

ABSTRACT

DAMM, KEVIN B. Incorporating Student Note-Taking into Online Computer-Assisted Instruction. (Under the direction of Associate Professor Robert St. Amant).

Computers have become an invaluable asset to students and instructors in the educational process. The increase in availability of computers and network technology in the classroom has led to further use of these technologies by both instructors and students. Instructors provide lecture materials through the World Wide Web and students often refer to these materials during study and when completing assignments [38].

Computers have also been used to assist students during the learning process. Research into Computer Assisted Instruction and Intelligent Tutoring Systems has developed dramatically in the decades since it began, and has been shown to be effective in encouraging the learning process and produce higher achievement [20]. However, further improvements and additions can be made to these systems by integrating them with the existing workflow of instructors and students.

I present a Computer Assisted Instruction system that utilizes an interface for creating and presenting lectures as well as making notes on course material. By developing this system as a web application, it can be accessible anywhere from the World Wide Web (WWW) at any time during the semester. I discuss the design decisions and implementation details of the web application interface and system components.

Novel to this system is its analysis of the student notes using Natural Language Processing and Information Retrieval techniques to determine the most similar lecture item. I then explain how this could potentially benefit Intelligent Computer Assisted Instruction systems that use a Bayesian Student Model for assessing the student's knowledge. Bayesian Student Models have been shown to be effective in knowledge assessment in other CAI systems and my proposed system offers additional parameters from which they can be built. I then evaluate this method to show its utility in an ICAI system.

Incorporating Student Note-Taking into Online Computer-Assisted Instruction

by
Kevin Blake Damm

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Computer Science

Raleigh, North Carolina

2008

APPROVED BY:

Dr. Christopher Healey

Dr. Donald L. Bitzer

Dr. Robert St. Amant
Chair of Advisory Committee

Dr. James C. Lester

Dedication

This thesis is dedicated to my mother, Donna Damm,
as well as everyone who has encouraged others to continue learning.

Biography

Kevin Damm was born and raised on Long Island, NY, where he first learned to use and program a computer. He later moved to North Carolina and obtained a Bachelor of Science from North Carolina State University. He is pursuing an offer of employment as a software engineer for Google in California.

Acknowledgements

I would like to thank my committee chair Dr. Robert St. Amant as well as Dr. James Lester for his feedback and encouragement. I would also like to thank my other committee members, Dr. Donald L. Bitzer and Dr. Christopher Healey.

I would like to thank my friends, family, and students who have helped encourage me throughout this process.

Table of Contents

List of Figures	vi
1 Introduction	1
1.1 Usage Scenarios	2
1.2 Design and Implementation Challenges	4
1.3 Contributions	6
2 Background & Rationale	8
2.1 Student Note-Taking	8
2.2 Computers in Instruction and Learning	9
3 Design & Implementation	12
3.1 System Description	13
3.2 Data Management	14
3.3 User Interface	16
3.3.1 Author Interface	16
3.3.2 Instructor Interface	17
3.3.3 Student Interface	19
4 Analysis of Student Notes	23
4.1 Evaluation	26
5 Conclusion	29
Bibliography	31

List of Figures

Figure 2.1 An example architecture for an Intelligent Computer Assisted Instruction system.....	10
Figure 3.1 The architecture for the proposed system.....	13
Figure 3.2 The database schema for the web application’s persistent data store	14
Figure 3.3 The overview page for the instructor and student interfaces.....	18
Figure 3.4 An example of the lecture list interface	19
Figure 3.5 A single slide of the presentation can display text, code examples, mathematical formulae and images.....	20
Figure 3.6 The calendar view displays the events for each month, including lecture dates, assignment deadlines and exams	21
Figure 3.7 The student can also search for key terms throughout the course lectures and notes.....	22
Figure 4.1 Comparison metrics and the prediction of rank	27

Chapter 1

Introduction

With the increased availability and affordability of computers in the classroom, and the prevalence of network connections on many University campuses, there is increased motivation for including computers and internet applications in pedagogical practice. Furthermore, the advent of the World Wide Web and its explosive growth provides a convenient and common interface for data format and content delivery.

Prior to the World Wide Web and the widespread adoption of software for browsing this network, the creation of a Computer Assisted Instruction framework was a tremendous undertaking, requiring the setup and maintenance of a particular hardware and software environment; today, web applications provide a delivery system for educational content that can more easily be integrated into a classroom environment [24].

This has led to the development of several WWW based educational environments [4, 5, 23]. Many of these environments take advantage of traditional Intelligent Computer Assisted Instruction techniques, especially Student Modeling. By modeling what the student knows, and by corollary what the student does not know, an educational application can determine which topics may be covered next and what questions may be asked to further assess the student's knowledge.

Many University courses are currently taught without any advanced CAI techniques, yet still make use of computers and the WWW to teach their class. At NC State University, nearly every Computer Science class has a web presence for distributing the syllabus and other lecture materials. Indeed, a course website is provided for every Computer Science class each semester. Often, the instructor will provide a broad outline of topics that will be covered during the semester and lecture slides for each lecture. Students may print

these materials out, annotate them and use them for review [38].

Studies have shown that when students have access to the instructor's notes in addition to their own notes, they perform better than if either of these materials were not available [17]. It follows, then, that students would benefit from having convenient access to notes and lectures through a uniform interface and instructors would benefit from a similarly available interface for creating and distributing lectures and notes.

Traditional approaches to ICAI have not provided a convenient way for students to take notes during a lecture, nor have they incorporated these notes into the assessment of a student's knowledge. A robust method for determining which lecture topics are relevant with respect to the notes a student has written would provide an ICAI system additional data about a student's knowledge, rather than building a student model entirely from question and answer sessions as in traditional Intelligent Tutoring Systems. Furthermore, incorporating this technology into the workflow of a traditional classroom could lead to an increase in the use of ICAI systems in University education.

1.1 Usage Scenarios

I will introduce some personas to illustrate several hypothetical events surrounding the use of a system such as the one I propose:

Ms. P is a professor teaching courses regularly at her university who has a course outline she developed years ago and follows rather closely each semester, with minor modifications. Mr. G is a new faculty member who has taught a few classes at another university before; he has been asked to teach a multi-section introductory course that had previously been taught by Ms. P.

Two students at the university, Mrs. A and Mr. B, are each registered in one of Mr. G's classes. Mrs. A typically takes copious notes and will spend some time reviewing those notes when she finds a concept difficult to grasp, but does not always choose to look at her notes. Mr. B does not usually like to take notes, but finds that he has benefitted from reading notes when studying for an exam.

Ms. P begins the semester's planning by consulting the academic calendar and entering notable events such as homework assignment dates and exam dates into the course calendar. Having already created many of the lecture presentations for the course, she only has to assign dates for when each one will be given.

Mr. G begins by looking at some of the previous lecture presentations made by Ms. P, but decides he wants to create his own lecture for the next class. He adds slides to the lecture, with some incorporating graphics and a few source code examples. He does not have to consider the visual layout for any of these slides, as this is handled by the rendering of the presentation. When he is done, he previews the entire slideshow and breaks one slide into two when he notices it is a little long. He asks Ms. P to look at the finished presentation and she suggests attaching notes to some of the image-based slides so they may be explained when viewed by the students outside of the lecture.

Mrs. A and Mr. B both use this system during class to take notes. Although Mr. B does not usually take notes during class, he likes to be able to view the lecture on his computer at his own pace, and he occasionally looks at the notes made public by other students in the class. Mrs. A takes notes during class in case she needs to review them for an exam or a homework assignment. She shares the notes she writes by making them public and disclosing herself as the author.

During the lecture, Mr. B follows along with the instructor's presentation, and when he sees a concept he doesn't remember clearly, he searches for it in the previous lectures to refresh his memory. After reviewing it, he writes a short note that summarizes the concept in very few words, with the intention of easily finding the concept in the future.

As Mr. G is giving the lecture, one of the students asks a question that raises an interesting point about the current presentation slide. Mr. G decides to write a note on this point using the lecture system so students can refer to it in the future, and so Mr. G can review it later for revising the presentation or use it in an assignment.

After class, Mr. B checks the calendar in the system to plan and prioritize the work he needs to do for his classes. He notices that there is an exam scheduled at the end of the next week and sends a message to Mrs. A because he has seen that she takes thorough notes for the class. The two of them, along with some fellow students, meet and use the notes each of them took earlier to share their understanding of the course topics. Mr. B uses each of the notes he's written to recall which concepts may be difficult to remember clearly, and Mrs. A uses her notes as a rewriting of the basic concepts and any material the instructor discussed but did not include in the presentation. Another member of the group printed the lecture slides out for greater portability and takes notes on the printed version during the study session.

1.2 Design and Implementation Challenges

When designing such a system, the expected use cases must be considered and the platform in which it is to be distributed and executed may also affect design decisions. Because the software is being used in an educational environment, there are regulations which impose certain accessibility requirements on the interface. There are various security issues to consider, as well, due to potentially personal information being stored and exchanged.

There are four basic roles of users for any Computer Assisted Instruction system: students, instructors, authors and system administrators [24]. The student role is unambiguous from the above example usage, and the administrative role deals mainly with the maintenance of hardware and system resources. There is a subtle distinction between author and instructor; the author role involves the creation of learning resources and the instructor role is involved with facilitating the presentation of this content. Often, both roles are held by the same individual, though there are obvious cases where a person would have only one of these roles.

My system handles these different roles through a role based authentication process whereby each user account has one or more roles associated with it, and actions are restricted to being available to certain roles. In some cases, the interface itself will vary dependent on which role or roles the user has. Because it is possible that the user has different roles for each of many classes, these role-dependent properties are additionally dependent on which course the relevant resource is associated with, and the user's role is defined for each course they are involved with.

Security considerations also involve guarding against malicious use of the system. Any web application that has a public interface is at risk against attacks such as SQL injection and Cross-Site Scripting, the result of which may be wrongly modified or missing data. These attacks are well documented, but implementing defenses requires awareness of the techniques used to avoid leaving available the vulnerabilities these attacks prey upon.

The accessibility of content to all users is also important. There have been many national and international software usability standards that require educational software be accessible to users with disabilities. Because the application is being made available to users across the World Wide Web through browser software, the efforts of the W3C and private interests have provided assistive technologies such as voice browsers to address the need of many users with disabilities [31]. The application itself needs to be aware of the appropriate

formats to use for these assistive technologies to transform the web page's content.

Beyond accessibility for users with disabilities, there is the issue of normalizing behavior across different browsers. Although each browser has adopted a large amount of functionality from standards or competitive innovations, the implementations vary slightly in many cases, because there is no reference implementation for browser software. This leaves corner cases where rendering of HTML or execution of JavaScript result in different, and sometimes broken, states in one or more browsers. The application developer needs to be careful to either avoid these corner cases or develop special routines to handle the desired behavior for a particular browser. Fortunately, there are several JavaScript libraries available for overcoming differences in JavaScript implementations, and the documentation available for web development eases the work required to provide a page structure suitable for all browsers. Testing on multiple platforms with various browsers is tedious but necessary.

To some extent, the responsiveness of the application to user actions is important. A delayed response can leave the user frustrated or disappointed in the application and avoid its use. To address this, the application developer can use a recently adopted technique of browser scripting which allows JavaScript to make a request for data without reloading the page. With this technique, the browser can continue presenting the current information, provide an indication that the user's action is being processed, and render the results of the server's response, all without disrupting the user's perceived workflow.

Unfortunately, older browsers may not have this capability or may have the scripting engine disabled. Although nearly all browsers in current use have this capability, there are a number of users who disable scripting for security reasons, which can also render this technique unavailable. It is possible to develop for both cases, where asynchronous data transfer is available or unavailable, through a technique called "graceful degradation" or "progressive enhancement," whereby the page is originally presented in its most basic form and, if JavaScript is enabled, the additional functionality is added after the page has been downloaded.

Progressive enhancement of the page requires that the structure of each page is developed separately for the basic and enhanced versions, and the server side similarly needs to have separate input processing and output generation for each version. This development overhead is only necessary when building the fundamental elements of the system's interface; the users of the system will only have to provide a single version of their content and the

data only needs a single representation, so the eventual benefit outweighs this initial cost.

In designing the system as content that has HTML markup added for layout and scripting added for progressive enhancement, the notes and lectures can be simply defined as the core content, and metadata easily provided. This allows instructors, authors and students to edit text without needing to know HTML or any web development languages, allowing them to focus on the content.

1.3 Contributions

I propose that student notes offer a useful collection of information that can be analyzed and classified with respect to which lecture or lecture slide a student was potentially referencing in their notes.

To evaluate this proposition, I developed an integrated note-taking and lecture presentation web application which allows for the authoring of lecture presentations and the recording of notes, as well as auxiliary tools to assist in the student's and instructor's workflow. An additional benefit is the machine-readability of the resulting data, allowing for processing which can aid in the assessment of student knowledge and reporting of overall class activity.

Although there exist systems which carry out parts of this process separately, such as commercial off-the shelf presentation creation, note-taking applications, course material storage through Content Management Systems, and student assessment through Intelligent Tutoring Systems. there is no current system that coordinates each of these aspects. I present the design of one such comprehensive system.

Following the design description, I offer a preliminary evaluation of a pivotal factor in its performance, namely the use of classical Information Retrieval techniques to compare a student note to each available lecture slide or entire lecture presentation. The evaluation provides evidence that student notes and lecture presentations are a unique context for Information Retrieval, and more robust algorithms for determining relevance of student notes is needed.

Provided with a reliable method for associating student notes to lecture slides, I also give an example construction of a directed acyclic graph automatically generated as a result of actions that professors and students currently perform. This can be used as an additional component in the student model for an Intelligent Tutoring System or other

similar Intelligent Computer Assisted Instruction system.

Chapter 2

Background & Rationale

2.1 Student Note-Taking

Note-taking is a common and important practice [37] and according to several studies it has been shown to correlate with better performance on exams [17, 18, 28, 34].

When asked, students report that they take notes because they want to do well in class [25]. Some students report that taking notes helps them maintain attention during class, and most students admit that they take notes because the resulting product helps them during review for an exam [19, 25].

Studies are equivocal about whether the process of taking notes is itself beneficial to the student or correlates to better exam scores [14, 19], but most studies show a strong correlation between reviewing notes and better recall of material during an exam [19]. In light of this, any educational technology being offered to college students should give them the ability to record and review notes.

Few Computer Assisted Instruction systems offer this capability, however. One system, StuPad, offers students the ability to take notes and attach them to recorded audio and video streams of lectures, as part of Classroom 2000 [35]. The system is currently being redesigned to better impact student behavior and performance [2], but the authors have not indicated any plans to analyze notes as part of an assessment of student knowledge.

In the past two years, several web-based note-taking applications have been developed [12, 27, 26] in response to browser technology becoming more mature and the apparent demand for managing the tremendous amount of information being made available through the WWW. None of these systems have any connection to Computer Assisted Instruction

technology, however.

In recent years there has also been an application of Content Management Systems (CMS) from business use to the area of course instruction. Some of these systems, often called Learning Management Systems, allow for the recording of notes, but none have a Student Model incorporated into them, nor do they do any analysis of the notes being stored.

Student notes are unlike the typical documents considered in Information Retrieval and Natural Language Processing, in part because the length of student notes range from very few words to entire paragraphs. Fortunately, student notes are typically directed by a constrained set of topics related to the lectures an instructor presents. However, the limited set of words used still contains ambiguities and anaphora that make a thorough analysis difficult. This is discussed further in chapter 4.

2.2 Computers in Instruction and Learning

Computer Assisted Instruction systems have existed as early as 1961 [3]. In the decades since, many excellent implementations of CAI systems have been developed [5, 8, 9, 13, 23, 30, 33, 39]. These systems have gone under the names Computer Assisted Instruction, Computer Aided Learning, Intelligent Tutoring Systems, eLearning Systems, Web-Based Education, Educational Hypermedia, and others.

Soon after the earliest use of computers in education, the practice was referred to as Computer Assisted Instruction in the US and Computer Aided Learning in Europe. As developments in AI, multimedia, hypermedia and communications technologies improved, projects were started in order to take advantage of these advancements in the context of CAI and CAL.

A noteworthy improvement in these systems was the addition of a Student Model which represents the knowledge and learning process of the student based on an analysis of the student's behavior, especially in terms of the correctness of answers given by students during quiz sessions. The system can use this model to adapt the instruction to the student's needs. Systems that make use of a student model to determine which questions should be asked, are called Intelligent Tutoring Systems (ITS), or generally, Intelligent Computer Assisted Instruction.

Another improvement distinctive of Intelligent Computer Assisted Instruction is

the inclusion of an expert system containing facts and rules regarding the domain being learned. This expert system allows for a much larger set of questions that may be asked of the student, as these questions can be formulated rather than directly authored. There is still a moderate amount of work involved in tracing "paths" through the knowledge content of the expert system, but many toolkits have been developed to assist this process [1]. Some of these ICAI systems also store facts and rules about various learning strategies of students so that the system can adapt its presentation of the material and questions to better suit the learner. An architecture for such a system, taken from [21], can be found in figure 2.1.

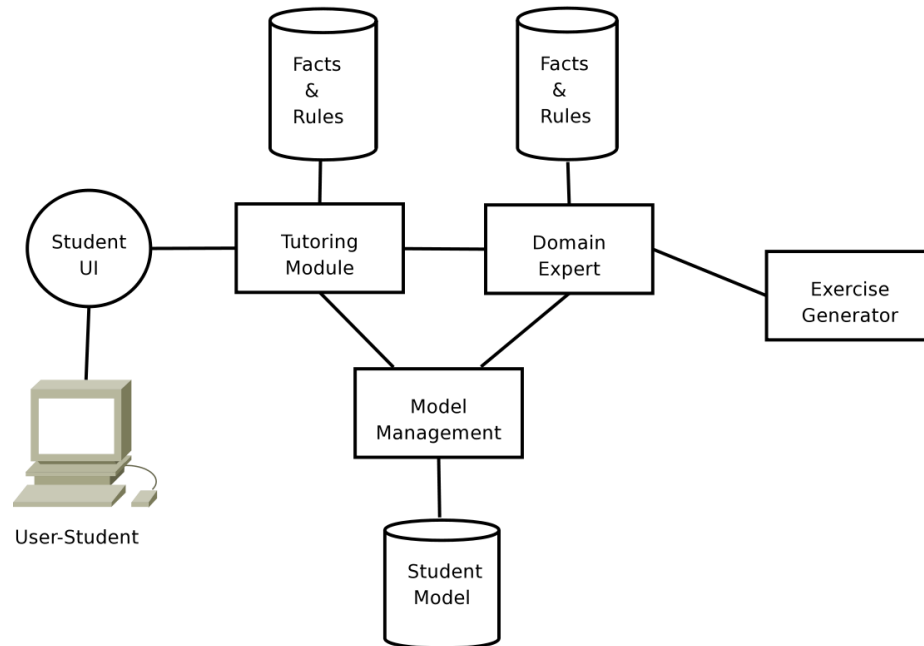


Figure 2.1: **An example architecture for an Intelligent Computer Assisted Instruction system**

Advances in hypermedia and network communications have led to another improvement in Computer Assisted Instruction – that of providing instructional resources through the World Wide Web [30, 6]. These systems, called eLearning systems or Web-Based Education systems, take advantage of the increasing availability of the WWW to make course-related resources available to students from any internet-connected computer at any time of the day. This has had far-reaching impact on Distance Education and has

in some ways decreased the burden on instructors and teaching assistants.

The sophistication of these systems ranges from simply storing course material to presenting an interface tailored especially to the student using it. Many of the less sophisticated of these systems are based on the Content Management Systems popular in business environments and have been termed Learning Management Systems (LMS). Although they offer a convenient interface for storing and retrieving class materials, they do not inherently have any learning-related functionality.

An aspect of the World Wide Web, and more specifically hypertext, the predominant format for documents, is the nature of linking the contents of one document to the location of another WWW document. This simple concept has created an entire area of research in Hypermedia. Combining this idea with that of a user model gives rise to Adaptive Hypermedia, wherein hypertext links are shown or hidden (adaptive navigation) or their color/size changed (adaptive presentation) based on user preferences determined from the user's previous navigation behavior. When the user model is replaced or supplemented with a student model, as in Adaptive Educational Hypermedia [4], the student is presented with course material appropriate to their level of knowledge and they avoid potentially getting lost or confused in the face of too much learning material [7]. This is especially valuable in student-directed learning, as well as in the context of University courses.

Research in this area has also explored the use of dialogue in Intelligent Computer Assisted Instruction systems [36, 13]. This may allow for more direct observation of the student's knowledge, especially if semantic and pragmatic aspects of the student's writings are interpreted.

While all of these systems certainly have their benefits, they are typically used outside of the traditional lecture context of University courses. Furthermore, there are activities in the normal workflow of this lecture environment, namely student note-taking, that produce artifacts which such Intelligent Computer Assisted Instruction systems could use to further assess knowledge and learning of the student.

Chapter 3

Design & Implementation

By designing this system as a web application, the application developer can take advantage of the familiarity most students have with finding course material on the WWW. Many university instructors currently provide a web site for retrieving lecture presentations and students have become accustomed to looking on the web for questions regarding the course work.

Typically the instructor will create a slideshow presentation through off-the-shelf software such as PowerPoint or L^AT_EX, and put the resulting .ppt or .pdf files on the course website [38]. My aim was to provide the author with an interface that is familiar while using hooks in the underlying system to allow for Natural Language Processing of the resulting content. To achieve this goal, I developed the system as a web application that the author, instructor and student can interface with using their choice of web browser.

One major benefit of providing an application through the World Wide Web is that users typically do not need to install any applications to access it. Updates to this software are applied automatically as soon as the user visits the page, because the program logic is stored, and distributed by, a server platform. Likewise, updates to content are applied as soon as the user requests the relevant data, so errata information can be directly published.

Although the user does need to have a web browser installed, modern operating systems (including Windows variants, Mac OS X and various Linux distributions) all include a web browser. The typical computer user is familiar with using a web browser, and the interface elements provided through them are consistent across most web pages. Furthermore, layout and rendering is managed by the browser, so most development can be

focused on the functionality of the application.

3.1 System Description

In terms of the system presented in figure 2.1, the Student UI has been expanded to provide an interface to the lecture slides and student notes for each type of user, as depicted in figure 3.1. The shaded items are those sections of the system described in this thesis:

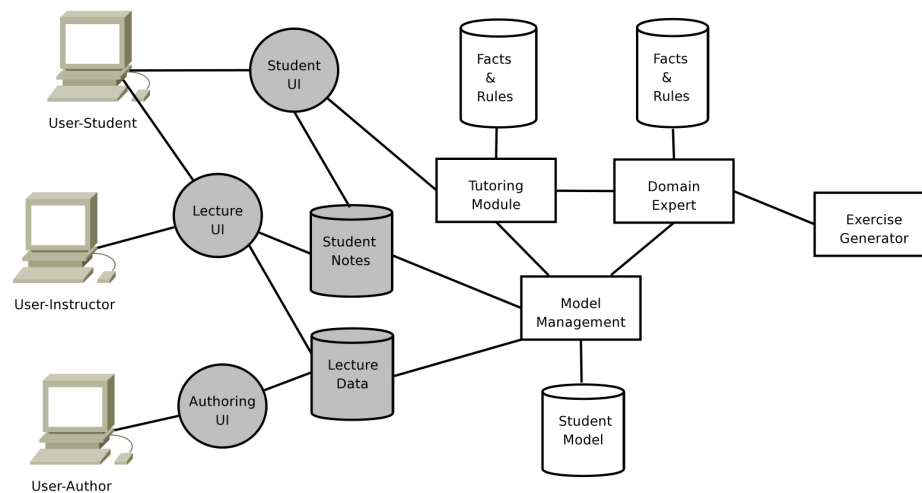


Figure 3.1: **The architecture for the proposed system**

The system is composed of three tiers: a database tier for persistent storage of lectures and notes, a Server tier for generating the dynamic webpages which serve this content, and the user/browser-agent tier for rendering the hypertext content being generated. This has been shown to scale well to a large number of users in client-server applications [10].

Other aspects of the system, such as the facts and rules of the expert system or the pedagogical planner, do not need to be altered to interface with this system. The student model can be supplemented with additional model data, as discussed in chapter 4.

3.2 Data Management

A relational database is used to persistently store the application data, allowing fast retrieval of indexed data and the ability to store relationships between items. The tables for this data are in fourth normal form both to eliminate redundancy and for the maintenance and performance benefits [16], were the system required to scale to multiple database machines (e.g. in the case of a very large student population). Figure 3.2 shows an Entity-Relationship model for the database schema used in this system.

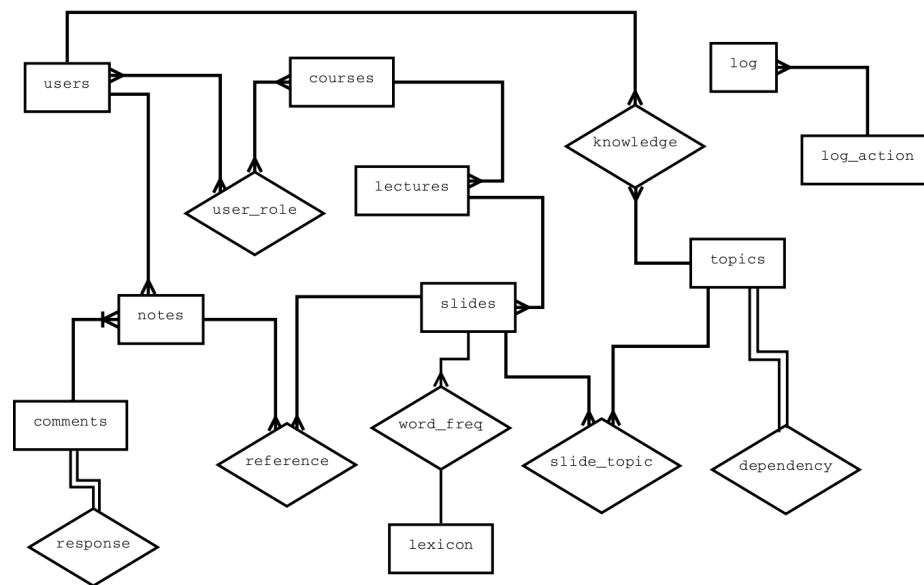


Figure 3.2: The database schema for the web application’s persistent data store

The entities were chosen to represent the main elements of a note-taking system, such as notes, lecture slides, lectures and courses. Relationships were chosen to provide flexible queries into the data set. Some entities and relationships, such as the slide_topic and lexicon tables, are generated whenever data in another table is created or updated, but most tables are populated directly from user input.

A user may have more than one role or be in more than one course, so user roles have a many-to-many relationship with rows in the users table and rows in the courses table. The users table includes contact information and authentication information, while courses contain descriptive information. Courses could be further normalized into departments and colleges, semesters and sections, if the need arose. Currently this information is stored as a

description column in the courses table.

The reference table between notes and slides is constructed based on the content of the student notes, as described in chapter 4. If a note's relevance to a slide exceeds a specific threshold, the similarity is recorded in the reference table for immediate retrieval at a later time. Following this computation, if the current user is a student in the course for the lecture in question, the topics which are related to the slide are discovered and the knowledge relationship for the student user is updated to reflect this.

The topics table is initially populated by creating a nameless topic associated with each lecture slide. Each lecture slide is enumerated, with a special 'zeroth' slide to represent the entire lecture. Because the slide_topic table is keyed to each slide and topic pair, a topic is also associated with each entire lecture presentation. The dependency relation has two one-to-one keys to build a hierarchical organization of topics. Thus, each time a slide is created, an appropriate topic dependency is created from the entire lecture to the new slide, and the resulting topic structure is implicit from the lecture structure.

The lexicon contains all words appearing in either lecture slides or notes, along with values for metrics such as the number of documents it occurs in and the number of notes it occurs in. Additionally, a word_freq table keeps track of the number of times each word appears in each slide. By updating these values whenever a lecture slide or note is created or edited, the entire corpus of lectures and notes does not have to be parsed when performing analysis on student notes; this is especially important if the collection grows to a very large amount of data.

This database also stores information about the user activity, including when the user logs in and out of the system, when they view a lecture or any individual slide, and when they create and edit notes. The instructor might find it useful to analyze this data to determine which topics the students have been returning to for review.

The database system, along with the interfacing code, implement transactions to avoid misalignment of data. Additionally, each row of every table has a unique primary key determined by insertion sequence, so rearrangement of slide ordering or renaming of other identifying properties do not disrupt the relationships represented in other tables.

3.3 User Interface

The interface is presented as a web application through a web browser such as Microsoft's Internet Explorer or the Mozilla Foundation's Firefox browser. Thus, most of the interface elements are readily available and well-tested, and my description of the user interface will be in terms of these elements.

Web browsers also afford convenient transmission of data from the client to the server through forms. A form tag in an HTML page has well defined semantics for the presentation of input collection and the packaging of the input data in a common format. This reduces a lot of development effort on the part of the application developer.

Popular web browsers also include a scripting environment for runtime parsing and modification of the Document Object Model (DOM) that represents the page the user is viewing. Recent advancement in browser technology also includes the ability to post and request data to/from the server of the original page, without reloading the page currently being displayed.

Through this technique, data transmission can be limited to the representation of individual objects, eliminating the redundant framing content of each web page, thus decreasing the bandwidth required for data transfer and increasing the perceived responsiveness of the web application. The server needs to be aware of whether the content should have markup added or not, but this can be handled with a separate parameter provided with the input.

My application takes advantage of these aspects of browser technology to provide a reliable, responsive interface to its users. As mentioned in chapter 1.1, the main classes of user are Author, Instructor and Student. There is also an administrator interface, but it does not differ significantly from that of other Intelligent Computer Assisted Instruction systems.

3.3.1 Author Interface

The author's main responsibility is to create lecture content in the form of slideshow presentations. These presentations are given by the instructor during a lecture, but can also be viewed by any user outside of the classroom.

The authoring interface provides a means for creating new lectures and assigning metadata such as the date it will be presented and summary information about the lecture's

contents.

For editing the lecture, a form-based interface is provided to edit each slide, and the author may also rearrange the ordering of the slides in the slideshow. The form for each slide includes input fields for the title and slide content, where the slide content may consist of bulleted points, paragraphs, source code, or image data. The type of content is indicated by a simplified markup scheme similar to what is found on Wikipedia and other WWW content creation interfaces. Often, the markup resembles the resulting content type, such as asterisks at the beginning of a line for bulleted lists, or wrapping source code excerpts in `{ { {` and `}} }`. Text in the slide may also contain markup in a subset of latex suitable for displaying math-environment formulas if placed between `$$` symbols.

This set of representations is a large subset of what is available in commercial presentation software such as Microsoft PowerPoint. Many lecturers are already familiar with this software for creating slideshows for presentations. Those features which are not included in my application, such as animations or sound, are seldom used in practice and nevertheless may be emulated with, for example, animated GIFs or a link to an audio file.

The author may also preview the slideshow, using the same interface provided for the student, to ensure that its appearance matches what is expected. Although commercial off-the-shelf presentation applications typically allow the user to place slide contents anywhere within the visible area, this application automatically formats the data based on its contents, centering and scaling as necessary, with scrollbars added to source code blocks if they go beyond a certain length. This allows the author to focus more on the content of the slides and worry less about precise placement of this content.

3.3.2 Instructor Interface

The instructor's main responsibility is to present the lecture and elaborate on each slide's content. Prior to each lecture, the instructor may alter the date each lecture is associated with, view the contents of the lecture slideshow, and provide additional notes to accompany a slide. These notes are not displayed during the slideshow presentation, but can be accessed outside of the presentation viewing interface and are used when analyzing the similarity of notes to slides or lectures.

After authenticating, the instructor is first greeted with the overview page for the course, shown in figure 3.3. The user's focus is drawn to the center of the page where

a description of the system and the next scheduled lecture appears. If any messages or comments have been left for the current user, information about them will also appear here.

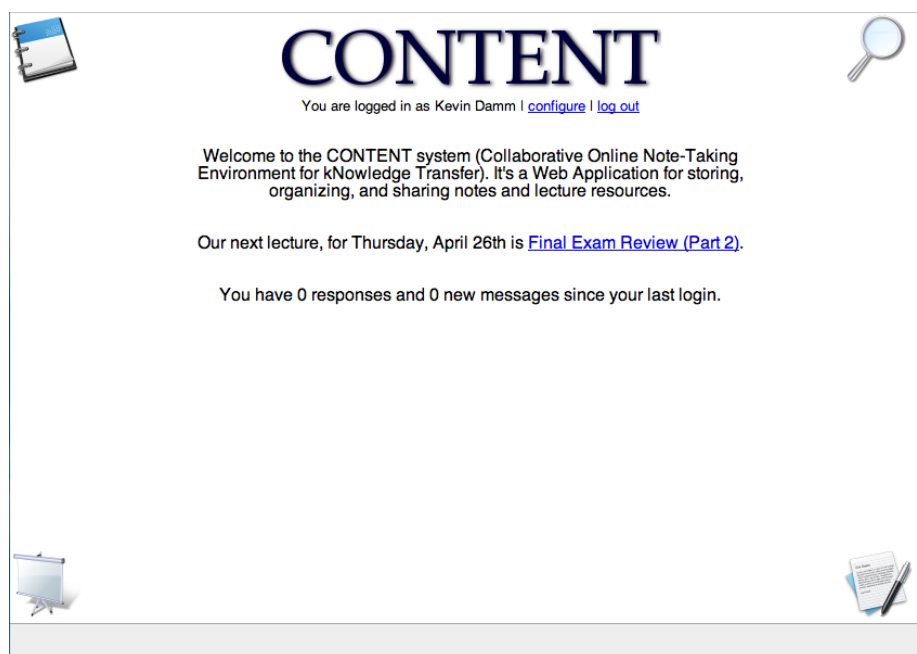


Figure 3.3: The overview page for the instructor and student interfaces

Each corner of this page contains a link to one of the main views into the lecture and note-taking system. The bottom-left corner gives access to the lecture list (see figure 3.4) where the user can select from a list of the available lectures, ordered chronologically. The top-left corner links to the calendar view (see figure 3.6), the top-right corner links to a search interface, and the bottom-right corner links to the collection of visible student notes. As shown in the figures, these icons maintain their location and appearance so navigation is intuitive.

The instructor can create or modify events, such as assignment deadlines or exam dates, through a form in the calendar view. These events will then appear on the appropriate dates when the student is browsing their calendar. For assignments and exams, the instructor can provide a URL for the appropriate web page or downloadable file giving further details about the requirements.

The instructor may also view the notes students have written recently, sorted by

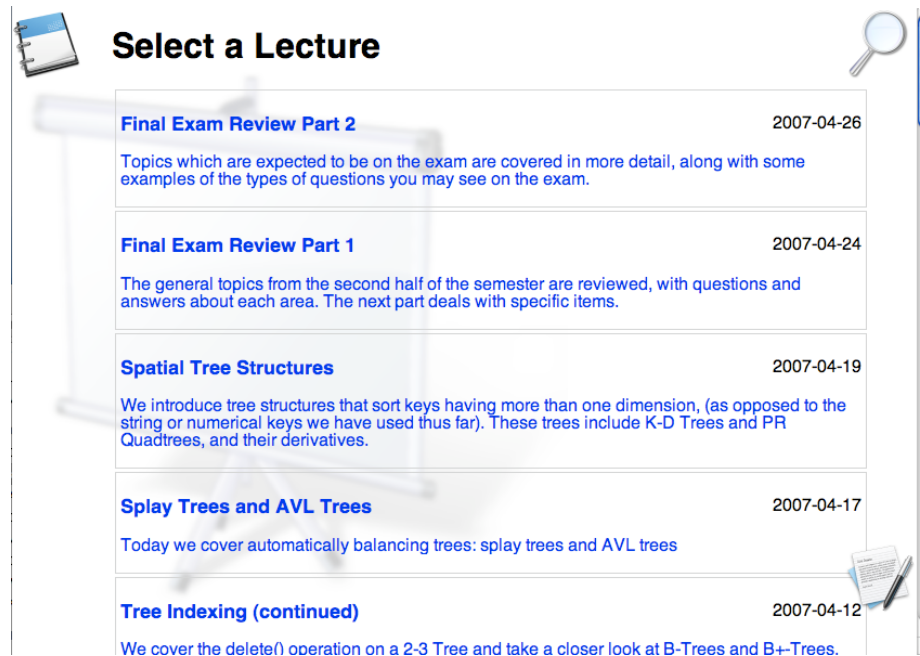


Figure 3.4: An example of the lecture list interface

student or date. Messaging functionality is built into the system so that the instructor or other students can leave comments on or respond to notes and comments. This functionality is modeled after forums and message boards that have become a popular method of communication in many universities.

When a slideshow is being presented, the instructor can click on the appropriate navigation links in the top-right corner to move forwards or backwards in the presentation.

If the instructor wants to add a note to the slide during the presentation, there is a link for creating a note that will add to the instructor notes for that slide. These notes are not visible during the presentation, but will appear when printing the lecture presentation or when looking at the collection of all notes for a presentation.

3.3.3 Student Interface

The student interface opens with the same overview page that greets the instructor (see figure 3.3), including a link to the next scheduled lecture and information about any messages that have been posted to the student.

The student may view a list of available lectures (figure 3.4) through the icon in

the top-left corner, which resembles a presentation easel. The role-based security for the system prevents the user from editing the summary or date information, but the student can select any of these lectures to begin viewing the slideshow presentation starting with the title slide.

BST findNode() [take note](#) «(4 / 13)»

The following is an implementation of how to find an element in a binary search tree

```
function findNode(BinNode n, Element e) {
  if (e.equals(n.element()))
    return n;
  else if (e.compareTo(n.element()) < 0) {
    if (n.left() != null)
      return findNode(n.left(), e);
    else return null;
  } else { // e > n.element()
    if (n.right() != null)
      return findNode(n.right(), e);
    else return null;
  }
}
```

Find the growth rate of this function in terms of the number of nodes in the binary search tree

SOME RIGHTS RESERVED Kevin Damm, CSC 316 002 [Spring, 2007] [Show Table of Contents](#)

Figure 3.5: A single slide of the presentation can display text, code examples, mathematical formulae and images

Figure 3.5 shows an example slide from such a presentation. Layout of the table and content of each slide is provided automatically by the interface, and the familiar action of clicking the mouse anywhere in the slide moves the slideshow forward one position. There is also a link in the top-right that allows the student to take a note without leaving the slideshow. The asynchronous transfer of data allows this note to be submitted without interrupting the flow of the slideshow.

If the lecture is currently being given, viewing the slideshow connects the student's interface to the pace of the instructor's presentation by automatically progressing the slideshow forward when the instructor moves to the next slide, provided the student is currently viewing that slide. The student may still choose to navigate the slideshow at their own pace by using the navigation links in the upper-right corner; visiting the slide

currently being presented puts them back in sync with the instructor.

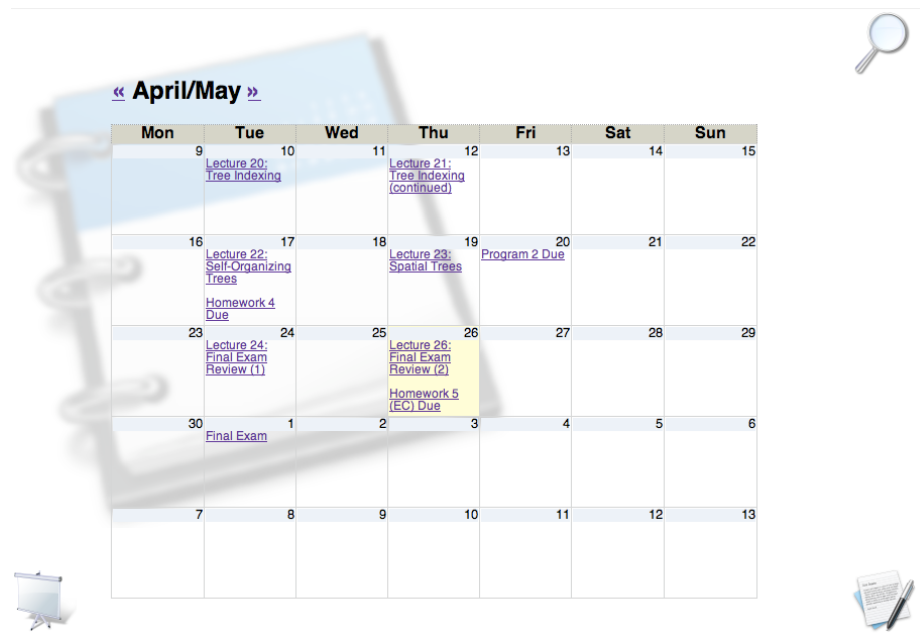


Figure 3.6: The calendar view displays the events for each month, including lecture dates, assignment deadlines and exams

Because time management is an important aspect of university study, a calendar view of course-related events is available (see figure 3.6). Clicking on the links provided for lectures, assignments and exams will take the student to the lecture slideshow, assignment description and requirements, or topics covered on the exam, as the case may be.

Students may view notes they have already made, with comments from other students shown in a threaded view. The student may choose to hide any of these comments, and the system keeps track of which items are visible and hidden so on subsequent visits the student sees only the notes and comments they are interested in.



Figure 3.7: The student can also search for key terms throughout the course lectures and notes

Chapter 4

Analysis of Student Notes

The notes that students take vary greatly in length, from a few words to a few hundred, and yet the largest note items are often much shorter than documents typically encountered in Information Retrieval. It is well known that classic Information Retrieval techniques can determine relevance of a document to a user query, but that longer documents have a higher chance of being judged relevant [32].

Student notes are unlike traditional documents in other ways; they often do not include complete sentences and even the sentence fragments need not be grammatically correct. Students often write notes with the sole intended audience being themselves, and in many cases only a few words are necessary to elicit the appropriate memory.

Student notes may also contain various types of structural information; on paper they will draw figures and diagrams, and even with computer-based input many students create visual depictions out of deliberately arranged characters. This is not easily analyzed by Natural Language Processing techniques.

Fortunately, these notes are being compared to lecture presentations where similar rules of small word count and incomplete sentences are the norm. On the other hand, students who know they can access the lecture slides at a later time may not write exactly what is indicated in the lecture presentation or slideshow and instead attempt to fill in the gaps based on the statements of the instructor or their own intuition and prior knowledge [25].

Some lectures are summary lectures, so they may contain representative terms that are repeated in other lectures or in more specific notes. In this case, it may be hard to determine whether a student's note that shares only one term with both of these lectures is more appropriate for the summary slide or the focused lecture. In some cases it may be

appropriate to indicate student comprehension of both instances.

Students may also take notes from text or online materials outside of lectures, though studies have shown that this occurs far less often [37]. In that case, the notes may still have relevance to one or more lecture slides, even if they were not directly taken in reference to that slide or slides.

There has not yet been an empirical study of the content of student notes. While a full analysis of all aspects of student notes would be outside the scope of this thesis, it may prove worthwhile to investigate the performance of classic Information Retrieval techniques on a collection of student notes and lecture slides in order to determine what value these techniques may have in future work.

Whenever a lecture slide is saved into the database, its contents are processed and a reverse index is updated. This index stores the term frequency for each word per slide and per lecture, as well as storing the number of individual slides it appears in. This information is used when comparing student notes to lecture slides. Not every note needs to be matched to a lecture slide; if it matches closely enough, however, it is a strong indicator that the student now knows the indicated topic.

When a student enters a note, its content is similarly tokenized and the note is compared to the collection of lecture slides. There are a variety of different similarity metrics developed in the field of Information Retrieval that are useful [22]. Of these, the vector-based measures TF-IDF and Cosine Similarity have proven useful in practice for information retrieval and text mining. TF-IDF is a statistical weight determined by term frequency (tf) and inverse document frequency. Here they are defined as:

$$\text{tf}(i, D_j) = \frac{n_{i,D_j}}{|D_j|}$$

$$\text{idf}(i) = \log\left(\frac{|D|}{o_i}\right)$$

where n_{i,D_j} is the frequency of word i appearing in document j , $|D_j|$ is the word count of document j , $|D|$ is the number of documents, and o_i is the number of documents in which term i occurs.

The product of these two values is the TF-IDF weight for the pair of term i and document j . TF-IDF benefits from increasing the weight of query terms that are infrequently found in documents and decreasing the weight of more common terms. A

query of multiple words can be evaluated by taking the sum of these weights for each term in the query:

$$\text{tfidf}(Q, D_j) = \sum_{q \in Q} \text{tf}(q, D_j) * \text{idf}(q)$$

This method does not take into consideration the distribution of terms across various queries, as the queries are typically considered independent of each other. The coslne similarity metric, typically useful for comparing documents to other documents, can be used [15]:

$$\rho(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \bullet \mathbf{B}}{|\mathbf{A}| |\mathbf{B}|} = \cos(\mathbf{A}, \mathbf{B})$$

where $\mathbf{A}$ and $\mathbf{B}$ are term vectors composed of weights corresponding to each term contained in documents A and B , respectively. Thus, $\rho(\mathbf{A}, \mathbf{B})$ corresponds to the angular distance of the two vectors, and $0 \leq \rho \leq 1$. The computation is simplified by the fact that whenever a term has a zero weight in either $\mathbf{A}$ or $\mathbf{B}$, it need not be computed because the effective contribution for that term will be zero. From this, we can use the set of terms T composed of terms with nonzero weights in both $\mathbf{A}$ and $\mathbf{B}$, and compute $\rho(\mathbf{A}, \mathbf{B})$ as

$$\rho(\mathbf{A}, \mathbf{B}) = \frac{\sum_{t \in T} a_t * b_t}{\sqrt{\sum_{i \in A} (a_i)^2 * \sum_{j \in B} (b_j)^2}}$$

where a_t and b_t are the weights of term t in document A and B , respectively.

When comparing a student note to each lecture slide, or the entire contents of a lecture, maximizing the value from either of these metrics should indicate the slide or lecture which is the closest syntactic match for the note in question. However, some terms appear very often across many documents, such as ‘the,’ ‘and,’ ‘of,’ etc. If this word is the only shared term between documents, it may give misleading nonzero values for similarity between two documents. This is addressed by eliminating consideration of the following “stopwords:”

stopwords = [‘I’, ‘a’, ‘about’, ‘an’, ‘are’, ‘as’, ‘at’, ‘be’, ‘by’, ‘com’, ‘de’, ‘en’, ‘for’, ‘from’, ‘how’, ‘in’, ‘is’, ‘it’, ‘of’, ‘on’, ‘or’, ‘that’, ‘the’, ‘this’, ‘to’, ‘was’, ‘what’, ‘when’, ‘where’, ‘who’, ‘will’, ‘with’, ‘www’]

Notable words may appear in different slides with different contexts, due to ambiguities in meaning or being referenced by a compositional or dependent concept. Words from

alternative or contradictory concepts may be mentioned as a point of reference. To avoid wrongly classifying these slides as relevant to a student’s note, the Rocchio classification method can be used by computing weights for each term using the following formula [15]:

$$w'_i = \alpha w_i + \frac{\beta \sum_{j \in C_i} d_j}{n_i} - \frac{\gamma \sum_{k \notin C_i} d_k}{N - n_i}$$

This classification method is semisupervised, as opposed to the two previous metrics which can be completely unsupervised. However, we can take advantage of the lecture organization to automate discovery of categories. In this sense, any words that appear in multiple lectures can be disambiguated by finding words unique to each of the lectures sharing the term and using these as $k \notin C_i$ for each appropriate C_i . This exclusion property can be determined when the note is being analyzed.

The objective when finding the closest matching slide or lecture is to associate the note with the topic or topics which that lecture or slide represents. In this way, the student model can be supplemented with information regarding the topics which are exhibited in the student’s notes.

A popular representation for a student model is the overlay model, where the student’s knowledge is considered to be a subset of the knowledge of an expert in the field, and this knowledge is represented as a graph [11, 21]. Vertices in the graph represent concepts and edges represent a dependency for one concept onto another.

In this way, the knowledge covered in a course can be dependent on each lecture and each lecture dependent on the slides within it. This graph need not be joined directly with an existing student model. Rather, question types in an ITS could be dependent on vertices in the existing student model and vertices in the student model extracted from lecture content, separately.

4.1 Evaluation

This system was provided to a class of 15 students in a Data Structures course at North Carolina State University. These students submitted notes related to various lectures being given, and the notes were compared to each lecture slide and the entire text of each of 23 separate lectures.

The average word length of each lecture is 785.54 with 365 total slides across all lectures and an average of 53.4 words per slide. Of the 17 collections of student notes, there

are 3,969 words and 879 unique words, with an average of 37 words per note item.

Comparisons were made using the sum of tfidf weights and the cosine similarity metric described above. For both of these metrics, the words were indexed in their original form or, alternatively, the words were stemmed using the Porter stemmer [29], providing four possible comparison metrics.

To compare these metrics, each was used to determine which lectures matched closest to the note text and ranking them in a zero-based ordering. The results in figure 4.1 show that all four metrics perform rather well, frequently finding the appropriate lecture in the top five matches, when comparing note items of a student from an entire day to entire collections of lecture slides.

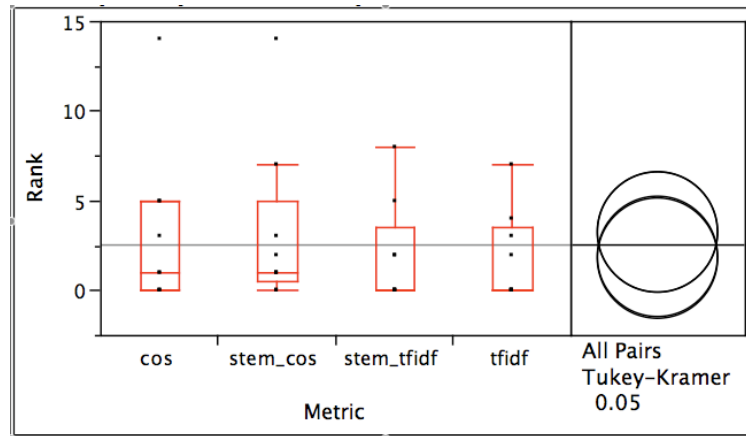


Figure 4.1: **Comparison metrics and the prediction of rank**

On closer inspection of the data, the outlying note with an associated lecture that was ranked 14 for cosine similarity had included back-references to previous lecture topics and many numerical values demonstrating an example that was given during class but not included in the lecture slides. The mean values for each metric are listed in table 4.1.

Table 4.1: Statistical results for each metric on entire notes

Metric	Mean	Standard Dev
tfidf	1.778	2.49
tfidf stemmed	1.889	2.85
cosine	3.222	4.52
cosine stemmed	3.222	4.58

When comparing individual notes items (determined as separated by empty lines in a single note) to individual slides, these classic IR techniques are less successful. Large weights were found to be given to pairings that did not match the expected lecture for each note.

Because this may be due to actual relevance to previous topics, each of 3,720 slide/note pairings were labeled according to how relevant the pairing is, along a scale where 0 denotes “no relevance” and 4 “completely relevant.” These labels were only performed by myself, so they are only anecdotal, but they give an idea of the difficulty of analyzing documents with such small word counts.

The results indicate that these relevance values are not significantly correlated to the weights returned by any of the above metrics. This may be often due to terms occurring in slides where they are merely being referenced as a point of comparison or to indicate alternative concepts, without focusing on that term. Student notes which mention this term are then given a strong weight for more slides than expected.

A more precise method of matching notes to lecture slides is needed, perhaps incorporating user feedback about relevance and conducting a semisupervised categorization, or through more advanced processing making use of contextual or semantic information.

Chapter 5

Conclusion

Involving the process of student note-taking and the overall workflow of University lectures into Computer Assisted Instruction and the World Wide Web allows for wider access to lecture and note material as well as providing the potential for automated processing of the

Intelligent Computer Assisted Instruction has great potential to assist students in the learning process, and its availability through the World Wide Web offers benefits in availability, accessibility and familiarity of its interface. One of the strongest features of Intelligent Computer Assisted Instruction is its Student Model which tailors instruction to each individual student.

My work provides a method of using the existing process of an instructor's lecture presentation creation and students' note-taking to create virtual artifacts that not only persist to assist the student but may also feed into the assessment of a student's knowledge.

If student does not take any notes, an Intelligent Computer Assisted Instruction system is no worse than if it did not incorporate student notes in the first place. As an alternative to note analysis, the artifacts produced via emailing the instructor or postings on message forums could also be used to assess the student's knowledge.

There are a lot of future directions this work could be taken. The comparison metrics are admittedly simple, and although they appear to work well, an evaluation at the semantic or pragmatic level may offer even better results. A deeper integration with existing techniques in semantic networks and knowledge representation could also lead to a more in-depth understanding of the student's knowledge, to the point of carrying a dialogue with the student as they study the course material. This approach of using dialogue in Intelligent

Tutoring Systems has been researched [36] but has not yet been integrated into the student note-taking process.

Another possible direction is that of using relevance feedback, either explicitly through the interface or implicitly via analysis of which notes and lectures the student returns to, or which comments the student hides from view. Relevance feedback within the context of Information Retrieval has been considered as early as 1971 [15], and could prove to be useful in the context of the appropriateness of student notes to lecture material.

Studies in Educational Psychology indicate that students who review their notes and the lectures perform better [17], so the information about the student's login frequency and lecture viewing could potentially be used in assessment. Unfortunately, simply knowing that the student has visited a web page does not necessarily indicate that the student has read or understood the content on that page. Writing text that relates to the lecture material seems to be a much clearer indicator.

Further work could be done on collecting a wider range of notes from a longer time period, as well as notes from various areas of study beyond Computer Science. The concept of collecting student writings and comparing them to a collection of documents may also be applied to areas outside the classroom or lecture environment, such as Knowledge Management.

With a more precise method of correlating student notes to relevant concepts in the lecture material, and the appropriate relationships to facts and rules in an ITS database, one could include the ITS questions directly into the appropriate lecture slides, so students who want to experiment with answering questions related to a certain subject can do so, encouraging exploratory learning.

That the relatively simple Information Retrieval techniques used in this system work at all is encouraging. It would be interesting to see if it remains as effective with a larger set of documents in the form of lecture presentations or if it is effective across different domains of study.

Bibliography

- [1] V. Aleven, J. Sewall, B. M. McLaren, and K. R. Koedinger. Rapid authoring of intelligent tutors for real-world and experimental use. In *Proceedings of the Sixth IEEE International Conference on Advanced Learning Technologies*, pages 847–851. IEEE Computer Society, 2006.
- [2] A. Bauer and K. Koedinger. Evaluating the effect of technology on note-taking and learning. *Computer-Human Interaction (CHI)*, April 2006.
- [3] D. L. Bitzer and Jr. J. A. Easley. Plato: A computer-controlled teaching system. In Sass & Wilkinson, editor, *Symposium on Computer Augmentation of Human Reasoning*, Washington, DC, 1965. Spartan Books.
- [4] Peter Brusilovsky. Methods and techniques of adaptive hypermedia. *User Modeling and User Adapted Interaction*, 6(2-3):87–129, 1996.
- [5] Peter Brusilovsky. Knowledgetree: A distributed architecture for adaptive e-learning. In *WWW*, 2004.
- [6] Peter Brusilovsky, John Eklund, and Elmar Schwarz. Web-based education for all: A tool for development adaptive courseware. In *Computer Networks and ISDN Systems*, volume 30, pages 291–300, 1998.
- [7] Peter Brusilovsky and Leonid Pesin. Adaptive navigation support in educational hypermedia: An evaluation of the isis-tutor. *Journal of Computing and Information Technology*, 1998.
- [8] H. Burns and C. Capps. *Foundations of Intelligent Tutoring Systems*. Lawrence Erlbaum Associates, 1988.

- [9] Alexandra Cristea and Toshio Okamoto. An adaptive distance learning environment for language teaching. In *Proceedings of the International Workshop on Advanced Learning Technologies*, 2000.
- [10] W. W. Eckerson. Three tier client/server architecture: Achieving scalability, performance, and efficiency in client server applications. *Open Information Systems*, 10(1), 1995.
- [11] C. Frasson and G. Gauthier, editors. *Intelligent Tutoring Systems: At the Crossroad of Artificial Intelligence*. Intellect Books, 1990.
- [12] Google. Google notebook. <http://www.google.com/notebook>.
- [13] A. C. Graesser, K. VanLehn, C. P. Rosé, P. W. Jordan, and D. Harter. Intelligent tutoring systems with conversational dialogue. *AI Magazine*, pages 39–51, Winter 2001.
- [14] James Hartely and Ivor K. Davies. Note-taking: A critical review. *PLET*, 18(3):207–224, 1978.
- [15] Jr. J. Rocchio. *Relevance Feedback in Information Retrieval*. Prentice Hall, 1971.
- [16] W. Kent. A simple guide to five normal forms in relational database theory. *Communications of the ACM*, 26(2):120–125, 1983.
- [17] K. A. Kiewra. Providing the instructor’s notes: An effective addition to student note-taking. *Educational Psychologist*, 20(1):33–39, 1985.
- [18] Kenneth A. Kiewra, Stephen L. Benton, Sung-Il Kim, and Nancy Risch. Effects of note-taking format and study technique on recall and relational performance. *Contemporary Educational Psychology*, 20:172–187, 1995.
- [19] Kenneth A. Kiewra and Nelson F. Dubois. Note-taking functions and techniques. *Journal of Educational Psychology*, 83(2):240–245, 1991.
- [20] J. A. Kulik and C. C. Kulik. Computer-based instruction: What 200 evaluations say. In *Annual Convention of the Association for Educational Communications and Technology*, Atlanta, GA, 1987.

- [21] R. Lelouche. The successive contributions of computers to education: A survey. *European Journal of Engineering Education*, 23(3):297–308, 1998.
- [22] M. Li, X. Chen, X. Li, B. Ma, and P. Vitányi. The similarity metric. In *The Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003.
- [23] P. Libbrecht, E. Melis, and C. Ullrich. The activemath learning environment system description. In *CalcuIemus Workshop at IJCAR*, pages 173–177, 2001.
- [24] M. R. Lyu, Y. S. Moon, W. K. Kan, and M. A. Vouk. Web-based education techniques: Workflow, collaboration, and quality of service. In *Proceeding of Quality in Teaching and Learning Conference*, 1998.
- [25] Peggy Van Meter, Linda Yokoi, and Michael Pressley. College students’ theory of note-taking derived from their perceptions of note-taking. *Journal of Educational Psychology*, 86(3):323–338, 1994.
- [26] MyNoteIT.com. Mynoteit. <http://mynoteit.com>.
- [27] NoteSake.com. Notesake. <http://notesake.com>.
- [28] Pauline A. Nye, Terence J. Crooks, Melanie Powley, and Gail Tripp. Student note-taking related to university examination performance. *Higher Education*, 13(1):85–97, 1984.
- [29] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
- [30] J. Prentzas, I. Hatzilygeroudis, C. Koutsojannis, and M. Rigou. Architectural aspects of a web-based its for teaching new information technologies. In *Proceedings of ED-MEDIA 2001*, 2001.
- [31] P. S. Reed, D. Gardner-Bonneau, and S. Isensee. Software accessibility standards and guidelines: Progress, current status, and future developments. *Universal Access in the Information Society*, 3(1):30–37, 2004.
- [32] A. Singhal, G. Salton, M. Mitra, and C. Buckley. Document length normalization. *Information Processing and Management*, 32(5), 1996.
- [33] D. Sleeman and J. S. Brown, editors. *Intelligent Tutoring Systems*. Academic Press, 1982.

- [34] Gary S. Thomas. Use of student notes and lecture summaries as study guides for recall. Reports - research, ED.GOV, 1975.
- [35] K. N. Truong and G. D. Abowd. Stupad: Integrating student notes with class lectures. *Computer-Human Interaction (CHI)*, 1999.
- [36] K. VanLehn. Student modeling. In M. Polson and M. Richardson, editors, *Foundations of Intelligent Tutoring Systems*. Lawrence Erlbaum Associates, 1988.
- [37] R. L. Williams and A. C. Eggert. Note-taking in college classes: Student patterns and instructional strategies. *The Journal of General Education*, 51(3):173–199, 2002.
- [38] M. A. Wirth. E-notes: Using electronic lecture notes to support active learning in computer science. *Inroads - The SIGCSE Bulletin*, 35(2), 2003.
- [39] Hai Zhuge and Yanyan Li. Active e-course for constructivist learning. In *In Proceedings WWW*, 2004.