

**Intinno: A Web Integrated Digital Library and Learning Content  
Management System**

Thesis to be submitted in Partial Fulfillment of the  
Requirements for the Award of the Degree of

**Master of Technology**  
**In**  
**Computer Science and Engineering**



*Submitted By:*

**Arpit Jain (03CS3009)**

*Under the supervision of*

**Prof. Pabitra Mitra**

Department of Computer Science and Engineering

Indian Institute of Technology Kharagpur

May 2008

# Certificate

This is to certify that the thesis titled *Intinno: A Web Integrated Digital Library and Learning Content Management System*, submitted by **Arpit Jain**, to the Department of Computer Science and Engineering, in partial fulfillment for the award of the degree of **Master of Technology** is a bonafide record of work carried out by him under our supervision and guidance. The thesis has fulfilled all the requirements as per the regulations of the institute and, in our opinion, has reached the standard needed for submission.

**Prof. Pabitra Mitra**

Dept. of Computer Science and Engineering  
Indian Institute of Technology  
Kharagpur 721302, INDIA

**Arpit Jain**

Dept. of Computer Science and Engineering  
Indian Institute of Technology  
Kharagpur 721302, INDIA

## **Abstract**

*The work describes the design of Intinno, an intelligent web based learning content management system. The system aims to circumvent certain drawbacks of existing learning management systems in terms of scarcity of content, lack of intelligent search and context sensitive personalization.*

*The scarcity problem is solved by using web mining to crawl learning content from the web. Web mining is done by using a focused crawler that is trained to mine only educational content from the web. The mined content is then automatically archived in SCORM format for further reuse and exchange.*

*The archived content used to develop concept maps that capture the various semantic relations among data. Automatic annotation using the concept maps is used to archive the crawled content into a digital library. Multi-parameter indexing and clustering is done to provide intelligent content based search.*

*The semantic content from the digital library is used to develop intelligent learning applications. These applications focus on making the learning process for a student both efficient and effective. Algorithms for learning applications like generation of memory maps and table of contents for a specific topic are proposed. Context sensitive and personalized recommendation on content is supported. The initial version of the system is available online at <http://www.intinno.com>*

# Table of Contents

Abstract.....	3
Chapter 1: Introduction.....	5
1.1    Learning Management Systems.....	6
1.1.1    Introduction.....	6
1.1.2    Problems faced by LCMS.....	7
1.1.3    Advantages of LCMS.....	8
1.1.4    LCMS + Web 2.0 = E-Learning2.0.....	9
1.2    Intelligent tutoring Systems (ITS).....	10
1.2.1    Introduction.....	10
1.2.2    Disadvantages of ITS.....	12
1.2.3    Advantages of ITS.....	12
1.3    Motivation.....	12
1.4    Proposed Approach.....	13
1.4.1    Introduction.....	13
1.4.2    Useful Applications.....	14
Chapter 2: Architecture of Intinno.....	17
2.1    Introduction.....	17
2.2    Learning Resources on Web.....	18
2.2.1    Open source and reviewed course pages.....	20
2.2.2    Content from University course pages.....	20
2.2.3    Unstructured data: Discussion forums and animation videos.....	22
2.2.4    General purpose collections like Wikipedia.....	23
2.2.5    Websites of industrial corporations.....	24
2.3    Mining Resources from the web.....	24
2.3.1    Why a focused Crawler?.....	24
2.3.2    Content from University Course pages.....	25
2.3.3    Possible approaches of mining course content from university homepages.....	26

2.3.4	Our Approach.....	28
2.4	Domain Knowledge Representation.....	29
2.4.1	Introduction.....	30
2.4.2	Why Knowledge Representation ?.....	31
2.4.3	An Approach.....	31
2.4.4	Our Approach – The Motivation.....	33
2.1.5	Our Design.....	34
Chapter 3:	Intelligent Applications for Intinno.....	36
3.1	Introduction .....	36
3.2	Memory Maps.....	38
3.2.1	Need for Memory Maps.....	38
3.2.2	Approach 1.....	40
3.2.3	Approach 2.....	41
Chapter 4:	Implementation and Results.....	45
4.1	Implementation of CMS.....	45
References	.....	48

## **Chapter 1: Introduction**

## **1.1 Learning Management Systems**

### **1.1.1 Introduction**

A Learning Management System (or LMS) is a software tool designed to manage user learning processes . LMSs go far beyond conventional training records management and reporting. The value-add for LMSs is the extensive range of complementary functionality they offer. Learner self-service (e.g. self-registration on instructor-led training), learning workflow (e.g. user notification, teacher approval, waitlist management), the provision of on-line learning, on-line assessment, management of continuous professional education, collaborative learning (e.g. application sharing, discussion threads), and training resource management (e.g. instructors, facilities, equipment), are some of the additional dimensions to leading learning management systems [1].

In addition to managing the administrative functions of online learning, some systems also provide tools to deliver and manage instructor-led synchronous and asynchronous online teaching based on learning object methodology. These systems are called Learning content management systems or LCMSs. An LCMS provides tools for authoring and re-using or re-purposing content as well as virtual spaces for learner interaction (such as discussion forums and live chat rooms). The focus of an LCMS is on learning content. It gives authors, instructional designers, and subject matter experts the means to create and re-use e-learning content more efficiently [2].

LCMSs provide instructors with the ability to perform the following tasks [3]:

- Place course materials online. Most CMSs provide pre-programmed buttons for the course syllabus, course schedule, and course materials linked to specific lessons, such as copies of readings and PowerPoint slides from lectures.

- Track student progress through assessment features, which enable instructors to give quizzes and tests online, and an online gradebook, where instructors can post student grades.
- Discussion board, where instructors and students can discuss readings and continue class discussions between formal class sessions.
- Other communications tools, which let instructors send announcements to classes and communicate individually with students.
- Lock box for students, where students can store class materials in a safe place—either a presentation to give later in class or backing up class assignments in a safe place.
- Course statistics, which provide information on the use of the course site, including who used the course site and when.

LCMSs also have proven popular in managing asynchronous academic distance courses, too, because of their ability to manage discussions. In addition, given that LCMSs were already installed and in wide use only adds to their popularity. When using a LCMS to manage a distance course, instructors post a core lessons master script, of sorts, that guides students through readings, discussions, and learning activities instead of merely posting readings and PowerPoint slides for each lesson,. Instructors then use the discussion board to manage the course discussions, which are usually more extensive than those used in classroom courses.

### **1.1.2 Problems faced by LCMS**

The current course management systems have a number of drawbacks which hinder their wide acceptance among teachers and students.

- One of them being the problem of cold start. Instructors who begin to make up a course don't have the material to start up.

- Seamless content reuse is often not possible.
- Materials presented may lack coverage of the subject area and thus fail to cater information needs of all students in a class.
- Students while studying or reading a lecture have to waste a lot of their time in searching for relevant resources from the web.

### **1.1.3 Advantages of LCMS**

The current course management systems have a number of advantages:

- LCMSs enable instructors to easily create a course website by following a template and uploading existing documents in PowerPoint, Word, Excel, Acrobat and other popular formats without converting them to a web format (like HTML), they require few specialized skills.
- LCMSs are easy to learn and were quickly adopted by instructors, even those who might claim to be luddites.
- LCMSs allow active participation of students in learning activities even outside the physical boundaries of a classroom.
- LCMSs help the instructors to create and archive of the course material and discussions which would be helpful to the students opting the course next year.



#### 1.1.4 LCMS + Web 2.0 = E-Learning2.0

The changes in e-learning are being driven by two primary forces [4]. The first force is a steady increase in the pace information creation, boosted by the availability of easy to use LCMS. This has led to a shift in work, especially knowledge work, and an evolution in information needs. The second driver affecting workplace learning is the advent of Web 2.0. In its most basic form, Web 2.0 means that anyone should be able to easily create and contribute content on the Internet. This ranges from writing a blog, to providing video on YouTube, to putting pictures on Flickr, to contributing written content on wikis such as Wikipedia, as well as developing a social network on something like MySpace. The key components to Web 2.0 are the ease of using the tools and the collaboration/social interaction that naturally results. One of the interesting results from Web 2.0 is something called collective intelligence. For example, consider how Amazon's user ratings and comments influence buyer behavior.

The term *E-Learning 2.0* was coined by Stephen Downes, a Canadian researcher, and it derives from the overall e-learning trends stated above in combination with Web 2.0. To begin to examine E-Learning 2.0, let's consider an example:

A small team of five practitioners in a corporate learning department has adopted e-learning 2.0 tools as part of their daily work. They need to define their strategy around the use of "rapid e-learning" and present it to management as part of the annual budget process. Here are some of the ways the workgroup will take advantage of E-Learning 2.0 tools:

- Search for useful web pages, then tag, add comments, and share them by using such social bookmarking tools as del.icio.us or Yahoo MyWeb. By using these tools, the team will keep a copy of each page; the page is full-text searchable; it can be accessed from any computer; and everyone on the team has access to the same links.
- Create public blog posts (using a tool like Blogger) that will outline the team's current thinking about how rapid e-learning fits into its future strategic plans. The blog also will solicit feedback from everyone on the team, as well as the larger e-learning blog community.

- Write or copy-and-paste notes into a wiki, which will become a shared resource that everyone on the team can edit.
- Use an RSS reader (for example, Bloglines) to track updates to the wiki, social bookmarking tools, and the blog. This eliminates the need for email as the reader becomes the single place each team member visits to see whatâ€™s happened recently.

E-Learning 2.0 is making an impact in formal learning settings, and they are particularly useful for collaborative formal learning. For example, wikis can be used as part of group projects; blogs can be used to submit written work and offer the opportunity for peers to provide feedback in a collaborative learning setting; and social bookmarking tools can be used as part of collaborative research. Again, the ease-of-use and collaborative nature of these tools make them a natural fit for learning.

## **1.2 Intelligent tutoring Systems (ITS)**

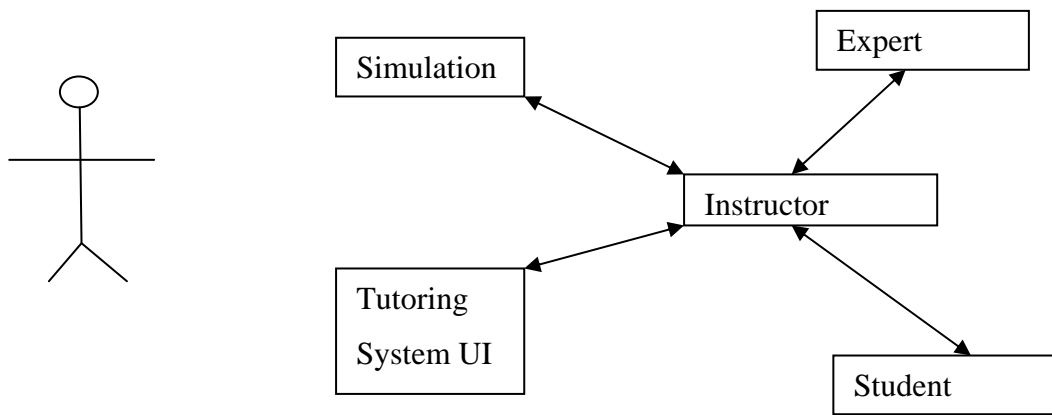
### **1.2.1 Introduction**

Imagine that each learner in a classroom has a personal training assistant who pays attention to the participant's learning needs, assesses and diagnoses problems, and provides assistance as needed. The assistant could perform many of the routine instructional interventions and alert the instructor of learning problems that are too difficult for it. By taking on basic assistance tasks, the instructor would be free to concentrate on training issues that require greater expertise.

Providing a personal training assistant for each learner is beyond the training budgets of most organizations. However, a *virtual* training assistant that captures the subject matter and teaching expertise of experienced trainers provides a captivating new option. The concept, known as intelligent tutoring systems (ITS) [5] or intelligent computer-aided instruction (ICAI), has been pursued for more than three decades by researchers in education, psychology, and artificial intelligence. Today, prototype and operational ITS systems provide practice-based instruction to support corporate training, schools and college education, and military training.

The goal of ITS is to provide the benefits of one-on-one instruction automatically and cost effectively. Like training simulations, ITS enables participants to practice their skills by carrying out tasks within highly interactive learning environments. However, ITS goes beyond training simulations by answering user questions and providing individualized guidance. Unlike other computer-based training technologies, ITS systems assess each learner's actions within these interactive environments and develop a model of their knowledge, skills, and expertise. Based on the learner model, ITSs tailor instructional strategies, in terms of both the content and style, and provide explanations, hints, examples, demonstrations, and practice problems as needed.

ITS systems typically rely on three types of knowledge, organized into separate software modules (as shown in Figure 1). The "expert model" represents subject matter expertise and provides the ITS with knowledge of what it's teaching. The "student model" represents what the user does and doesn't know, and what he or she does and doesn't have. This knowledge lets the ITS know who it's teaching. The "instructor model" enables the ITS to know how to teach, by encoding instructional strategies used via the tutoring system user interface.



*Figure 1: Components of an intelligent Tutoring System*

### **1.2.2 Disadvantages of ITS**

ITS needs careful preparation in terms of describing the knowledge and possible behaviors of experts, students and tutors. This description needs to be done in a formal language in order that the ITS may process the information and draw inferences in order to generate feedback or instruction. Therefore a mere description is not enough, the knowledge contained in the models should be organized and linked to an inference engine. It is through the latter's interaction with the descriptive data that tutorial feedback is generated. All this is a substantial amount of work. This means that building an ITS is an option only in situations in which they, in spite of their relatively high development costs, still reduce the overall costs through reducing the need for human instructors or sufficiently boosting overall productivity. Such situations occur when large groups need to be tutored simultaneously or many replicated tutoring efforts are needed. Cases in point are technical training situations such as training of military recruits and high school mathematics.

### **1.2.3 Advantages of ITS**

An ITS system has the following advantages:

- They provide the benefits of one-on-one instruction automatically in a cost effective manner.
- ITS-taught students generally learn faster and translate the learning into improved performance better than classroom-trained participants.
- Provides direct feedback to the students without the intervention of human beings.

## **1.3 Motivation**

LCMS and ITS systems are two ends of same rope. Both the systems have the basic goals of making the learning process efficient for the students and reducing the work required to be done by the teacher. However both the systems differ in the way they go about achieving the basic goals. LCMS provide a platform where it is easier for the teacher to upload content, students have a central place for all their learning materials and the discussion/questions are extended out from the physical boundaries of the classroom. LCMS systems are easier to build

and such systems involve active participation from the student community. However LCMS suffer from the problem that they have no intelligence built into them.

ITS systems are intelligent. They model how a teacher would teach in the class and also keep a track of the student's performance. Such systems use the record of students performance to enhance their learning process. However these systems are expensive to build and model. They require much human expertise and are domain specific. Thus we see that neither LCMS nor ITS system is a one stop solution to making the learning process efficient. ITS systems are difficult to build and LCMS aren't intelligent enough. An intelligent application would be the one that would combine the benefits of both the systems into one, i.e. it is as easy a CMS for the use of the teacher (eliminating the task of annotating the sources), involves high participation from the user and is also intelligent enough to make the user learning process easier and efficient.

## **1.4 Proposed Approach**

### **1.4.1 Introduction**

We propose a LMS, *Intinno*, motivated from the above discussion. Intinno being an LMS would be easy to build and would have all the benefits of an LCMS. To make Intinno intelligent we propose the addition of intelligent applications to the system. These applications would be directly integrated in the LCMS and would use Data-mining techniques [6] to make the learning process efficient. Till date there has been no work on integrating intelligent application into an LCMS that enhance the students learning process. However there have been several attempts to develop stand alone intelligent applications.

The work in [7] focuses on helping a teacher moderate a classroom of students using e-discussion tools in which the students comprise multiple discussion groups. Generally a teacher can bring to bear his or her experience and moderation expertise to steer the discussions when problems occur and provide encouragement when discussions are productive. However when multiple e-discussions occur simultaneously, a single teacher may struggle to follow all of the discussions. To direct the teacher's attention to the 'hot spots,' this paper proposes software tools that pre-process, aggregate, and summarize the incoming flood of data. [8] proposes Information

Retrieval techniques to detect conflicts within the same exam. A Conflict exists in an exam if at least two questions within that exam are redundant in content, and/or if at least one question reveals the answer to another question within the same exam. However none of these mentioned application aim to enhance the student's learning process.

### **1.4.2 Useful Applications**

From the student's point of view, an application that abstracts the learning material and presents it in way that is easier to grasp would be highly useful. Such an application would help the student to learn the whole concept by learning small related set of concepts. Proposed set of such applications are:

- Memory Maps: This application would extract a set of keywords from the document and present a graph of these keywords connected by a set of associations. It would be similar to making an automatic memory map of the concept to be learned.
- Presentation Module: This module would abstract a given page in the form of a power point presentation. This application would help the users to grasp a few important aspects of the concepts to be learned. It would also prove as the starting for preparing presentations from a given piece of content.

With the explosion of Web2.0 the content on the web has increased manifold. This has provided the user with treasures of information. However there is a catch here. Since the number of sources that present the same content to the user is large, a large portion of the user's time is spent in searching for appropriate material on the web. The same logic applies to students. Also there exist no educational search engines that focus only on educational content. Applications that can be developed to cater to the above problems:

- Recommendation Engine: This engine would implicitly help the user in getting similar content. When a student is reading a lecture then similar content would be automatically recommended to him. The recommendation would be content diversified, i.e. if the

person is currently studying Lectures then he/she will be recommended questions/quizzes/Assignments. however on the other hand if the person is bust doing Assignments or solving questions then he/she will be recommended lectures/tutorials on that particular subject from the digital library. Recommendation will be personalized i.e it will be based on the courses that the user has done and also on his/her level of understanding which can be judged from his courses list.

- Educational Search Engine: This will provide two additional capabilities in addition to keyword based search, namely (i) Content based search for similar courses, and (ii) Intelligent search for course materials (i.e. if a search is given for material on biochemistry course then materials from molecular biology course should also turn up in the results.

Until now the applications mentioned above have focused only on improving the efficiency of learning from the student's point of view. However an intelligent LCMS should minimize the efforts required from an instructor. Thus the following applications are proposed:

- Automatic Evaluator for Coding Assignments: In large number of courses the assignments given to students require a coded solution. The final answer is same for every student. This evaluator will automatically check the answers minimizing the efforts required from the instructors.
- Automatic Question Answering: One of the major advantages of LCMS is the discussion forums. However it may so happen that a question/query asked by a particular student may have been answered before in some other course. This module will seek out such answers and will present them to the users, automatically.
- Duplicate Detection in Assignments: This application would detect duplication in submitted assignments by the use of text matching algorithms. The instructor would be provided of the percentage match between any two submitted assignments.

The current LCMSs have the drawback of non availability of free content. LMS's assume that the content will be put up by users i.e. teachers and students. This leads to the cold start problem. Instructors who begin to make up a course don't have the material to start up. Our system solves the above problem to a large extent. The web interfaced educational digital library will solve the cold start problem faced by instructors. While putting up new course, assignment or a lecture, similar resources would be available from the digital library either by search or by recommendations.



## Chapter 2: Architecture of Intinno

### 2.1 Introduction

The functionalities provided by the system include, web mining, learning content management, semantic representation of mined knowledge, search and recommendation. Accordingly, Intinno has the following major components:

1. Focused Web Miner for Learning Content
2. SCORM module
3. Knowledge Representation Module
4. Intelligent Application Builder

A block diagram of Intinno system is shown in Figure 2. The system tasks may be mainly classified into the following major steps:

1. Building up of digital library from the content crawled from the web. The subtasks of this step are
  - a. Collection of resources from the web
  - b. Automatic Tagging of collected data for indexing, i.e. converting the downloaded data into SCORM compatible format.
  - c. Extraction of semantic knowledge from the SCORM compatible content and representing it in the form of ontology.
2. Application Builder Module:
  - a. This module uses the knowledge stored in the digital library for developing intelligent applications.
3. User Interaction:
  - a. User mainly interacts with the Intinno system through the Applications developed and the LCMS Module. The user interactions are recorded for further use by the applications to enhance user's learning process.

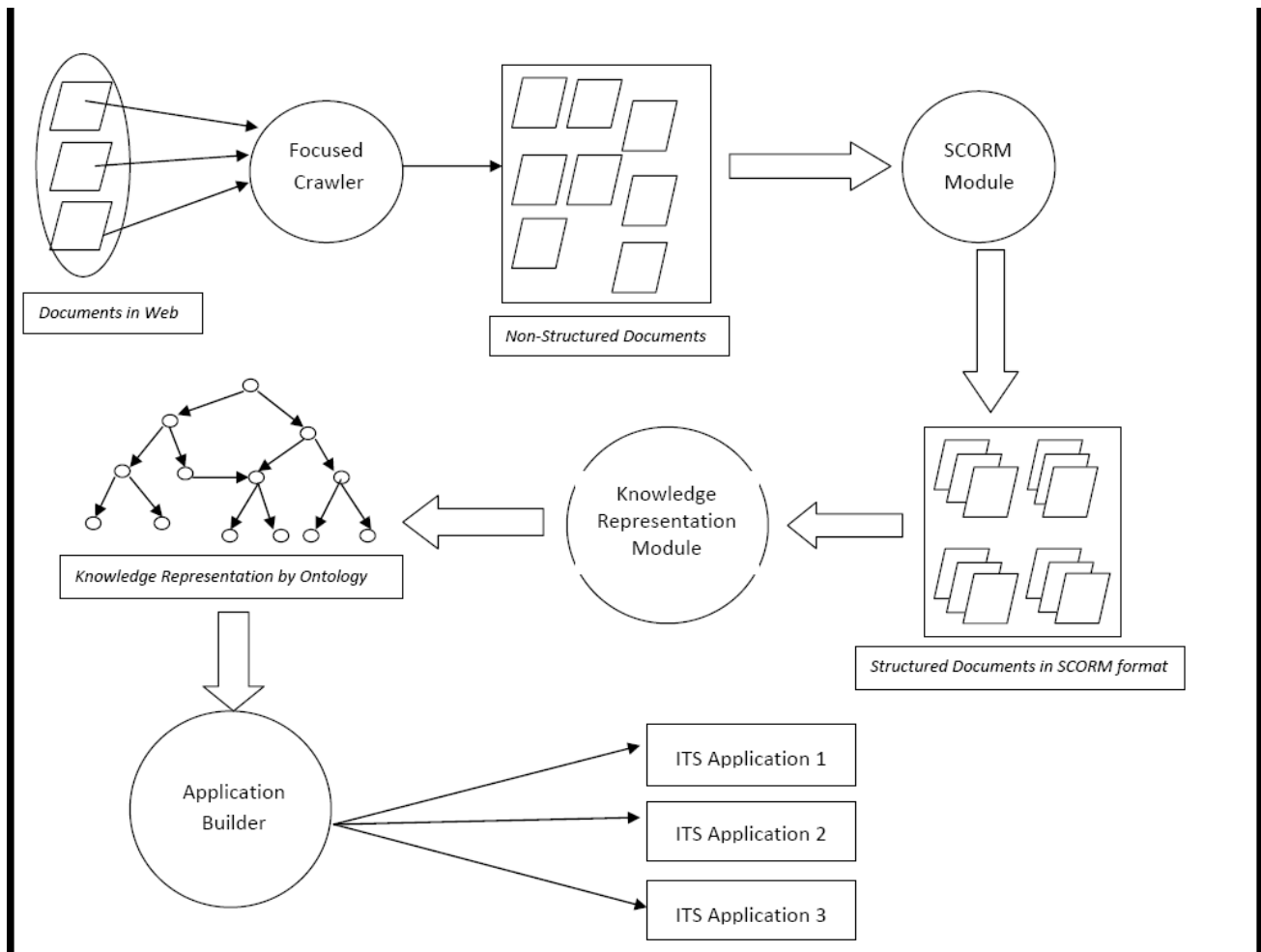


Figure 2: Block Diagram of the Intinno LCMS

## 2.2 Learning Resources on Web

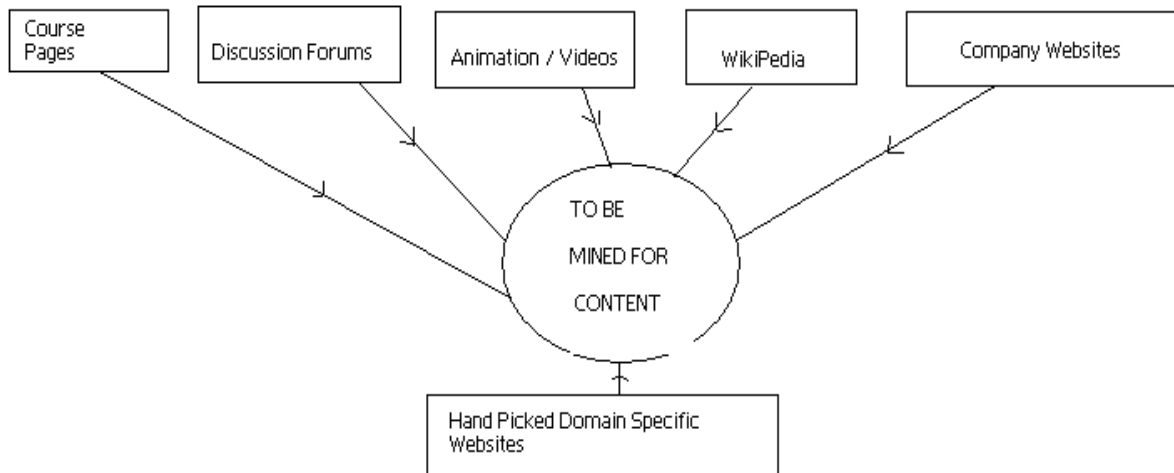
Web being a rich repository of learning content, we attempt to collect high volume of learning material from web using a web miner [10]. The type of content required for the digital library would include.

1. Courses
2. Assignments
3. Lectures and Tutorials
4. Animation and Videos
5. Case Studies

6. Questions and Quizzes
7. Information of relevant technologies from the industry.

The content described above can be mined from the following major resources:

- (a) Websites hosting standardized, reviewed and open source course material like MIT Open Courseware [12], NPTEL [13] India.
- (b) Course websites of large international universities. We have considered US universities currently.
- (c) Discussion Forums - Google Groups, Yahoo Answers
- (d) Websites for animations/videos - Youtube, Google Video and metacafe
- (e) Websites for general content - Wikipedia, Mathworld
- (f) Company Websites for product related info and case studies
- (g) Domain specific websites for questions, tutorials etc.



*Figure 3: Pictorial description of sources on the web to be mined for*

Strategies for crawling the above resources are mentioned below.

### **2.2.1 Open source and reviewed course pages**

A general purpose crawler to crawl all the courses from MIT OCW and NPTEL is employed. Content is structured and thus is easier to crawl. Also it provides us a list of basic courses to include in the digital library. Courses from MIT OCW can be downloaded directly and the download data is arranged into folders. The content from NPTEL is ad hoc and cannot be downloaded directly. Hence, data downloaded from NPTEL will have to be catalogued.

### **2.2.2 Content from University course pages**

Full crawl is not possible in this case and we opt for focused crawling [14]. Focused crawling is possible due to the following observations in most universities page structures.

- Every University page has a page listing all its schools and departments
- Every Dept will have a page listing all its faculty members
- Every faculty member will have links of the courses on his home page.

The above structure is utilized to tune the focused crawler. Crawling the above pages is modeled as a problem of sequence learning. CRF [15] techniques are used to learn the model parameters of the sequence learning problem and Reinforcement learning [16] is used to prioritize the download of links. A detailed description of the crawling technique is presented in Chapter 3.

Other than using the complex approach of focused crawling, we also tried our hands at another simpler approach. We attempted direct search on Google for university course material. Using Google search keywords of the form: <name of course> course page syllabus returns course pages. However this approach has the following problems:

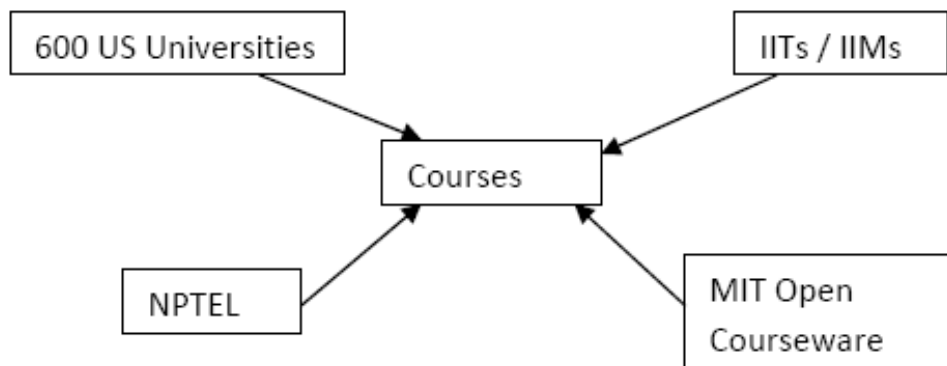
- We need a pre defined, exhaustive list of courses in order to find them through a search engine, which is always not possible.
- Relying on a search engine for Data Mining is not a feasible option as the number of automatic queries that can be issued per day are limited.

- It is difficult to quantify the recall of this method.

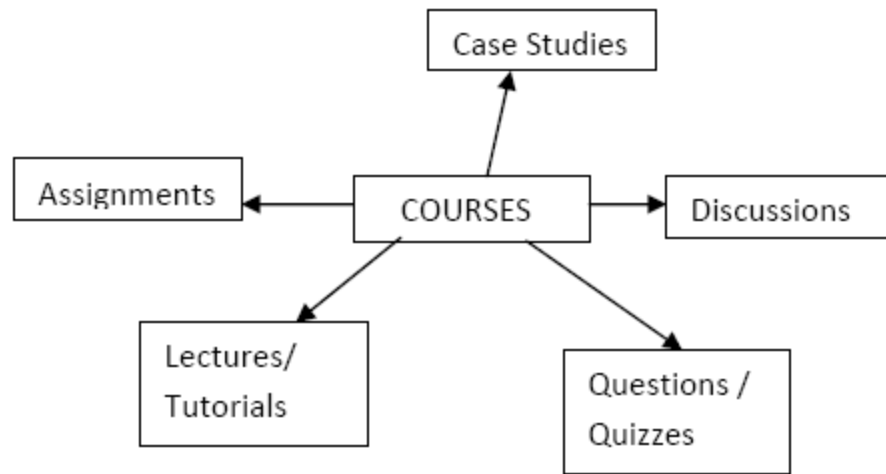
Another issue involved for such course pages is that of extraction of learning content from courses on the web. The data downloaded from a course on the web, may be arranged in various ways and needs to be processed to extract the relevant information. The data is represented in SCORM format to enable easier exchange and reuse. Here we propose a simplistic algorithm for doing it in each of the following two cases:

**Case 1:** All the data of a particular course lies on one page. In this case, different kinds of data will be under corresponding headings. For example all the assignments of a particular course will be under the assignments headings and all the questions will be under the questions heading. To extract data from such a course, we detect the headings on a particular page and we hypothesize that all the data under a heading is of that type. The algorithm for the detection of headings has about 65% accuracy.

**Case 2:** The course page has separate links for separate kind of data i.e. the assignments are on one page and the questions on another. We assume that these separate pages have such an anchor text that indicates the type of content on the page. For example the main course has links to Assignments / Lectures and Quizzes. To extract data from such a course we assume that type of content on each page to be given by the anchor text on the hyperlink.



*Figure 4: Sources of course material on the*



---

*Figure 5: Type of contents available from course*

### 2.2.3 Unstructured data: Discussion forums and animation videos

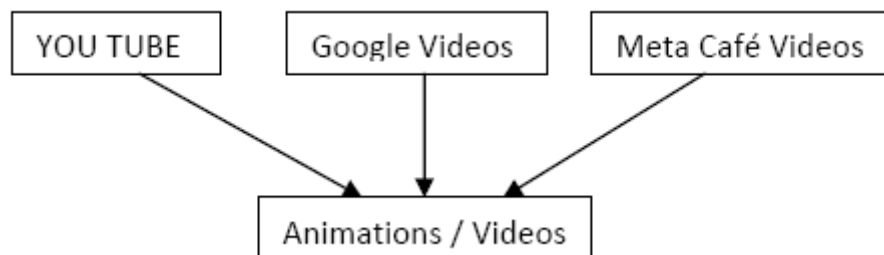
Full Crawl is irrelevant and is also not possible. Focused crawling is the approach adopted to get related content. From the courses already stored in the digital library now extract a set of keywords:

- Terms from the name of the course
- Terms from the syllabus of the course
- Terms from assignment heading/name, Lecture heading/name.

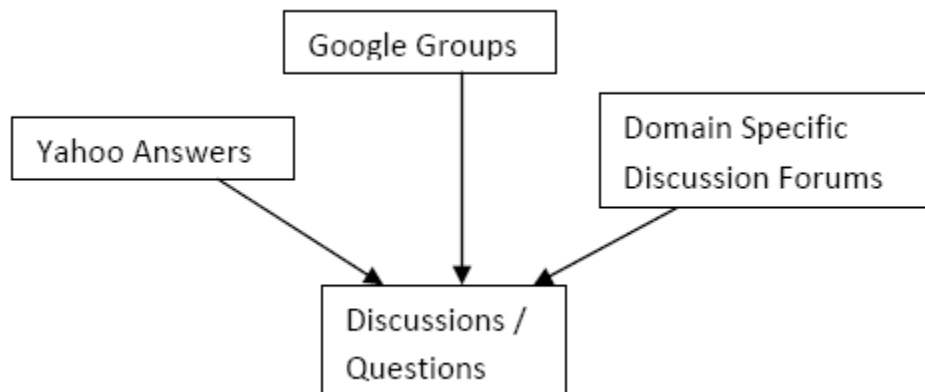
Next we search for discussions/Animations/Videos from the web which match the above list of keywords and index the results obtained above with the keywords with which they were found and the content of the entity obtained.

## 2.2.4 General purpose collections like Wikipedia

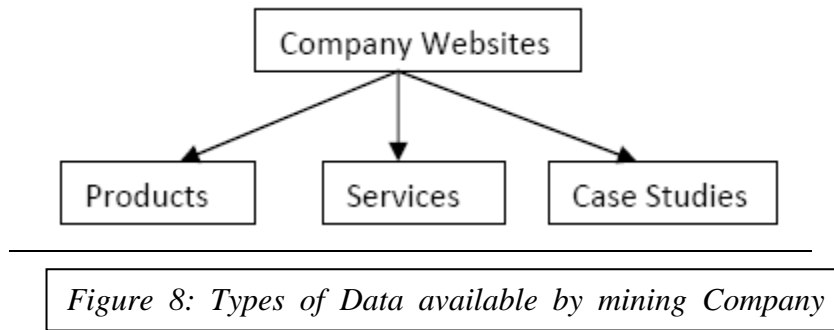
Full Crawl of Wikipedia is possible and can be obtained as a single XML document. However, full crawl/download may not be necessary and may in fact weaken precision of the search on digital library. We use a keyword based focused approach described above to limit the pages being indexed in wikipedia. Each wikipedia article can be characterized a lectures or tutorials. While indexing the articles of Wiki more importance should is given to the headings and the sub headings on the page.



*Figure 6: Sources of Animations and Videos on the*



*Figure 7: Sources of Discussions/Questions on the*



### **2.2.5 Websites of industrial corporations**

Websites in these categories will have to be handpicked and will be few in number. Examples of company websites include whitepapers, manuals, tutorials obtained from research labs of companies like IBM, Google, Microsoft, GE. Handpicked websites of popular corporate training resources like those offering questions/quizzes on C and those offering tutorials like How Stuff Works.

## **2.3 Mining Resources from the web**

### **2.3.1 Why a focused Crawler?**

The size of the publicly indexable world-wide-web has provably surpassed one billion documents [17] and as yet growth shows no sign of leveling off. Dynamic content on the web is also growing as time-sensitive materials, such as news, financial data, entertainment and schedules become widely disseminated via the web. Search engines are therefore increasingly challenged when trying to maintain current indices using exhaustive crawling. Even using state-of-the-art systems such as Google, which reportedly crawls millions of pages per day, an exhaustive crawl of the web can take weeks. Exhaustive crawls also consume vast storage and bandwidth resources, some of which are not under the control of the search engine.



Focused crawlers [10, 11, 14] aim to search and retrieve only the subset of the world-wide web that pertains to a specific topic of relevance. The ideal focused crawler retrieves the maximal set of relevant pages while simultaneously traversing the minimal number of irrelevant documents on the web. Focused crawlers therefore offer a potential solution to the currency problem by allowing for standard exhaustive crawls to be supplemented by focused crawls for categories where content changes quickly. Focused crawlers are also well suited to efficiently generate indices for niche search engines maintained by portals and user groups [18], where limited bandwidth and storage space are the norm. Finally, due to the limited resources used by a good focused crawler, users are already using personal PC based implementations. Ultimately simple focused crawlers could become the method of choice for users to perform comprehensive searches of web-related materials.

### **2.3.2 Content from University Course pages**

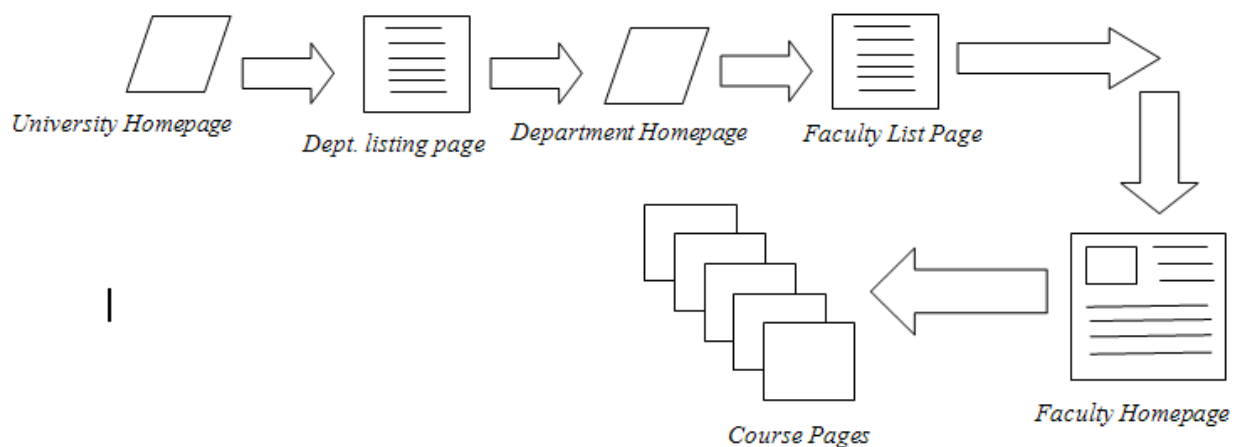
Out of the above mentioned sources, course websites of different Universities are the richest source of learning content. The advantages of this content are:

1. Since this content is hosted on the University site, under the professor/teacher taking this course, the content is deemed to be authenticated and correct.
2. Also this type of content is used in a real scenario to teach the students and hence is most relevant to the students.

However, along with being the richest source of valid educational content this type of content is most difficult to mine. This is due to the fact that this content is non-structured in nature. There are following difficulties in mining this content:

- Every teacher has his/her own way of hosting the content. Some might be putting up the whole content in a single page while others might be having a more structured representation of content with different sections for assignments, lectures, etc.
- Every University has their own sitemap. A set of rules to reach the course pages starting from University homepage, if designed for a particular university, might not work for every case.

One of the solutions to get the course pages from a particular university would be to crawl the whole university and separate out the course pages from the set of all crawled pages. The separation of course pages will be done by a binary classifier that will be trained on the prior set of course pages that can be obtained with the help of a search engine. However crawling the whole university for course pages would be inefficient both in terms of Time and Space required. Hence we need a focused crawling [11] technique to efficiently mine relevant course pages starting from the university homepage.



*Figure 9: Sequential mining of course pages starting from university*

### **2.3.3 Possible approaches of mining course content from university homepages**

One possible method of solving the problem is to train a classifier that can discriminate the goal pages from the non-goal pages. Then, extract from the classifier the set of prominent features to serve as keywords to a search engine that index all the websites of interest. By restricting the domain to each given starting URL in turn, we issue a keyword search to get a set of candidate pages. We further classify these pages to identify if these are goal pages or not. However this method cannot provide high accuracy for the simple reason that the goal page itself may not hold enough information to correctly identify it as the goal page. The path leading to the goal page is important too.

The major open problem in focused crawling is that of properly assigning credit to all pages along a crawl route that yields a highly relevant document. In the absence of a reliable credit assignment strategy, focused crawlers suffer from a limited ability to sacrifice short term document retrieval gains in the interest of better overall crawl performance. In particular, existing crawlers still fall short in learning strategies where topically relevant documents are found by following off-topic pages.

The credit assignment for focused crawlers can be significantly improved by equipping the crawler with the capability of modeling the context within which the topical materials is usually found on the web [19]. Such a context model has to capture typical link hierarchies within which valuable pages occur, as well as describe off-topic content that co-occurs in documents that are frequently closely associated with relevant pages.

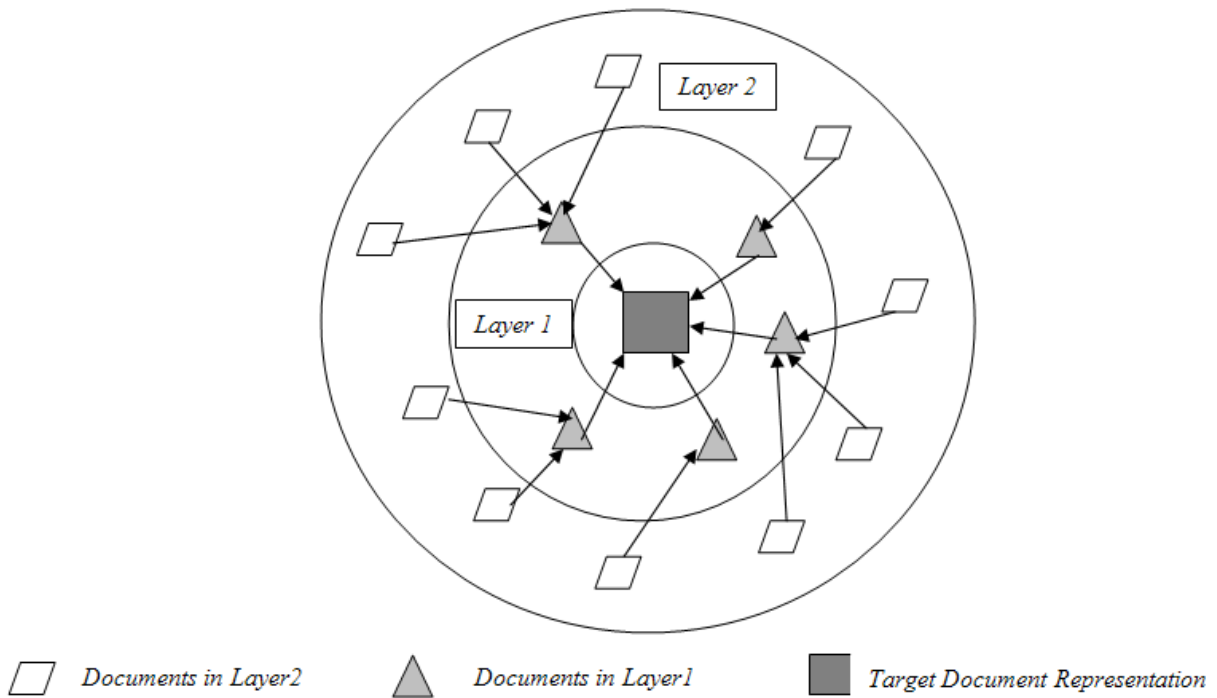


Figure 10: A context graph represents how a target document can be accessed from the web. In each node a web document representation is stored. The graph is organized into layers: each node of layer  $i$  is connected to one (and only one) node of the layer  $i-1$  (except the single node in layer 0). There are no connections between nodes at the same level. The seed document is stored in layer 0. A document is in layer  $i$  if at least  $i$  steps (link followings) are needed to reach the target page starting from that document.

A Focused crawler must use information gleaned from previously crawled page sequences to estimate the relevance of a newly seen URL. Therefore, good performance depends on powerful modeling of context as well as the current observations. Probabilistic models, such as Hidden Markov Models( HMMs)[20] and Conditional Random Fields(CRFs)[15], can potentially capture both formatting and context.

Thus a second approach and the one that we use in Intinno is to treat crawling as a sequential labeling problem where we use Hidden Markov Models (HMMs) and the Conditional Random Fields to learn to recognize paths that lead to goal states. We then superimpose ideas from Reinforcement Learning [16] to prioritize the order in which pages should be fetched to reach the goal page. This provides an elegant and unified mechanism of modeling the path learning and foraging problem.

#### **2.3.4 Our Approach**

There are two phases in our approach:

1. **Training phase:** where the user teaches the system by clicking through pages and labeling a subset with a dynamically defined set of classes, one of them being the Goal class. The classes assigned on intermittent pages along the path can be thought of as “milestones” that capture the structural similarity across websites. At the end of this process, we have a set of classes  $C$  and a set of training paths where a subset of the pages in the path are labeled with a class from  $C$ . All unlabeled pages before a labeled page are represented with a special prefix state for that label. The system trains a model using the example paths, modeling each class in  $C$  as a milestone state.
2. **Crawling phase:** where the given list of websites is automatically navigated to find all goal pages. The system uses the model parameters learnt in the training phase to prioritize the download of links trying to optimize the ratio of the number of relevant pages downloaded to the total number of pages downloaded.

Formally, we are given a website as a graph  $W(V,E)$  consisting of vertex set  $V$  and edge set  $E$ , where a vertex is a webpage and an edge  $e = \langle u, v \rangle$  is a hyperlink pointing from a webpage  $u$  to a webpage  $v$ . The goal pages  $P_G$  constitute a subset of pages in  $W$  reachable from starting seed page  $P_S$ . We have to navigate to them starting from  $P_S$  visiting fewest possible additional pages. Let  $P : P_1, P_2, \dots, P_n$  be one such path through  $W$  from the start page  $P_1 = P_S$  to a goal page  $P_n \in P_G$ . The ratio of relevant pages visited to the total number of pages visited during the execution is called the **harvest rate**. The objective function is to maximize the harvest rate.

This problem requires two solutions.

1. **Recognizing a page as the goal page.** This is a classification problem where given a webpage we have to classify it as being a goal page or not. Often the page alone may not hold enough information to help identify it as the goal page. We will need to consider text around the entire path leading to the goal page in order to decide if it is relevant or not. For example, if we want to get all course pages starting from a university root page, then it is necessary to follow a path through departments' homepages and then through professors' homepage. A course page on its own might be hard to classify.
2. **Foraging for goal pages.** This can be thought as a crawling exercise where, starting from the entry point, we want to visit as few pages as possible in finding the goal pages. This problem is different from the previous work on focused crawling [11] where the goal is to find all web pages relevant to a particular broad topic from the entire web. In our case, we are interested in finding course pages starting from a University homepage. We exploit the regularity in the structures of University websites to build more powerful models than is possible in the case of general-purpose focused crawlers.

## 2.4 Domain Knowledge Representation

## 2.4.1 Introduction

There are representation techniques such as frames, rules and semantic networks which have originated from theories of human information processing. Since knowledge is used to achieve intelligent behavior, the fundamental goal of knowledge representation is to represent knowledge in a manner as to facilitate inferences (i.e. drawing conclusions) from knowledge. Problem Solving can be simplified by an appropriate choice of *knowledge representation (KR)*. Representing knowledge in some ways makes certain problems easier to solve.

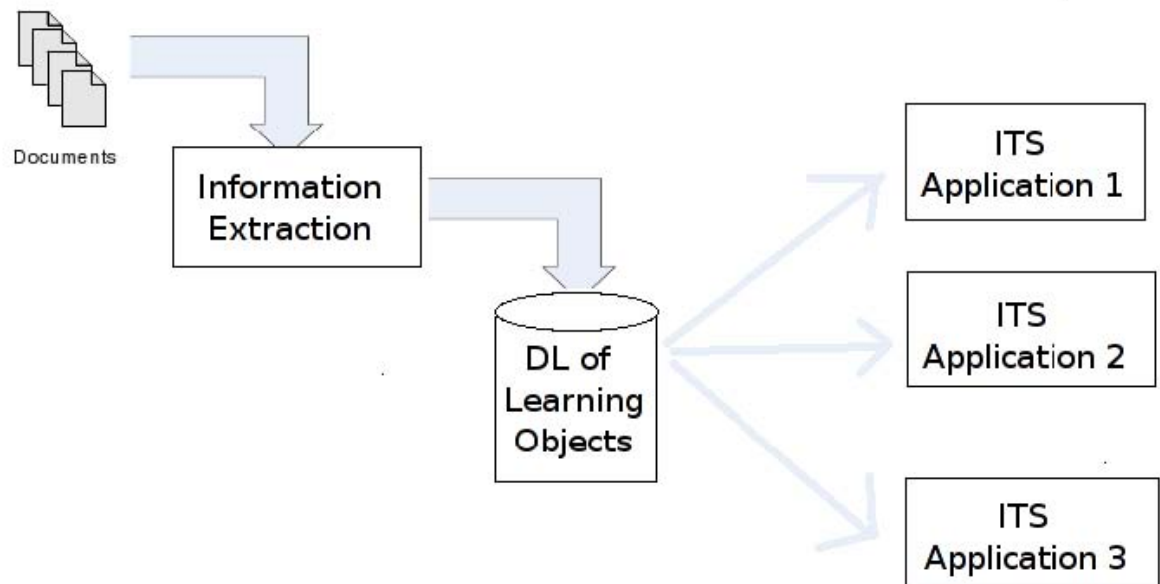


Figure 15: Applications of Knowledge Representaion

KR is most commonly used to refer to representations intended for processing by computers, and in particular, for representations consisting of explicit objects (the class of all humans, or Ram a certain individual), and of assertions or claims about them ('Ram is a human', or 'all humans have one head'). Representing knowledge in such explicit form enables computers to draw conclusions from knowledge already stored ('Ram has one head').

## **2.4.2 Why Knowledge Representation ?**

The knowledge is crucially important in the development of an intelligent tutoring system for e-learning. For this work, we assume that we have a repository of educational documents mined from the web. The content described above can be mined from the following major resources

- (i) MIT Open Courseware, NPTEL India
- (ii) .edu domain
- (iii) Discussion Forums -Google Groups, Yahoo Answers
- (iv) YouTube, Google Video and Metacafe
- (v) Wikipedia, MathWorld
- (vi) Company Websites for product related info and case studies
- (vii) Domain specific websites for questions, tutorials etc.

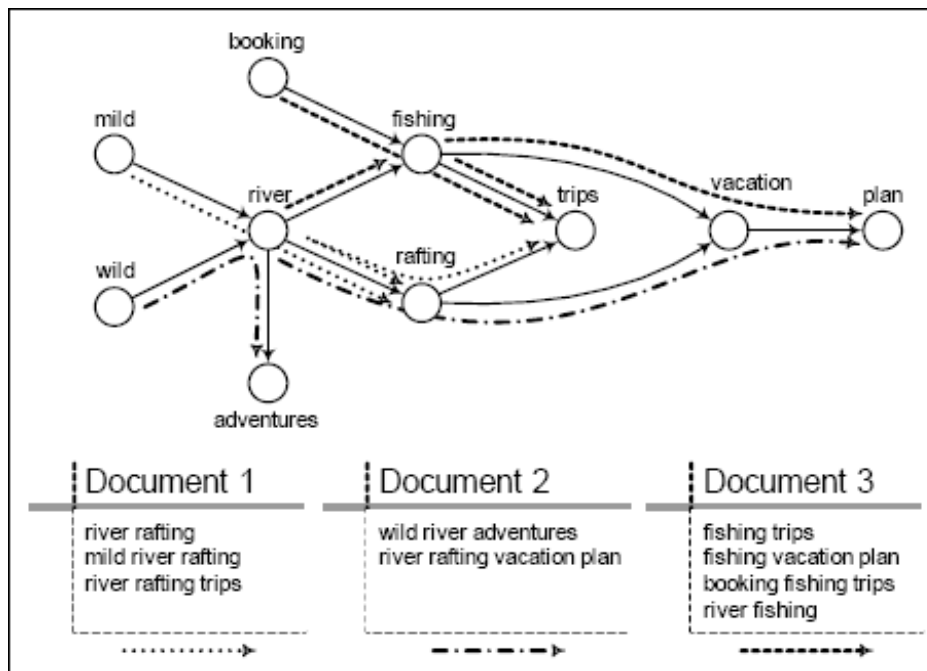
Open repositories like Wikipedia and information pages authored as blogs etc:- by casual users if used efficiently can be a very good resource for learning. All this knowledge needs to be represented efficiently for use by e-learning systems.

The goal of this work is to explore approaches for representation of knowledge for efficient use of resources for an intelligent learning system. We intend to find an approach which can help in capturing the semantics of the crawled resources and efficiently implement a set of learning applications. Hence the final goal is to have a knowledge representation technique specially designed to support intelligent tutoring applications like automatic annotation of text and construction of memory maps.

## **2.4.3 An Approach**

The work by Hammouda and Kamel [25] presents an innovative approach for performing data mining on documents, which serves as a basis for knowledge extraction in e-learning environments. The approach is based on a radical model of text data that considers phrasal features paramount in documents, and employs graph theory to facilitate phrase representation and efficient matching. In the process of text mining, a grouping (clustering) approach is also employed to identify groups of documents such that each group represents a different topic in the underlying document collection. Document groups are tagged with topic labels through unsupervised key phrase extraction from the document clusters.

The model presented by Hammouda and Kamel [25] for document representation is called the Document Index Graph (DIG). This model indexes the documents while maintaining the sentence structure in the original documents. This allows use of more informative phrase matching rather than individual words matching. Moreover, DIG also captures the different levels of significance of the original sentences, thus allowing us to make use of sentence significance. Suffix trees are the closest structure to the proposed model, but they suffer from huge redundancy [26].





*Figure 16: Example of the Document Index Graph*

The DIG is built incrementally by processing one document at a time. When a new document is introduced, it is scanned in sequential fashion, and the graph is updated with the new sentence information as necessary. New words are added to the graph as necessary and connected with other nodes to reflect the sentence structure.

Upon introducing a new document, finding matching phrases from previously seen documents becomes an easy task using DIG. This is done by incremental graph building and phrase matching. The approach serves in solving some of the difficult problems in e-learning where the volume of data could be overwhelming for the learner; such as automatically organizing documents and articles based on topics, and providing summaries for documents and groups of documents.

#### **2.4.4 Our Approach – The Motivation**

The archived content crawled by the focused crawler is used to develop concept maps that capture the various semantic relations among data. The semantic content from the digital library is then used to develop intelligent learning applications. These applications focus on making the learning process for a student both efficient and effective.

The knowledge representation is crucially important in the development of an intelligent learning system for e-learning. In order to use the document corpus effectively and efficiently, not only the contents but also the representation of contained knowledge is important. The effectiveness of the learning applications like memory maps will depend significantly on the knowledge representation architecture. The content mined from the web can be divided into two categories in terms of its usability for an e-learning system.

- (i) Structured content (courses)
- (ii) Non-Structured educational content.

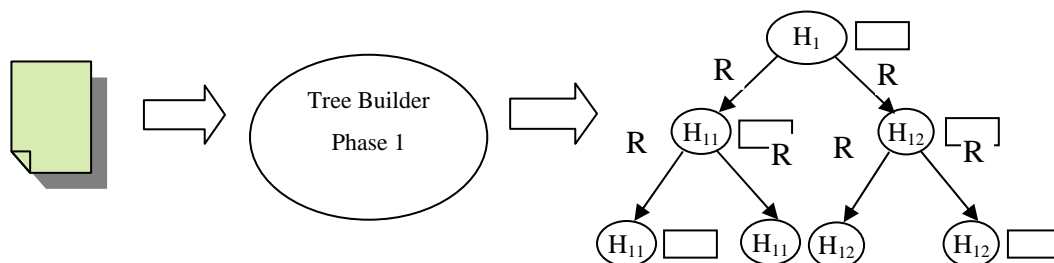
*Structured course content* crawled in section 3 is annotated before it is archived in the digital library. We identify a set of tags which are required to represent the course in SCORM standard. We extract information from the structured courses to convert them into (semi-) SCORM format. In addition to the above tags we also store some entity specific meta-tags that are important from the point of view of indexing and parameterized search. For both the above category of tags hand crafted wrappers are used for information extraction [27]. Adding the search keywords in the meta-tags ensures that information about related course/course material is added in the tags of the entity. This will ensure that if the search is made in the name of the course then related material also turns up in the results.

*Non-Structured content* is available in abundance on the web. Open repositories like Wikipedia and information pages authored as blogs etc:- by casual users if used efficiently can be a very good resource for learning. All this knowledge needs to be represented efficiently for use by e-learning systems. In our work, we report the existing approaches. We also propose a new approach for automatic construction of concept maps from the content mined from the web. Our knowledge representation technique is specially designed to support intelligent tutoring applications like automatic annotation of text and construction of memory maps. We have designed and implemented a heuristic based algorithm to extract the headings from web documents.

### 2.1.5 Our Design

The document representation module consists of the three main modules namely:

- Tree Building Phase 1
- Tree Building Phase 2
- Key-phrase Resolution



Heading Extraction

Text Resolution

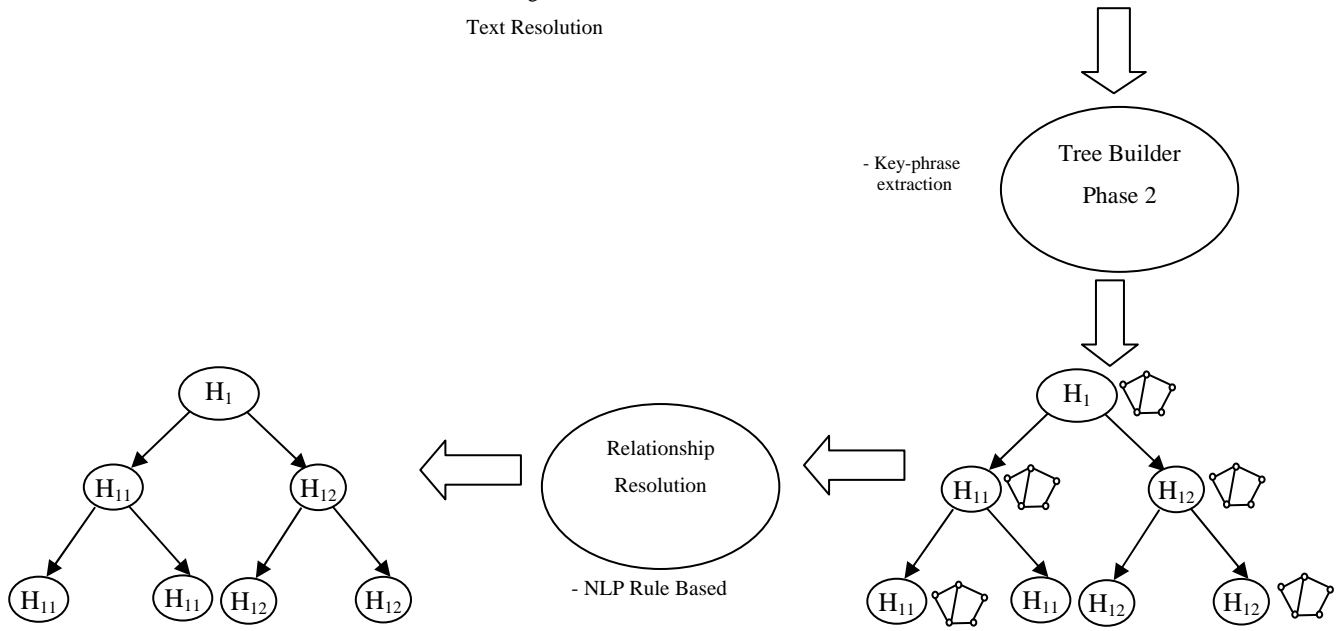


Figure 17: Document representation process

## **Chapter 3: Intelligent Applications for Intinno**

### **3.1 Introduction**

Section 1.3 discusses that neither ITS nor LCMS are one stop solutions for enhancing the learning process. LCMS provide a platform where it is easier for the teacher to upload content, students have a central place for all their learning materials and the discussion/questions are extended out from the physical boundaries of the classroom. LCMS systems are easier to build and such systems involve active participation from the student community. However LCMS suffer from the problem that they have no intelligence built into them.

ITS systems are intelligent. They model how a teacher would teach in the class and also keep a track of the student's performance. Such systems use the record of student's performance to enhance their learning process. However these systems are expensive to build and model. They require much human expertise and are domain specific.

The proposed solution to this problem was to build an intelligent LCMS application which is composed of two parts: 1) Content Management System and 2) Intelligent applications. Intelligent applications are the features that are directly integrated into the content management system. These applications take advantage of the semantic knowledge contained in the data and the user behavior recorded during the interactions with the content management system, to make the learning process efficient for both teachers and students.

The list of such applications that can be developed are a) Memory Maps, b) PPT Summarization Module, c) Recommendations, d) Educational Search Engine e) Automatic Evaluator for coding Assignments, f) Duplicate Detection Module and g) Automatic Question Answering system. Figure 11 shows the applications to be developed and the motivations for each of them. In the next sections we present related works that can be used to develop some of the above mentioned applications.

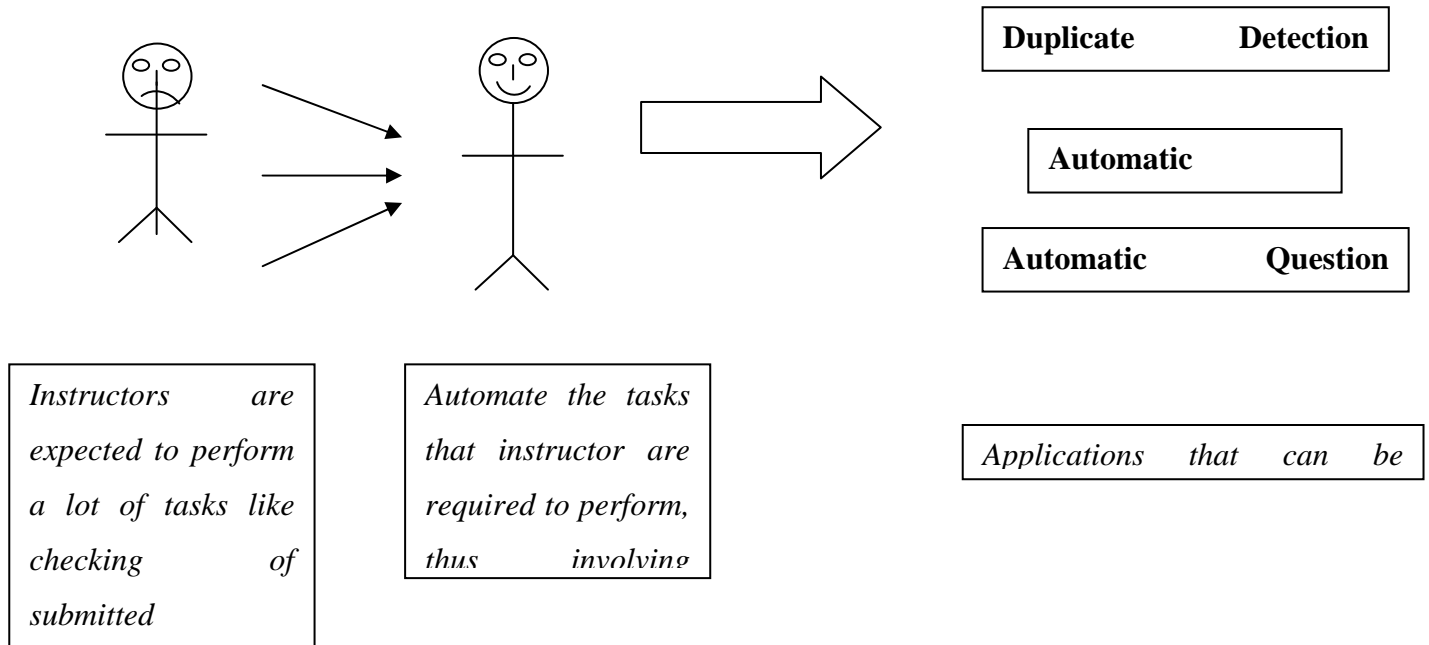
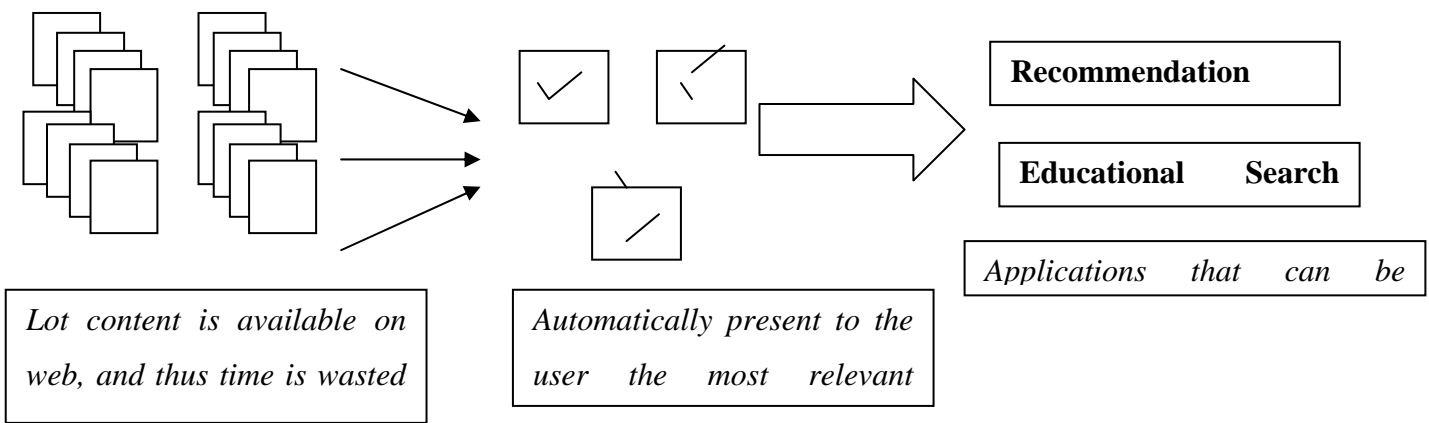
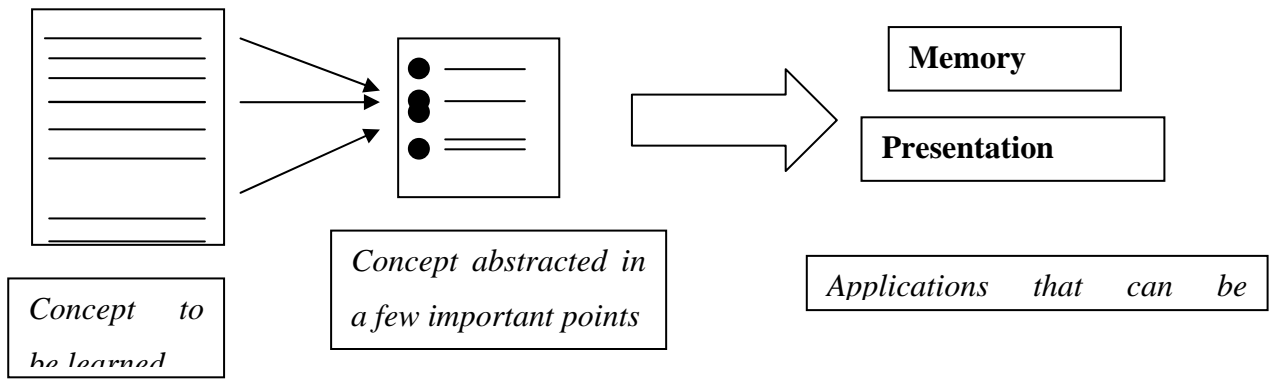


Figure 11: Intelligent applications for the Intinno LCMS, along with their

## **3.2 Memory Maps**

### **3.2.1 Need for Memory Maps**

Memory maps provide a general, powerful, and user-oriented way to navigating the information resources under consideration in any specific domain. A topic map provides a uniform framework that not only identifies important subjects from an entity of information resources and specifies the information resources that are semantically related to a subject, but also explores the relations among these subjects. When a user needs to find some specific information on a pool of information resources, he only needs to examine the topic maps of this pool, selects the topic he thought interesting, and the topic maps will show him the information resources that are related to this topic as well as its related topics. He will also recognize the relationships among these topics and the roles the topics play in such relationships. With the help of the topic maps, we no longer have to browse through a set of hyperlinked documents and hope they may eventually reach the information we need in a finite amount of time even when we know nothing about where we should start. We also don't have to gather some words and hope that they may perfectly symbolize the idea we interest and are conceived well by a search engine to obtain reasonable result. Topic maps provide us a way to navigate and organize information, as well as to create and maintain knowledge in an info glut.

Topic Maps provide an external meta-structure (a knowledge navigation layer) in the form of a dynamic, semantically based hypertext. As a result, TM-based courseware can offer the following benefits:

- For learners: efficient context-based retrieval of learning resources; better awareness in subject-domain browsing; information visualization; customized views, adaptive guidance, and context-based feedback.
- For instructors: effective management and maintenance of knowledge and information; personalized courseware presentations; distributed courseware development; reuse and exchange of learning materials, collaborative authoring.

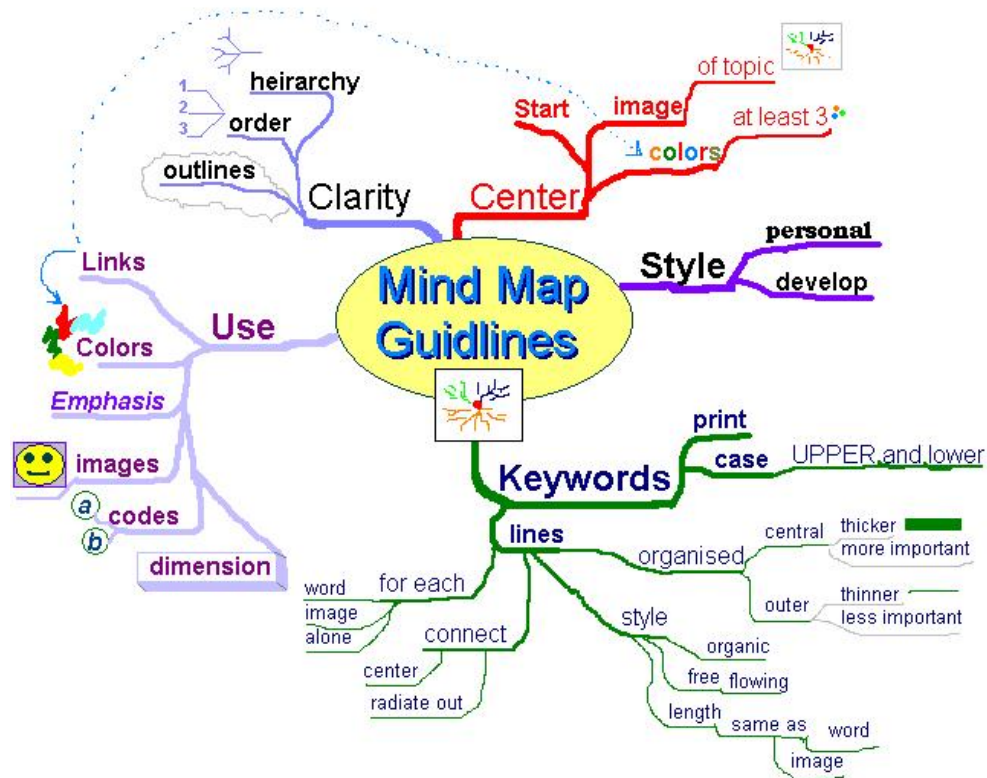


Figure 12: Example of a memory

To construct a topic map for a set of information resources, human intervention is unavoidable in present time. We need human effort in tasks such as selecting topics, identifying their occurrences, and revealing their associations. Such need is acceptable only when the topic maps are used merely for navigation purpose and the volume of the information resource is considerably small. However, a topic map should not only be a 'topic navigation map'. Besides, the volume of information resource under consideration is generally large enough to prevent manual construction of topic maps.

To expand the applicability of topic maps, some kind of automatic process should involve during the construction of topic maps. The degree of automation in such construction process may differ for different users with different needs. One may only need a friendly interface to automate the topic map authoring process, and another one may try to automatically identify every components of a topic map for a set of information resources from the ground up.

Possible approaches for building Memory Maps:

### 3.2.2 Approach 1

This approach is inspired from the study described in [21], which looked at the feasibility of using HTML tags as proxies for semantic content as well. Three Web page elements have been studied: *Link (anchor) text*, *Heading text*, and *Comment text*. Using the scale ‘no help’, ‘little help’, ‘helpful’, and ‘very helpful’, the human experts involved in the study evaluated that 61% of web page link text was ‘helpful’ or ‘very helpful’ in indicating semantic contents. Taking this as encouraging (especially because the pages have been arbitrarily chosen), this approach explores the possibility of extracting semantic information by parsing the web pages in a web site specified by the user.

This approach defines a set of heuristics to construct a topic map by parsing an HTML document. The heuristics are as follows:

1. A new topic is created in the topic tree for each web page visited by the crawler.
2. All the topics created in the process of parsing a specific web page are sub-topics of the “page” topic for that page.
3. Naming of “page” topics:
  - a. The “root” topic provided by the user in the Extractor interface is considered as corresponding to the webpage, which is the “entry point” for the crawler (with the specified URL).
  - b. All other “page” topics are named using the text provided in the body of the corresponding anchor element.
4. Heading represents a topic that is more general than the topics extracted from the text below it (if any).
5. The topics extracted from HTML headings of different types can be organized hierarchically so as to correspond to the HTML headings “weights” (1, 2, etc.). Thus the topic extracted from an <h2> tag will be a sub-topic of the topic extracted from the <h1> tag, etc.



6. Heading tags on a referenced web page (through an 'anchor' element, see 2.3.2) will be seen as relating to the topic representing that page.
7. The topics extracted from the cells of one column in a table are related since they can be considered as values of the same attribute (represented by the column header).
8. The topics extracted from the cells of one column in a table are subtopics of the topic corresponding to the column header.
9. Group the topics extracted from the same HTML element together since this grouping indicates some kind of relatedness of the topics.

The problem with the above approach is that it only offers a way to construct the draft topic map and the real power comes by the involvement of the user. Also it uses links/anchor text to construct one of the features of the memory maps. However this might not work in the case when we need a memory map from a single document itself.

### **3.2.3 Approach 2**

This work [22] presents a novel approach for semi-automatic topic map construction. The approach starts from applying a text mining process on a set of information resources. Two feature maps, namely the document cluster map and the word cluster map, are created after the text mining process. It then applies a category hierarchy development process to reveal the hierarchical structure of the document clusters. Some topics are also identified by such process to indicate the general subjects of those clusters located in the hierarchy. It then creates topic maps according to the two maps and the developed hierarchy automatically. Although this method may not identify all kinds of components that should construct a topic map, this approach seems promising since the text mining process achieves satisfactory result in revealing implicit topics and their relationships.

To reveal the relationships between documents, the popular self-organizing map (SOM) [23] algorithm is applied to the corpus to cluster documents. It adopts the vector space model to transform each document in the corpus into a binary vector. These document vectors are used

as input to train the map. It then applies two kinds of labeling process to the trained map and obtained two feature maps, namely the document cluster map (DCM) and the word cluster map (WCM). In the document cluster map each neuron represents a document cluster that contains several similar documents with high word co-occurrence. In the word cluster map each neuron represents a cluster of words that reveal the general concept of the corresponding document cluster associated with the same neuron in the document cluster map. The text mining process described above provides us a way to reveal the relationships between the topics of the documents.

The method to identify topics arranges them in a hierarchical manner according to their relationships. A neuron in the DCM represents a cluster of documents that contain words that often co-occurred in these documents. Besides, documents that associate with neighboring neurons contain similar set of words. Thus it constructs a super-cluster by combining neighboring neurons by the following algorithm:

- Find the neuron with the largest supporting cluster similarity. Selecting this neuron as dominating neuron.
- Eliminate its neighbor neurons so that they will not be considered as dominating neurons.
- If there is no neuron left or the number of dominating neurons exceeds a predetermined value, stop. Otherwise go to Step 1.

A dominating neuron may be considered as the centroid of a super-cluster, which contains several clusters. It then assigns every cluster to some super-clusters using a method similar to K-Means clustering. A super-cluster may be thought as a category that contains several subcategories. The category topics are selected from those words that associate with these neurons in the WCM. This scheme selects the word that is the most important to a super-cluster. The topics that are selected by the above mechanism form the top layer of the category hierarchy. To find the descendants of these topics in the hierarchy, it applies the above process to each super-cluster and obtains a set of sub-categories. These sub-categories form the new super-clusters that are on the second layer of the hierarchy. The category structure can then be revealed by recursively applying the same category generation process to each new-found super-cluster.

This approach also identifies the topic types by the constructed hierarchy. A topic on higher layers of the hierarchy represents a more important concept than those on lower layers. For a parent-child relationship between two concepts on two adjacent layers, the parent topic should represent a important concept of its child topic. Therefore, it uses the parent topic as the type of its child topics. Such usage also fulfills the requirement of the topic map standard that a topic type is also a topic.

Both the above approaches are contrasting in the sense that the first one uses heuristics and the second one uses pure data-mining approach to build the topic case. Our final approach will utilize the advantages of both the approaches for making the topic maps.

---

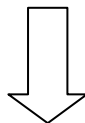
## Sorting algorithm

From Wikipedia, the free encyclopedia

In [computer science](#) and [mathematics](#), a **sorting algorithm** is an [algorithm](#) that puts elements of a [list](#) in a certain [order](#). The most-used orders are numerical order and [lexicographical order](#). Efficient [sorting](#) is important to optimizing the use of other algorithms (such as [search](#) and [merge](#) algorithms) that require sorted lists to work correctly; it is also often useful for [canonicalizing](#) data and for producing human-readable output. More formally, the output must satisfy two conditions:

1. The output is in nondecreasing order (each element is no smaller than the previous element according to the desired [total order](#));
2. The output is a [permutation](#), or reordering, of the input.

Since the dawn of computing, the sorting problem has attracted a great deal of research, perhaps due to the complexity of solving it efficiently despite its simple, familiar statement. For example, [bubble sort](#) was analyzed as early as 1956.<sup>[1]</sup> Although many consider it a solved problem, useful new sorting algorithms are still being invented (for example, [library sort](#) was first published in 2004). Sorting algorithms are prevalent in introductory computer science classes, where the abundance of algorithms for the problem provides a gentle introduction to a variety of core algorithm concepts, such as [big O notation](#), [divide-and-conquer algorithms](#), [data structures](#), [randomized algorithms](#), [best](#), [worst](#) and [average case](#) analysis, [time-space tradeoffs](#), and lower bounds.



---

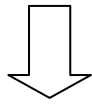
## Sorting algorithm

From Wikipedia, the free encyclopedia

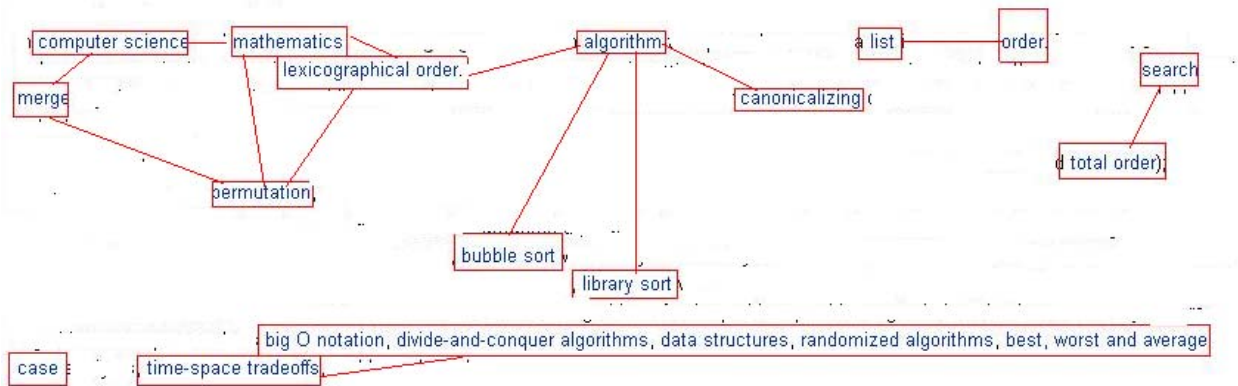
In [computer science](#) and [mathematics](#), a **sorting algorithm** is an [algorithm](#) that puts elements of a [list](#) in a certain [order](#). The most-used orders are numerical order and [lexicographical order](#). Efficient [sorting](#) is important to optimizing the use of other algorithms (such as [search](#) and [merge](#) algorithms) that require sorted lists to work correctly; it is also often useful for [canonicalizing](#) data and for producing human-readable output. More formally, the output must satisfy two conditions:

1. The output is in nondecreasing order (each element is no smaller than the previous element according to the desired [total order](#));
2. The output is a [permutation](#), or reordering, of the input.

Since the dawn of computing, the sorting problem has attracted a great deal of research, perhaps due to the complexity of solving it efficiently despite its simple, familiar statement. For example, [bubble sort](#) was analyzed as early as 1956.<sup>[1]</sup> Although many consider it a solved problem, useful new sorting algorithms are still being invented (for example, [library sort](#) was first published in 2004). Sorting algorithms are prevalent in introductory computer science classes, where the abundance of algorithms for the problem provides a gentle introduction to a variety of core algorithm concepts, such as [big O notation](#), [divide-and-conquer algorithms](#), [data structures](#), [randomized algorithms](#), [best](#), [worst](#) and [average case](#) analysis, [time-space tradeoffs](#), and lower bounds.



## Sorting algorithm



*Figure 13: An example of Memory Maps. Picture 1 shows the concept to be grasped/learned. Picture 2 shows the automatic extraction of keywords from the document.*

# Chapter 4: Implementation and Results

## 4.1 Implementation of CMS

The content management portion of Intinno [24] has been completed and is already under production at <http://www.intinno.com> under the name of Learning Groups. Intinno system is currently under use in 50 courses of IIT Kharagpur.

The screenshot shows the Learning Groups website interface. At the top, there is a logo with a stylized 'i' in a square, followed by the text 'Learning Groups' in a cursive font, and the tagline '... the classroom++' to the right. Below the header, the main content is divided into two columns. The left column features a sub-header 'A web2.0 approach extending the limits of education beyond classrooms.' followed by three user roles: 'Instructors' (with an icon of a professor), 'Students' (with an icon of a student), and 'Administrators' (with an icon of a man in a suit). Each role has a brief description of their benefits. The right column contains a login form with fields for 'Login ID' and 'Password', a 'Login' button, and links for 'New User?' and 'Forgot Password?'. Below the login form are two buttons: 'Browse' and 'About Us', each with a corresponding text label.

**i Learning Groups** ... the classroom++

A web2.0 approach extending the limits of education beyond classrooms.

**I**nstructors  
Simple and Intuitive way to manage your courses online.  
Save hours every week.

**S**tudents  
Enhance the way you learn.  
Resources at your fingertips; Save your time scavenging for them elsewhere.

**A**dministrators  
Efficiently manage and control your educational resources.  
Get up and running in less than 5 minutes.

Login ID   
Password   
**Login**

[New User ?](#)  
[Forgot Password ?](#)

Have a look around **Browse**

Know more about Intinno **About Us**



- [My Courses](#)
- [Notifications](#)
- [Favorites](#)
- [View Profile](#)
- [Edit Profile](#)

### My Courses

S.No.	Name	Created on	Role
1.	<a href="#">Demo Course</a>	08 Jan 02:28	Student (C
2.	<a href="#">Machine Learning : CS60050</a>	17 Jan 12:33	Student (El
3.	<a href="#">OS &amp; Networks Lab</a>	29 Feb 19:52	Teach
4.	<a href="#">PDS</a>	05 Feb 00:18	Teach
5.	<a href="#">Programming and Data Structures - Section 6</a>	08 Jan 02:45	Ta
6.	<a href="#">Test Course</a>	08 Jan 12:19	Student (C

### Notifications

S.No.	Notification	Course	T
1.	<a href="#">+ Lecture LDA added.</a>	<a href="#">Machine Learning : CS60050</a>	24 A
2.	<a href="#">+ Lecture PCA added.</a>	<a href="#">Machine Learning : CS60050</a>	24 A
3.	<a href="#">+ Lecture Support vector Machines added.</a>	<a href="#">Machine Learning : CS60050</a>	20 A
4.	<a href="#">+ Lecture Conditional Random Fields added.</a>	<a href="#">Machine Learning : CS60050</a>	20 A
5.	<a href="#">+ Lecture Maximum Entropy added.</a>	<a href="#">Machine Learning : CS60050</a>	20 A




### Programming and Data Structures - Section 6

- [Overview](#)
- [Syllabus](#)
- [Assignments](#)
- [Resources](#)
- [Announcements](#)
- [Groups](#)
- [GradeSheets](#)
- [Forum](#)
- [Members](#)

### Assignments

S. No.	Title	Deadline	Given On	Discussion	Edit
1.	<a href="#">Demo Assignment</a>	15 Jan 00:00	08 Jan 15:13	<a href="#">Discuss</a>	<a href="#">Edit</a>
2.	<a href="#">Assignment 1</a>	29 Jan 00:00	29 Jan 15:16	<a href="#">Discuss</a>	<a href="#">Edit</a>
3.	<a href="#">Assignment 2</a>	29 Jan 00:00	29 Jan 14:56	<a href="#">Discuss</a>	<a href="#">Edit</a>
4.	<a href="#">Assignment 3</a>	01 Feb 00:00	30 Jan 07:48	<a href="#">Discuss</a>	<a href="#">Edit</a>
5.	<a href="#">Assignment 4</a>	12 Feb 00:00	05 Feb 15:18	<a href="#">Discuss</a>	<a href="#">Edit</a>
6.	<a href="#">Assignment 5</a>	19 Feb 00:00	11 Mar 15:04	<a href="#">Discuss</a>	<a href="#">Edit</a>
7.	<a href="#">Lab Test 1</a>	19 Feb 00:00	19 Feb 14:06	<a href="#">Discuss</a>	<a href="#">Edit</a>
8.	<a href="#">Assignment 6</a>	05 Mar 00:00	11 Mar 15:05	<a href="#">Discuss</a>	<a href="#">Edit</a>
9.	<a href="#">Assignment 7</a>	18 Mar 00:00	11 Mar 15:21	<a href="#">Discuss</a>	<a href="#">Edit</a>
10.	<a href="#">Assignment 8</a>	18 Mar 00:00	18 Mar 14:25	<a href="#">Discuss</a>	<a href="#">Edit</a>
11.	<a href="#">Assignment 9</a>	01 Apr 00:00	25 Mar 14:45	<a href="#">Discuss</a>	<a href="#">Edit</a>
12.	<a href="#">Assignment 10</a>	15 Apr 00:00	01 Apr 15:13	<a href="#">Discuss</a>	<a href="#">Edit</a>
13.	<a href="#">Lab Test 2</a>	08 Apr 00:00	08 Apr 14:00	<a href="#">Discuss</a>	<a href="#">Edit</a>

[New Assignment](#)



**Programming and Data Structures - Section 6**

- Overview
- Syllabus**
- Assignments
- Resources
- Announcements
- Groups
- GradeSheets
- Forum
- Members

## Syllabus


**Laboratory Component** : To be conducted on a 3-hour slot. It will be conducted in tandem with the theory course so the topics given in the lab are already initiated in the theory class. The topics taught in the theory course should be appropriately sequenced for synchronization with the laboratory. A sample sequence of topics and lab classes for the topic are given below :

- 1.Familiarization of a computer and the environment and execution of sample programs
- 2.Expression evaluation
- 3.Conditionals and branching
- 4.Iteration
- 5.Functions
- 6.Recursion
- 7.Arrays
- 8.Structures
- 9.Linkd lists
- 10.Data structures

**Theory Component** : Introduction to the Digital Computer ; Introduction to Programming ♦ Variables, Assignment; Expressions; Input/Output; Conditionals and Branching; Iteration; Functions; Recursion; Arrays; Introduction to Pointers; Structures; Introduction to Data-Procedure Encapsulation; Dynamic allocation; Linked structures; Introduction to Data Structure ♦ Stacks and Queues; Search Time and space requirements. (A programming language like C/C++ may be used as a basis language. The same language must be used in the laboratory).

[Edit Syllabus](#)

Home | Profile | My Courses | All Courses | **Feedback** | Favorite it | **INTINNO** | Welcome uidt | Settings | Log out



**Programming and Data Structures - Section 6**

- Overview
- Syllabus**
- Assignments
- Resources
- Announcements
- Groups
- GradeSheets
- Forum
- Members
- Settings**

### Edit Course

#### Privacy Settings

<input checked="" type="radio"/> Public	Course contents are open to anyone.
<input type="radio"/> Private	Course contents are open to the members of this course only.

[Save Settings](#)

#### Enrollment Settings

<input checked="" type="radio"/> Public Enrollment	Any user is allowed to enroll.
<input type="radio"/> Request Based	Users will submit request for Enrollment. Those approved by Teachers/Tas will be allowed to enroll.
<input type="radio"/> Invitation Based	Only the users invited by Teacher/Ta(s) will be allowed to enroll.
<input type="radio"/> Password Based	Any user having the password will be allowed to enroll

[Save Settings](#)

#### Functionalities For This Course

Widgets for this course:

*Figure 14: Snapshots of the Intinno*

## References

- [1] J. Cole and H. Foster, *Using Moodle: Teaching with the Popular Open Source Course Management System* (O'Reilly Media Inc., 2007).
- [2] V. B. Devedzic, Key Issues in Next Generation Web Based Education, *IEEE Trans. System Man Cybernetic: Part C*, Vol. 33, pp. 339 (2003).
- [3] LCMS: <http://www.learningcircuits.org/2005/nov2005/carliner.htm>
- [4] Understanding E-Learning 2.0: <http://www.learningcircuits.org/2007/0707karrer.html>
- [5] Joseph Psocka, Sharon A. Mutter (1988). *Intelligent Tutoring Systems: Lessons Learned*. Lawrence Erlbaum Associates.
- [6] Hammouda, K. and Kamel, M., "Data Mining in e-Learning", in Samuel Pierre (ed.) "E-Learning Networked Environments and Architectures: A Knowledge Processing perspective", series: Advanced Information and Knowledge Processing, Springer Book Series, 2007.
- [7] Scheuer, O. & McLaren, B.M. (2008). Helping Teachers Handle the Flood of Data in Online Student Discussions. In the *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*.
- [8] Hage, H.; Aimeru, E., "ICE: A System for Identification of Conflicts in Exams," *Computer Systems and Applications, 2006. IEEE International Conference on.* , vol., no., pp. 980-987, March 8, 2006
- [9] SCORM: <http://en.wikipedia.org/wiki/SCORM>
- [10] S. Chakrabarti, *Mining the Web: Discovering Knowledge from Hypertext Data* (Morgan-Kauffman, 2002).



- [11] S. Chakrabarti, M.H. Van den Berg, and B.E. Dom, "Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery," *Computer Networks*, vol. 31, nos. 11–16, pp. 1623–1640.
- [12] MIT Open Courseware: <http://ocw.mit.edu/OcwWeb/web/home/home/index.htm>
- [13] NPTEL India: <http://nptel.iitm.ac.in>
- [14] V.G.Vinod Vydiswaran and Sunita Sarawagi, *Learning to extract information from large websites using sequential models*(In COMAD, 2005. SIGKDD Explorations. Volume 6, Issue 2 - Page 66)
- [15] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th International Conference on Machine Learning (ICML-2001)*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001
- [16] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- [17] “Web surpasses one billion documents: Inktomi/NEC press release.” available at <http://www.inktomi.com>, Jan 18 2000.
- [18] A. McCallum, K. Nigam, J. Rennie, and K. Seymore, “Building domain-specific search engines with machine learning techniques,” in *Proc. AAAI Spring Symposium on Intelligent Agents in Cyberspace*, 1999.
- [19] M. Diligenti et al., "Focused Crawling Using Context Graphs," *Proc. 26th Int'l Conf. Very Large Data Bases (VLDB 2000)*, Morgan Kaufmann, 2000, pp. 527–534.
- [20] Lawrence R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77(2), pages 257–286, February 1989.

- [21] Hodgson, J. (2001). Do HTML Tags Flag Semantic Content? *IEEE Internet Computing*, Vol(5), pp. 25.
- [22] Hsin-Chang Yang, [Chung-Hong Lee](#): Building Topic Maps Using a Text Mining Approach. [ISMIS 2003](#): 307-314.
- [23] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, 1997.
- [24] Intinno: <http://www.intinno.com>
- [25] Hammouda, K., & Kamel, M. (2006), "Data mining in e-learning". In Samuel Pierre (Ed.), *E-learning networked environments and architectures: A knowledge processing perspective*, Springer Book Series: Advanced information and knowledge processing (pp. 1 – 28).
- [26] K., S., "Reducing the space requirement of suffix trees." *Software — Practice and Experience* 29 (1999) 1149–1171
- [27] N. Kushmerick, Gleaning the Web, *IEEE Intelligent Systems*, Vol. 14, 20 (1999).