristic percentage Threshold and correlation H4 is as shown below in FigureB.5

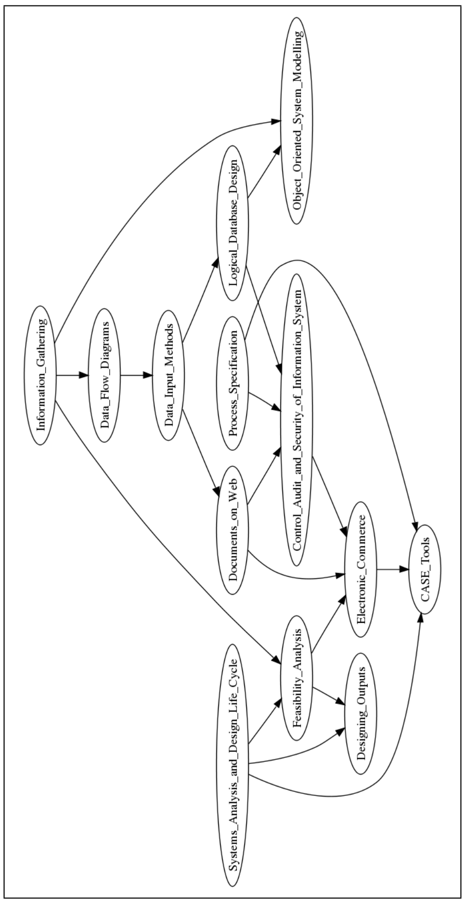


Figure B.5: DAG for the Systems Analysis and Design course - Heuristic H4

References

[ATu] [online, cited 02-07-2008]Available from World Wide Web: `http://www.atutor.ca/atutor/index.php`.

[CDE] CDEEP - centre for distance engineering education program [online, cited 02-07-2008]. Available from World Wide Web: `http://www.cdeep.iitb.ac.in`.

[DOT] [online, cited 02-07-2008]Available from World Wide Web: `http://en.wikipedia.org/wiki/DOT_language`.

[GC05] Weimin Ge and Yuefeng Chao. Implementation of e-learning system for unu-iist. 2005.

[HG04] Erik Hatcher and Otis Gospodnetic. *Lucene in Action (In Action series)*. Manning Publications Co., Greenwich, CT, USA, 2004.

[Kha05] Khan. Managing e-learning: Design, delivery, implementation and evaluation. 2005.

[Lea] [online, cited 02-07-2008]Available from World Wide Web: `http://en.wikipedia.org/wiki/Learning_management_system`.

[Luc] [online, cited 02-07-2008]Available from World Wide Web: `http://en.wikipedia.org/wiki/Lucene`.

[Lucene] [online, cited 02-07-2008]Available from World Wide Web: `http://www.onjava.com/pub/a/onjava/2003/01/15/lucene.html`.

[Luk] [online, cited 02-07-2008]Available from World Wide Web: `http://www.getopt.org/luke/`.

[MIT] MIT open courseware [online, cited 02-07-2008]. Available from World Wide Web: `http://ocw.mit.edu`.

[NPT] National programme on technology enhanced learning [online, cited 02-07-2008]. Available from World Wide Web: `http://www.nptel.iitm.ac.in`.

[OLA] [online, cited 02-07-2008]Available from World Wide Web: `http://en.wikipedia.org/wiki/OLAT`.

[sea] [online, cited 02-07-2008]Available from World Wide Web: `http://en.wikipedia.org/wiki/List_of_search_engines`.

[Sie04] Siemens. Categories of e-learning. 2004.

[Sta] [online, cited 02-07-2008]Available from World Wide Web: http://scil.stanford.edu/news/eLearning.html.

[Uta] [online, cited 02-07-2008]Available from World Wide Web: http://www.hewlett.org/Programs/Education/OER/OpenContent/Utah+State+University.htm.