An Automatic Tutoring System to Teach Microsoft Excel

A Thesis
SUBMITTED TO THE FACULTY OF
UNIVERSITY OF MINNESOTA
BY

VIDYA ATTIVILLI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Dr. Richard Maclin

August, 2015

Acknowledgements

I would like to thank my thesis advisor, Dr. Richard Maclin, for giving me an opportunity to work with him. I thank him for his continuous and valuable guidance throughout the development of this thesis. I would like to thank my thesis committee members, Dr. Pete Willemsen and Dr. Marshall Hampton for their patience and support. I would also like to thank the entire Computer Science faculty for supporting me throughout my Masters course.

I would like to thank my friends and family for their continuous love and support without which I would not be where I am now.

Abstract

The growth in the use of Automatic Tutoring systems has significantly increased over the past few years. New ideas and techniques are being used to make the automatic teaching as close as possible to human teaching. This project is designed to teach Microsoft Excel with the help of pre-recorded videos. The users are provided with a test designed in Excel that can be taken by the user to check how much the user has learnt. The main contribution of this thesis is the automatic test generation system. This system allows the user to take a test any number of times. It will create new tests based on templates which allow each generated test to have questions of the similar format and level of difficulty.

Table of Contents

List of Tables

List of Figures

1 Introduction

In the past 20 years, we have seen a rapid increase in the use of computers and the Internet. This proved to be an excellent platform for the growth of Intelligent Tutoring Systems [1]. These systems consist of programs used in learning and that can be considered somewhat intelligent. Hence, this field has a wide range of application in other areas such as natural language processing and social computing [2]. There are four building blocks for a tutoring system [3] – the domain model, the student model, the tutor model and a method for knowledge tracing. The domain model consists of the possible courses of actions that the user can take to solve a problem. The student model also consists of the steps to solve the given problem but is more closely related to the particular actions taken by the student. It also keeps track of the deviations (if any) from the domain model. The tutor model is a tracing algorithm that keeps track of the student's progress and is also responsible for deciding on which teaching strategies to use based on the student's performance. The knowledge-tracing algorithm evaluates the student's performance throughout the course of tutoring and gives feedback on the learning process of the student.

In this thesis work, we have implemented a simple automatic tutoring system that teaches Microsoft Excel. This is a web-based program that requires the user to register first and then login to the system. It presents the user with a set of lessons aimed at teaching the basics of Microsoft Excel. Each lesson is associated with a video explaining the concepts designed for that lesson, a simple Excel test and a PDF file that contains instructions on how to complete the test. After every test the system calculates a score and checks to see if it can push the user to the next lesson or if it should suggest the user retake the current lesson again. For any lesson, the student is allowed to retake the test any number of times. But, for this to be possible, we need to be generating tests that have questions of the same difficulty level to maintain the evaluation fairness of the system. The main contribution of this system is the automatic test generation system. It uses the concept of templates and generates tests dynamically consisting of equivalent questions.

Thesis Outline:

This thesis is organized as below:

- **Background Study**: This chapter explains the history of automatic tutoring systems and the different types of architectures used in building these systems. It introduces us to the Microsoft Excel tool and the Apache POI library, which have been used in this thesis.
- **Implementation**: This chapter contains the details of the system implemented in this thesis work. We explain the coursework that we have followed throughout the system, how we use templates to generate files to test the student and the evaluation procedure followed to assess the student's learning**.**
- **Results**: This section consists of some of the screenshots and results obtained as a result of this thesis.
- **Conclusion**: This section summarizes the work implemented in this thesis and also provides some interesting ideas for future research on this topic.

2 Background Study

2.1 Intelligent Tutoring Systems

From the Pressey Machine [4] to modern systems like Andes [5] and Atlas [6], Intelligent Tutoring Systems (ITSs) [7] [8] have come a long way on the road to being useful in education. The main goal of these systems is to provide an effective means of learning using computing techniques.

In the early years of ITS systems, around the 1960s, tutoring by machines involved a series of objective questions answered by the student. The systems maintained a threshold and if the student's score reached it, then the lesson was complete and they proceeded to the next stage. One such system was called Programmed Logic for Automated Teaching Operations (PLATO) [9][10] and this system used a programming language called Tutor, which was developed to make it easy to design new learning modules. At the basic level, a Tutor-generated problem had a question and each question had a pool of answers. Any one among these was considered as the correct answer by the system. A simple pattern matching technique was then used to match the user's response to the pool of answers and the results were graded. For example,

Question: *"What is the color of the sky?"*

Answer bank in database: *<the, color, of, sky, is, it> (blue,blu)*

This accepts answers like “The color of the sky is blue,” “it is blue,” “The color is blu.” or just “blue.” This pattern matching technique is at the heart of the PLATO system. It was around this time that research on essay grading systems started picking up along with a lot of progress in Natural Language Processing field [11]. By the 1990s, automatic tutoring systems started becoming more sophisticated and reliable. Automated essay grading systems like PEG [12] used linguistic features such as the frequency of occurrence of different words and punctuation, and used multiple regression analysis to find an equation that would score the essay. The goal was to find variables that would produce essay scores close to the human-graded essay scores. Another example is E-Rater [13]. In its basic version, E-Rater considers the syntax of an ungraded essay and compares it to others. If the syntax matches another essay with a high score, a high score is assigned to the ungraded essay. If the syntax is similar to an essay with a low score, the ungraded essay is then graded with a low score. By 2006, adaptive learning was incorporated into this system. This addition allowed the system to keep track of the student’s mistakes and helped score the essay based on the progression of the student’s learning. Furthermore, machine-learning algorithms were used to upgrade the understanding capability of the system. For example, the E-Rater system dropped colloquial terms and considered a bag-of-words feature set of the essay in question [14]. Bag-of-words refers to the set of words, along with the frequency of their occurrences without considering the order in which they are presented in the essay. It also used other complicated feature sets like grammar, style, lexical complexity, semantic meaning between the words, etc.

2.1.1. Architectures

According to Murray, T [15], ITS architectures can be categorized into six types, namely,

Curriculum sequencing and planning.

Systems designed with this architecture categorize and organize the curriculum into different modules, each one covering different topics. The modules are interrelated to each other and consist of instructions that are generally in the form of text or graphics. Student-system interaction is through objective types of questions such as multiple choice and fill-in-the-blanks. Depending on the student's performance, the next module in the sequence is dynamically determined. Due to the use of minimalistic modes of instructions used in this type of architecture, the domain knowledge is not represented extensively and feedback is moderate too.

Tutoring strategies.

This type of architecture is similar to curriculum sequencing and planning in terms of the way instructions are presented to the student and use of shallow domain knowledge representations. However, apart from sequencing the information modules dynamically, tutoring systems with this type of architecture tend to make micro-level decisions that is, based on the students' responses, the system decides an appropriate tutoring strategy for that scenario. Decisions such as when and how to give hints, explanations or examples are also dynamically. Examples of tutoring systems with this architecture are GTE [16] and Redeem [17].

Device simulation and equipment training.

In device simulation and equipment training, the equipment that it is designed to teach is simulated along with basic instructions. Simulation is difficult part of designing tutoring systems with this type of architecture. Designing a tutoring module to ask questions or give explanation is fairly easier. Tutoring systems with this kind of architecture are also used widely for various purposes and hence they should be generic in design. It supports applications that have basic simulation models as well the ones with rigorous models that teach the inner workings of a device. This architecture’s advantage over the previous two architectures is that it provides working knowledge.

Expert systems and cognitive tutors.

Another type of ITS architecture is the Cognitive Tutor. Tutors with this type of architecture are based on a set of rules. Depending on the student’s responses, the tutor builds a cognitive model of the student’s knowledge base. This model is then compared with an expert model. The deviations from the expert model allow the tutor to give appropriate and timely feedbacks to the students. The expert model helps the tutor to give the necessary hints whenever the student asks and also favors automatic problem solving. One of the best examples of this model is the Cognitive Tutor Algebra course, which was a commercial hit. Yet, these types of systems have their own limitations. For example, it can be designed only for a few domains like geometry and algebra that have an extensive set of rules. They do not provide any flexibility to personalize the system according to the learning process of the student.

Multiple knowledge types.

For different kinds of domain knowledge representations, we have different kinds of tutoring strategies. For example, a strategy designed to teach facts differs from a strategy designed to teach concepts. The architecture contains this pre-defined information relating to different knowledge types and the strategies used to tutor each type. Each strategy has a different way of presenting the instructions, giving feedbacks or hints and sequencing the concepts. Templates are provided for each knowledge type that can be filled in by the author. The architecture also provides tools to link various components of the templates that are interrelated. Although the templates cannot accommodate all kinds of information or knowledge types, they still have many applications.

Special purpose systems.

Some tutoring systems are designed to work with a specific task or domain. These come under the category of special purpose systems. Initially, a tutoring system is selected and it is then modified to focus on a particular task. Once these modifications are done, the system then follows a fixed path to solve only the problem that it was designed for.

We also have web-based tutors which follow a client-server learning environment. In this case, the student is considered to be on the client side while the teacher is on the server side. In [18], Goran Shimic and Vladan Devedzic discuss Code Tutor, which follows the architecture in Figure 1.

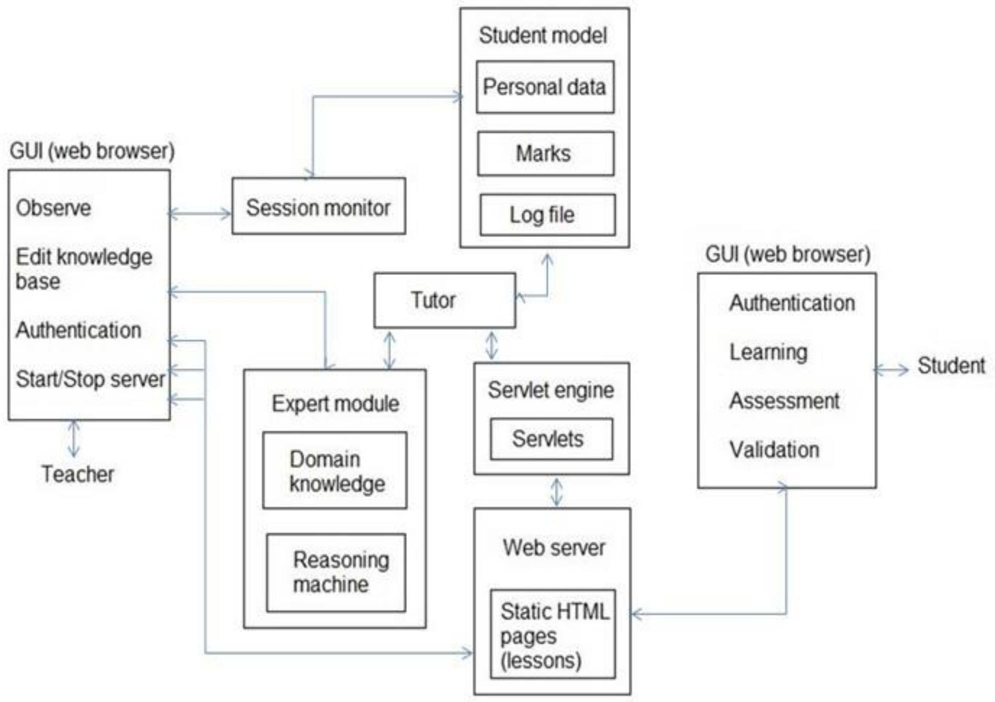


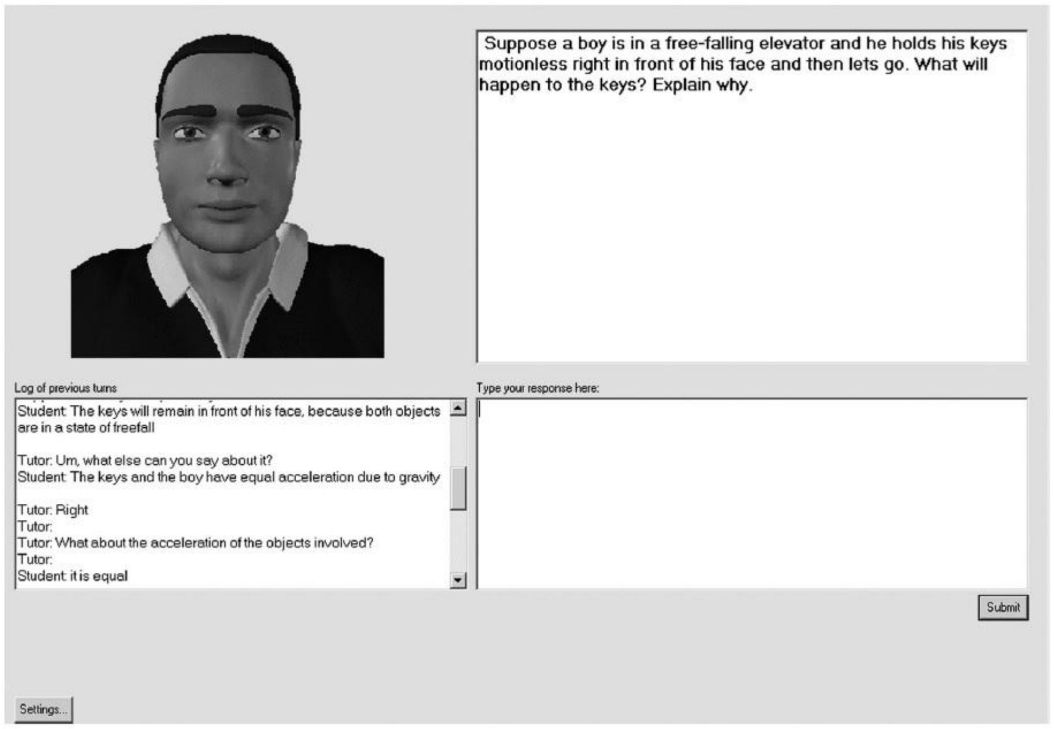
Figure 1 : Architecture of Code Tutor.

Code Tutor uses a standard web browser for student-system interaction and Java Servlet technology to generate and serve HTML pages dynamically. The student first logs into the system. From here, he has access to a series of lessons designed to teach a particular topic. Once the student selects a lesson, he can view the instruction in the form of text, visual graphics and media content. The tutor also provides student assessment by providing test after each lesson. Once the test is submitted, it is graded and even a single wrong answer will require the student to repeat the lesson again. Since it is web-based it can be distributed and accessed easily.

2.1.2. Auto-Tutor

Today, Intelligent Tutoring Systems are able to engage the students in a conversation while helping them learn various concepts. Auto-Tutor [19] is one such system, which helps students learn physics, critical thinking and computer concepts with a natural language dialogue. Its interface consists of an animated tutor that can speak and make gestures. It allows the student to participate in a conversation with the tutor with the help of the keyboard and also displays the entire dialogue. This tutor is interesting because it works with open-ended questions that need reasoning and the answer is slowly gathered from the student through a conversation between him and the tutor. It starts by posing a question that requires a fair amount of explanation (around 7-10 lines). Each question has a set of expectations and anticipated misconceptions associated with it. As defined in the paper, expectations are correct answers and misconceptions are misunderstandings that often occur. Each one of them has a set of keywords associated with them, which define them. As the student answers the question over a series of turns, the tutor guides the student by giving hints and feedbacks while constructing an answer. This continues until all the expectations are met and all the misconceptions are explained.

The most interesting part of the tutor is the dialogue management system. When the student initially answers a question, the system analyzes the response and forms an estimate of the student's knowledge. The system uses latent semantic analysis (LSA) to measure the similarity between the response and the expectations that have not yet been covered.


Suppose a boy is in a free-falling elevator and he holds his keys motionless right in front of his face and then lets go. What will happen to the keys? Explain why.

Student: The keys will remain in front of his face, because both objects are in a state of freefall

Tutor: Um, what else can you say about it?
Student: The keys and the boy have equal acceleration due to gravity

Tutor: Right
Tutor:
Tutor: What about the acceleration of the objects involved?
Tutor:
Student: it is equal

Figure 2 : Screenshot of the graphical user interface of Auto-Tutor.

Auto-Tutor compares each expectation with the student's response, calculates the LSA score and starts with the one, which has the highest score. This process is repeated continuously for every response given by the student. In this way, we can follow a systematic approach to the final answer in such a way that the student understands every concept one step at a time and in an orderly fashion. Once an expectation is chosen to be focused upon, the dialogue management system comes into play. Here the system uses the state of the student response and the expectation that is currently focused to generate its next response. If the student asks a questions like “Why?” or “Can you repeat the question again?” the system answers to them appropriately. In this case, “meeting an expectation” means that the student’s response consists of all the keywords associated with that particular expectation. The Auto-Tutor’s dialogue management system has been represented as a finite state transition network. The goal states are the nodes and the arcs represent the tutor’s course of action. Suppose, we want to meet an expectation *E*, the goal state would be the expectation *E*. According to the conditions that are met, the tutor takes the respective arc (prompts, hints or pumps) to reach the node. If it is the student’s first time dealing with that particular expectation, then the tutor opts to pump the answer out of him. If the student fails to hit one of the keywords in the keywords set for *E*, then the tutor prompts. In case, the student fails to hit most of the keywords of the set, then tutor provides him with a hint that can help him cover the expectation.

2.1.3 Effectiveness of Tutoring Systems

The main goal of designing tutoring system is to achieve effectiveness equivalent to human tutoring. VanLehn [20] provides us with a detailed report on the effectiveness of tutoring systems versus human tutoring. Human tutoring, here, is generalized as an adult teaching a single student. Tutoring systems, on the other hand, can be divided into two categories. Systems belonging to the first category give immediate feedback or hints when the student asks for it. Using these systems, the students might have to use a piece of paper to perform the necessary calculations in order to answer the questions. The second category consist sof tutoring systems that are dialogue based or have an interface with which they can communicate with the student. They pose a question and prefer to pump the answer out of the student. So instead of the using paper for calculations, these systems tend to allow the student to interact with them while calculating on a digital canvas and also provide immediate feedbacks and hints at every step.

VanLehn [20] has compared and contrasted human tutoring and tutoring systems. It would seem that human tutors can infer a better model of the student's knowledge base and hence they would be able to tend to the needs of the students better. But, as [21][22][23] point out, human tutors are not aware of all the misconceptions and false beliefs that the students might have. It is also difficult for the human tutor to analyze the students' skills on the spot. It requires a set of rules and condition statements to correctly diagnose a skills related problem. With the help of a series of questions, tutoring systems can easily find them out. With the human tutors, questions are rarely asked. On the other hand, human tutors can easily find out the strong points of the tutee. This behavior is common with the tutoring systems as well, which brings human tutoring and tutoring systems to the same level. One might also argue that human tutors are more effective when it comes to selecting tasks for the individual student to help him learn. However, research tells us that the human tutors usually follow a curriculum based on which they assign tasks ranging from an easy level to a difficult one [23] [24] [25]. Since tutoring systems have the advantage of having the knowledge of the students' misconceptions, they are able to exhibit same behavior and also use methods that can improve the task selection. Another arguable point is that students are free to ask a human tutor any sort of questions relating to a wide range of domains. The human tutor, in turn, can answer the questions and in some cases, use irony to point out the absurdness of a question or an answer. However studies suggest that the number of students who ask questions out of the domain is very low [25] [26] [27] and human tutors rarely use irony [26]. Human tutors can provide additional information like the origin and evolution of a certain principle or even their presence can motivate their students to learn. Such characteristics of human tutoring cannot be shown in tutoring systems but research shows that these traits do not directly affect the learning process of the student [25] [26] [28]. The two areas, which make the human tutors superior to tutoring systems, are the ability to give prompt feedback and scaffold. Human tutors can provide scaffolding in case the student is stuck and needs a push to finish the reasoning. They can also point out the flaws in the student's understanding even as the student explains a concept. The granularity of the feedback and scaffolding is what makes the human tutoring so effective. Achieving the same in tutoring systems is difficult and yet attempts like Auto-Tutor [19] take us one step closer.

While having a 1:1 student-teacher ratio can be costly and practically difficult, mass classroom instructions have failed to pace the learning process according to the students' learning ability. This is where Intelligent Tutoring Systems can help tremendously. Although Tutoring Systems were initially developed for classroom based or a simple concept based training, they have advanced to such a level that we now have tutoring systems like SHERLOCK that can train Air Force personnel on how to diagnose problems with electric circuits of F-15 jets, CARDIAC that can help medical personnel learn on cardiac problems and how to intervene and CODES that can teaches music.

2.1.4. The Future of ITS

Intelligent Tutoring Systems can model student’s knowledge and accordingly select a strategy to tutor them. To improve the effectiveness of these systems, Samuel Alexander et al [29] proposes a new kind of systems called Affective Tutoring Systems (ATS). These systems are designed using a series of embedded systems that can detect gestures, moods and various other bio-signals to utilize this information to select a tutoring strategy. The system is built upon a detection system that can detect and analyze facial expressions and gestures. A simulated model called Eve is used to carry out the tutoring process. In this system, when the student responds to a particular question, the facial expressions and gestures are detected by the detection system and a data file is generated. The student’s response along with the information from the data file is used to update the student’s history. This history is then used by a case-based method to generate a set of actions recommended to Eve. Based on a weighted average, Eve selects one of the actions and performs it. This action is recorded into the student’s history and the system waits for the student’s next response.

2.2 Microsoft EXCEL

Microsoft Excel is an application design by Microsoft for Windows and Mac operating systems. It consists of spreadsheets made up of a grid of cells. Each cell is identified by its numbered row and alphabetized column. With these cells containing the data, performing various arithmetic operations becomes easy, graphical representation of the data is possible which allows us to view the data statistically and also features like replicating and auto-filling some of the data helps us to save time on duplicating data. Excel also allows you to arrange the rows and columns according to our desired criteria.

In recent years, Excel has been used for a vast number of purposes like managing budgets and other financial records, keeping track of a customer database, creating reports and so on. This thesis work is mainly focusing on assisting students of Chemistry department to learn the Microsoft Excel tool. It has been further researched that some of the main purposes for which Excel is used by the Chemistry department is to keep a record of information like concentration amounts of the chemicals, add new data (rows and columns) to the existing data, perform basic operations like formula based calculations on the entered data, sort the rows and columns according to a given criteria and graphically represent the data along with the results. [30] Provides us with an analysis on where students usually go wrong while working with Excel. According to the authors, Tanja Reinhardt and Nelishia Pillay, formulae and functions are two of the most error-prone topics. They found that 82% of the students made an error while creating a formula with absolute cell references, 87% made mistakes while using an IF statement with the functions and 93% failed while using financial functions.

In order to save time and resources of teaching the use of such an important, yet basic tool like Excel, we needed tutoring systems. Many tutoring systems use Microsoft Excel as the platform for the tutoring. One such system is the XTutor [31]. This is a basic tutor where the students are given a problem statement and instructions, which tell them how and where to enter the data. This system is an example-tracing tutor where the database consists of one example per problem type and each example can be filled in with different data such that the final output would be different. The students are required to enter data into some cells, some cells have a pre-fixed set of data from which the student can choose one and finally some cells are auto-filled. When they need help with entering data into the cells, the system provides them with a series of hints. The last of hint prompts them the answer. The data entered into the cell is checked and immediate and appropriate feedbacks are also given. If any of the examples require the use of formulas, the system makes sure that each of the variables used in the formula is defined. One drawback of the system is that the problem solving is not automatic. Instead, the user has to decide which approach he is going to take before starting to solve it.

2.3 Test Generator

A critical area in Intelligent Tutoring Systems is to test what the students have learned with the help of the tutor. It determines the effectiveness of the tutor and the teaching techniques it used. According to [32], students who misuse tutoring systems learn only ⅔ of what they could learn without misuse. This misuse is generally in the form of prompting the system for hints until it gives out the right answer or it could be that the students have by-hearted some lessons after a certain number of tries. This leads us to the conclusion that sometimes it is necessary to create unique tests. This ensures that the test cannot be plagiarized and also makes sure that if a student wants to retake a test, he is provided with a different one each time.

Yoo et al. [33] talks about a test generating system that has two parts - a template editor and a question generator. Figure 3 shows us the architecture of the system. The template editor is provided with a well defined graphical user interface that allows the teacher to fill in details of the generalized question into the template. Each template is used by the question generator to generate a set of questions that are conceptually similar. The template also stores information like the difficulty level of the questions, right and wrong answers and other examples or explanations that will help the student to understand the concepts covered in the questions. Questions can be generated on-demand. The template editor is built on Linux platform and consists of a graphical user interface, which allows us to define various parameters for the question template. The concept base consists of all the concepts that the curriculum aims to cover. Rules are used to represent the relationships among the concepts and sub-concepts. The template editor is interfaced with the concept base so that each template that is generated is associated with a concept. We can also use the template editor to open, edit and save an existing

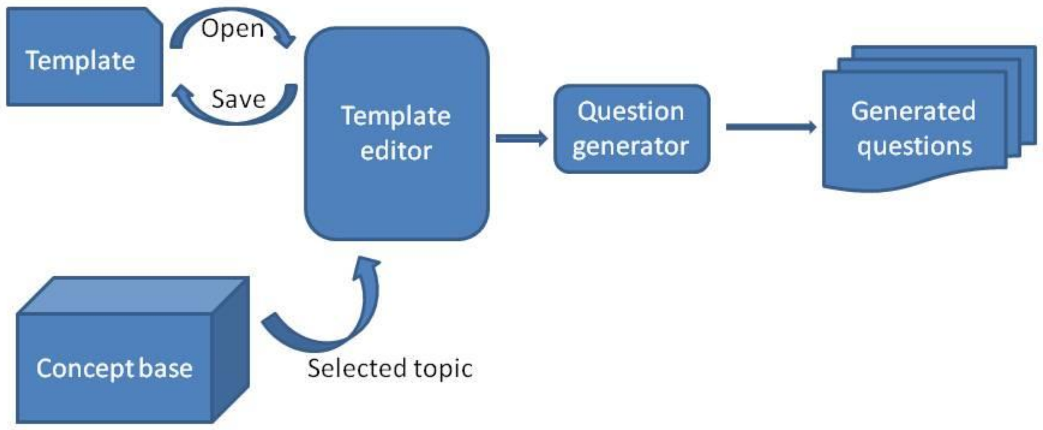


Figure 3 : Architecture of the question generator.

template. Generated templates are represented using XML. Each has a set of attributes like Question body, answer, wrong answers, explanations, variables, etc. that are represented as top-level XML elements. Scripts are also provided with some templates that help us define relationships between different variables of a problem. The template editor is designed to be user-friendly and displays the question templates in a form readable by the user. Template input given by the user is also translated into XML format. The system can also show us a preview of a question generated by a template.

Using the same idea of templates, Lentini et al. [34] describes a tutor generator system to generate self-instructive programmed spreadsheets. The main goal of the system is to make programmed spreadsheets readily available to the end users along with tools that support and help understand the usage of the spreadsheets. For example, the system takes in an input as shown in Table 1 and gives out the output as shown in Table 2. The student fills in the empty cells while the cells with formulae are evaluated by the system. The first part of the problem lies in reconstructing the spreadsheet by identifying the underlying relationships between different variables or columns of the spreadsheet. The second step is to generate explanations depending on the cell relationships after updating. In case of a large spreadsheet, it is subdivided and relationships between individual components are also identified.

Table 1 - Programmed spreadsheet input to the system.

	A	B
1	Variables	Values
2	X	10
3	Y	7
4	Z= (X-Y)	B2-B3

Table 2 : Evaluated spreadsheet output from the system.

	A	B
1	Variables	Values
2	X	10
3	Y	7
4	Z= (X-Y)	**3**

It consists of two modules - the knowledge acquisition module and the tutor generator. The architecture of the system is shown in Figure 4.

2.3.1 Knowledge Acquisition Module

As the name specifies, it extracts the knowledge coded into the spreadsheets and forms a knowledge base from it, which is then used to generate the tutor. The knowledge base constructed consists of knowledge about the teaching domain called expert knowledge and knowledge about teaching strategies called tutor knowledge. The expert knowledge is made up of three parts. Application knowledge (domain dependent knowledge) is the knowledge that is automatically acquired by the knowledge acquisition module from the programmed spreadsheet. Domain-independent knowledge is knowledge fixed for all domains and consists of mathematical rules, definitions of operations and functions etc. External knowledge is knowledge added by the tutor designer when he/she feels it necessary. It includes external domain knowledge that is not present in the programmed spreadsheets like some colloquial terms relating to the domain of interest.

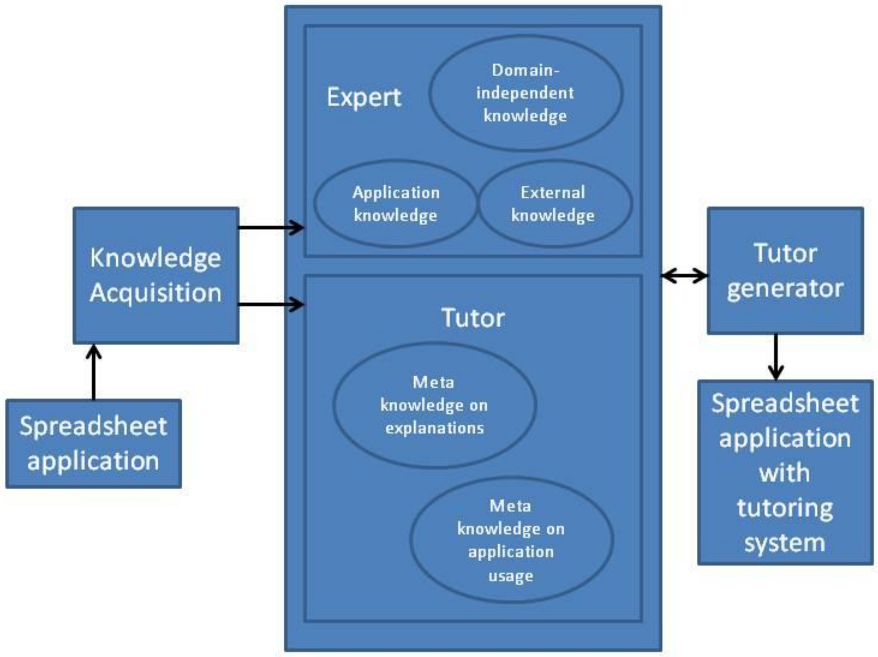

Expert
Domain-independent knowledge
Application knowledge
External knowledge
Knowledge Acquisition
Tutor
Meta knowledge on explanations
Meta knowledge on application usage
Spreadsheet application
Tutor generator
Spreadsheet application with tutoring system

Figure 4 : Architecture of the self-instructive spreadsheet generator system.

The tutor knowledge consists of meta-knowledge on explanations that helps the generated tutoring system to select the appropriate explanations to give at any particular point. The designer provides this information. The meta-knowledge on application usage consists of information regarding the separate components of the spreadsheet. It is formed as a result of an analysis of the object knowledge and the spreadsheet. It helps in sequencing the actions performed by the end user while filling in the data into the spreadsheet. This module acts as a parser and iterates through the spreadsheet. Each cell is then represented as a PROLOG instruction.

For example, a cell in row X and column Y with some content CONTENT will be represented as,

Cell (X, Y, CONTENT)

CONTENT can be a number, string or a formula. Once this model is built, the cells containing numeric data or formulae are assigned special names to identify them for furthering processing. The next step is to identify the attributes that have the same formula. These are called parametric quantities. Relationships among these quantities are established with the help of the formulas they contain. For example, consider the example in Table 2. X, Y and Z are parametric quantities here and the relationship among them can be represented shown in Figure 5,

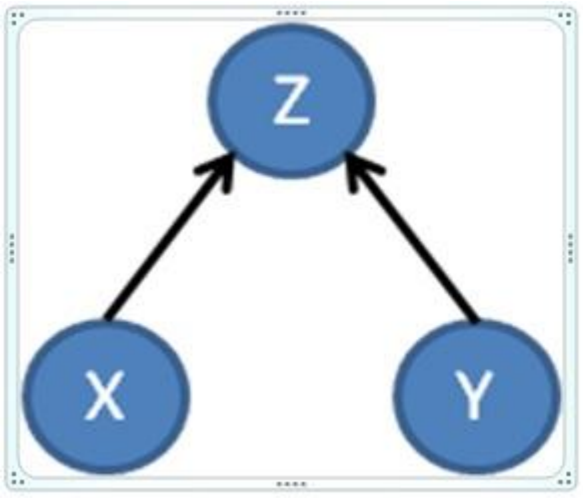


Figure 5 : Relationships among parametric quantities.

2.3.2 Tutor Generator

This module uses the knowledge base to generate tools to understand and use the application. It consists of a guide that helps understand the mathematical model of the programmed spreadsheet. This guide contains information about the relationships between various variables of the spreadsheet, the relationships between different components of the spreadsheet and information about the quantities of the spreadsheet that affect the calculations of other quantities. The tutor generator also consists of an interactive tutor that checks the end user's activity.

Excel is a tool that is widely used by a very huge section of the IT industry all over the world. There are wide ranges of applications for Excel. In the above examples, we have seen tutoring systems based on Microsoft Excel. All of them assumed that the students have prior knowledge of the Excel tool. As a result, teaching the working of the tool becomes as important as learning it. However there are no proper classes exclusively designed to teach and test students on Excel. In this thesis, we propose a web-based tutor that can teach Microsoft Excel to the students and also generate unique tests based on the lessons dynamically.

2.4 Apache POI

Apache POI project designs Java APIs that have the ability to read, edit and write various file formats such as Microsoft Word, Microsoft PowerPoint and Microsoft Excel. These APIs help us to perform many operations like create a new file, manipulate it with any changes that we want to make and also help to save back our changes to the file. Currently, it consists of API that can support most of the OOXML and OLE2 formats such as Microsoft Office 2007 and 2008 (.xlsx, .docx, .pptx). It is also working on developing interfaces that can handle both the OLE2 and the OOXML formats for each of the MS Office applications. Support for Microsoft Outlook has also been added recently. This is a powerful API that can be very useful for the development of software and tools that generate various reports for surveys, analysis or decision support.

Apache POI provides two different API to handle Microsoft Excel file formats – HSSF and XSSF. HSSF is used to work with Excel's '97(-2007) (.xls) file format while the XSSF is used to work with Excel's 2007 OOXML (.xlsx) file format. Both of the APIs have been implemented using Java and can help in creating new spreadsheets, modifying the spreadsheets, reading and writing from the spreadsheets.

3 Implementation

The following sections give a complete overview of the different modules implemented in developing the tutoring system. The first section outlines the coursework that we have designed for this tutoring system and the approach that we have taken to teach the course. The second section discusses the automatic question generating capability of the system. In the third section, we talk about how the system evaluates the student and gives feedback.

A walkthrough of the tutoring system in action:

- Register yourself.
- Login with your username and corresponding password.
    - o If the password is wrong, you are redirected to the Login page again.
    - o If the username is not in the system, you are redirected to the Register page.
- You are logged in, and at the Home page, which shows you a welcome message and the list of lessons that are planned to be covered in the course. Each lesson has a description of what the lesson consists of and a button ‘Watch the video’ which takes you to the next page.
- Clicking on ‘Watch the video’ button of any lesson, takes you to the page with a tutorial video for that particular lesson. After watching the video, the user can click on ‘Take the Test’ button to go to the next page.
- The ‘Take the Test’ buttons gets you to the next page, which consists of a video explaining how to complete the test. There is also a link to the PDF file consisting of

instruction on how to complete the test. The user can download the test by clicking on the 'Download the test' button. Once the user finishes all the steps required to complete the test, he/she can submit the test to the system by clicking on 'Browse' and selecting the appropriate file. Once the user clicks on 'Submit', the system then evaluates the file submitted by the user and gives feedback in the next page.

- This feedback consists of the score of the user along with a choice if he/she wants to move on with the process or if he/she wants to retake the same lesson again in order to improve their score.

When the teacher needs to introduce a new chapter, the following need to be created:

- A video explaining the necessary concepts in .mp4 format.
- Templates file to generate equivalent tests dynamically.
- A video and/or PDF file instructing the student on how to finish the test.

3.1 Coursework

The coursework for this system has been designed assuming that the student has never used Microsoft Excel before and has no knowledge of how to use it. This coursework goes through the very basics of Microsoft Excel that can get the student familiar with its usage. The following is the outline of the lessons taught in this course.

· Lesson 1: Introduction

This introduces the user to the layout of an Excel sheet and the various terminologies used while navigating in it. It explains about cells, cell references and how to find them.

· Lesson 2: Adding data

The lesson explains the different ways to add and remove data from the cells in Excel sheets. It also explains how to add new rows and columns into the sheet and to save their progress.

· Lesson 3: Formatting

Here the users learn about how they can format the data to look more presentable and also how to transform their data into tables.

· Lesson 4: Formulas

This lesson explains how to generate data automatically. It also explains the concept of formulas and functions in Excel sheet and how they can be used.

Since we have a lot of menus and different ways to reach a particular place in Microsoft Excel, the best way to teach it would be through videos. This allows the users to watch the video at their own pace and also allows them to watch it any number of times. So, we have recorded a few videos, which show the lessons in action and also have a narrative in the background. These videos are of .mp4 format and have been recording using the Camtasia screen recording tool and Microsoft Excel on a Windows operating system.

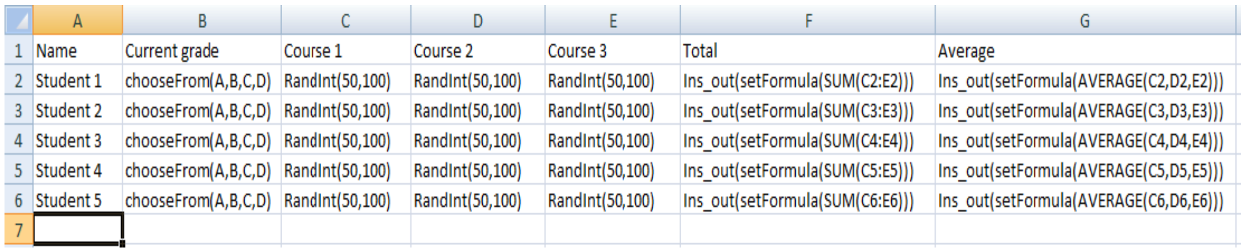
3.2 Automatic Test Generation

Being an online tutoring system, the user is allowed to take the tests designed for each lesson, any number of times. As Yoo et al. [26] explain, this puts a lot of pressure on the teacher to design a new test with equivalent questions as the previous one, every time a student decides to retake a test. To reduce the amount of time and effort put on this task, we use the idea of templates.

Templates are a guide to generate or create more than one thing with a same format. In this thesis, these templates are Excel files that need to be created only once for each test and can be used any number of times to generate tests for the students dynamically. These help to generate more than one test with a similar format and preserve the difficulty and nature of each test. The process of generating and using these templates is described below.

3.2.1 Templates

In this thesis, the templates are Excel files in .xls format. Each test has its own template file. Every time the student clicks on 'Download the test' button, the tutoring system processes the template file. It first reads the template file and creates two new files – first one is the output file. This is the final output that the tutoring system is expecting from the student. Figure 6 shows you an example of the template used in this project.



	A	B	C	D	E	F	G
1	Name	Current grade	Course 1	Course 2	Course 3	Total	Average
2	Student 1	chooseFrom(A,B,C,D)	RandInt(50,100)	RandInt(50,100)	RandInt(50,100)	Ins_out(setFormula(SUM(C2:E2)))	Ins_out(setFormula(AVERAGE(C2,D2,E2)))
3	Student 2	chooseFrom(A,B,C,D)	RandInt(50,100)	RandInt(50,100)	RandInt(50,100)	Ins_out(setFormula(SUM(C3:E3)))	Ins_out(setFormula(AVERAGE(C3,D3,E3)))
4	Student 3	chooseFrom(A,B,C,D)	RandInt(50,100)	RandInt(50,100)	RandInt(50,100)	Ins_out(setFormula(SUM(C4:E4)))	Ins_out(setFormula(AVERAGE(C4,D4,E4)))
5	Student 4	chooseFrom(A,B,C,D)	RandInt(50,100)	RandInt(50,100)	RandInt(50,100)	Ins_out(setFormula(SUM(C5:E5)))	Ins_out(setFormula(AVERAGE(C5,D5,E5)))
6	Student 5	chooseFrom(A,B,C,D)	RandInt(50,100)	RandInt(50,100)	RandInt(50,100)	Ins_out(setFormula(SUM(C6:E6)))	Ins_out(setFormula(AVERAGE(C6,D6,E6)))
7							

Figure 6 : An example of the template format used.

Each cell contains a command in the form of text that is to be processed by the tutor system in order to dynamically generate the new test. The format of the text represents a standard function call, which is easy to understand by the teacher designing the template. Each text contains the name of the operation that we need to be performed on the cell along with a set of parameters/information that we pass on to the tutor, which contain details about the cell formatting, content and cell formulas.

command (param1, param2, param3, …)

The operations currently supported by this tutor system are the described below along with the corresponding command and parameters that need to be used to perform the operation. The two new files on processing the template are referred to as the 'Gen_Test' file, which contains the test questions and is presented to the user, and the 'Gen_Output' file which is the output file generated. Each command is written in a separate cell to which the command needs to be applied.

- *Set(String param1)*

The parameter passed into this command ‘param1’ is a string and is set as the value of the cell in both the Gen_Test file and the Gen_Output file.

- *RandInt(int param1, int param2)*

  This command takes in two integer values ‘param1’ and ‘param2’. It considers param1 as the lower limit and param2 as the upper limit and generates a random integer value existing between these two values. This generated integer is then set as the cell’s value in both the Gen_Test file and the Gen_Output file.

- *RandFloat(float param1, float param2)*

  This command has the same functionality as *RandInt* except that it takes in two float parameters and generates a random float value.

- *chooseFrom(String param1, String param2, … , String paramN)*

  This command takes in a list of strings as parameters. It randomly chooses one of the values and sets it as the cell’s value in both the Gen_Test file and the Gen_Output file.

- *Percentage(double param1, double param2)*

  This command takes in a start value and an end value both are of double type. Using the given limits, it generates a random double value between them and outputs it.

- *On_Off (String param1, String param2)*

  When we need to set a different value to a cell at the same position in Gen_Test file and the Gen_Output file, we use this command. This command takes in two string parameters, param1 and param2. The cell in Gen_Test is set with 'param1' while the corresponding cell with the same reference number in Gen_Output file is set with 'param2'.

- *Ins_gen (String param1)*

  This command can be used to insert a cell only in the Gen_Test file. When the tutor system processes a cell with the *Ins_gen* command in it, it sets the cell with the parameter passed into it (i.e. param1) and inserts it only into the Gen_Test file. The cell is deleted from the Gen_Output file.

- *Ins_out (String param1)*

  This command performs the same function as *Ins_gen* except that it is used to inserts the new cell into the Gen_Output file instead of the Gen_Test file.

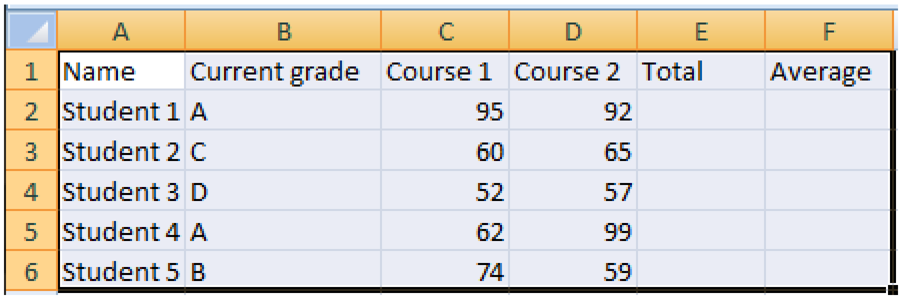
- *SetFormula(String param1)*

  To set a formula to a particular cell in Gen_Output, we have the setFormula command which takes in a string parameter that is in the format of the formula we want. For example, setFormula(SUM(A1,A2))

- *formatCell (String bgColor, String fontSize, b, i, String fontColor)*

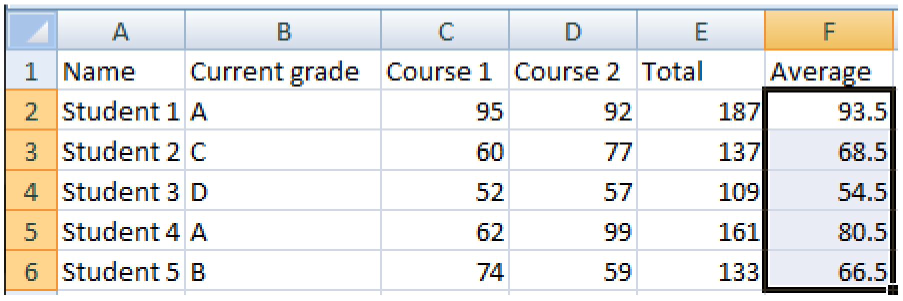
  This command takes in five parameters, each of which contains formatting information. The parameter '*bgColor'* allows us to specify the background color for the cell. '*fontSize'* allows us to specify the font size we want the cell's text to have. The third and fourth parameters are Boolean values that specify whether the text should be bold or italic respectively. If the Boolean value is 1, it turns on the bold or italic format and a value 0, turns them off. The final parameter is the *fontColor*. This is any static color value belonging to the java.awt.Color class (Example: white, lightGray, etc.)

Once the tutor system processes the template file, it produces the Gen_Test, which is given to the student as his test, and the Gen_Output file which is saved on the server for evaluating the student's response at a later stage. Furthermore, instructions specifying what the students are expected to do in the test are also provided to the user in the form a PDF file and a video. For the example in Figure 6, the Gen_Test and Gen_Output files look like below,



	A	B	C	D	E	F
1	Name	Current grade	Course 1	Course 2	Total	Average
2	Student 1	A	95	92		
3	Student 2	C	60	65		
4	Student 3	D	52	57		
5	Student 4	A	62	99		
6	Student 5	B	74	59		

Figure 7 : Generated test file - Gen_test



	A	B	C	D	E	F
1	Name	Current grade	Course 1	Course 2	Total	Average
2	Student 1	A	95	92	187	93.5
3	Student 2	C	60	77	137	68.5
4	Student 3	D	52	57	109	54.5
5	Student 4	A	62	99	161	80.5
6	Student 5	B	74	59	133	66.5

Figure 8 : Generated output file - Gen_Output

3.3 Evaluation

Once the user finishes taking the test, he/she submits the file back to the system. This file is then stored on the server along with a unique ID by which the file can be identified. In this evaluation phase, the user-submitted-file is compared with Gen_Output file generated during the Automatic Test Generation phase.

The following are the steps by which the system determines if it should proceed further with the evaluation or not.

Step 1: It compares the number rows and the number of columns in both the files. If they are same, it goes to step 2. Otherwise, it stops further processing and asks the user to check their solution or gives them the option to take the lesson again.

Step 2: There are four main features of a cell that need to be compared – cell value, formatting, cell formula and the data type of the cell. We recursively traverse through every cell of both the files comparing all the four features at every step. The system maintains error counts in all the four categories and the final score is calculated by giving a certain weight to the features. The system uses these five scores to give feedback to the student.

Step 3: A certain threshold level is maintained by the system by which it determines whether to give an option of proceeding to the next lesson or not.

4 Conclusion

In this thesis, we have implemented an automatic tutor system for teaching Microsoft Excel with the help of videos. The main contribution of this work is the automatic test generation system. Using the constructs of Java, this web-based tutor allows a user to login and provides him/her with a list of lessons to select from. Once the user decides upon the lesson, a video related to the user's choice is presented. The user is then allowed to take a test which contains questions related to the video that the user has just viewed. The user can take the test any number of times. The dynamic test generation system allows us to create tests with equivalent questions but unique data hence preventing any form of cheating. The system also provides feedback to the user based on the number of mistakes he/she has made in the test. We have tested the system by simulating answers from a student with a good score and a student with a poor score. The good student receives his/her score and is allowed to proceed to the next lesson while the poor student was almost always prompted to watch the video again.

4.1 Future Work

4.1.1 Automatic Template Creation

In the current work, creating the templates can be difficult as the preparer needs to know the formats and the supported commands before creating it. It would be nice to automate this process with a probable GUI where the preparer is presented with the supported commands along with their formats and he/she has the ability to select the cells in which the selected command needs to be applied.

4.1.2 Evaluation process

This tutor system blindly evaluates a student by comparing the cell values of the student's response with the expected result and gives feedback based on the number of mistakes counted. This process can be refined so that the system allows for human errors like spelling mistakes and punctuation mistakes. In the current system, if a student fills in "New Yok" instead of "New York", the system evaluates it as completely wrong. It can be argued that such trivial mistakes should not play a role in evaluating Microsoft Excel skills of the student.

4.1.3 Limitations of Apache POI

Graphs and charts are an integral part of Microsoft Excel and they play a major role in presenting data. They make the data easier to read and to compare. Large amounts of data can easily be visualized with the help of graphs making it easy to analyze. Although Apache POI has many latest features that support reading and manipulating Excel files, it has its own limitations. There is very limited support for graphs and charts in POI. It only allows creating a few simple charts but there is no way to modify these charts.

Macros and Pivot tables are also not completely supported by Apache POI. Files containing these features can be processed by the library but it has no capability to create a new Macro or a Pivot table. It would be interesting to see if we can overcome these limitations of the library with our own custom code.

References

[1] Nwana, H. S. (1990), “Intelligent Tutoring Systems: An Overview”, Artificial Intelligence Review 4 (4).

[2] Barrón-Estrada, M.L., Zatarain-Cabada, R., Beltrán, J.A., Cibrian, F.L., Hernández Pérez, Y.: An Intelligent and Affective Tutoring System within a Social Network for Learning Mathematics. In: Pavón, J., Duque-Méndez, N.D., Fuentes-Fernández, R. (eds.) IBERAMIA 2012. LNCS, vol. 7637, pp. 651–661. Springer, Heidelberg (2012)

[3] Yue Gong (November 2014). Student Modeling in Intelligent Tutoring Systems. Worcester Polytechnic Institute.

[4] Pressey S.L. 1927. A machine for automatic teaching of drill material. *School and Society*, **25** (645), 549–552.

[5] VanLehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., et al. (2005). The Andes physics tutoring system: Lessons learned, International Journal of Artificial Intelligence in Education, 15(3).

[6] VanLehn, K., Jordan, P., Rosé, C. P., Bhembe, D., Bottner, M., Gaydos, A., Makatchev, M., Pappuswamy, U., Ringenberg, M., Roque, A., Siler, S., & Srivastava, R. (2002). The architecture of Why2-Atlas: A coach for qualitative physics essay writing. In S. A. Cerri, G. Gouarderes, & F. Paraguacu (Eds.), Proceedings of the Sixth International Conference on Intelligent Tutoring Systems (pp. 158- 167). Berlin: Springer-Verlag.

[7] Corbett, A. T., Koedinger, K. R., & Anderson, J. R. (1997). Intelligent tutoring systems. In M. Helander, T. K. Landauer, & P. Prabhu (Eds.), Handbook of human-computer interaction (2nd ed., pp. 849-874). New York: Elsevier

[8] Graesser, Arthur C.; Conley, Mark W.; Olney, Andrew. Intelligent tutoring systems. Harris, Karen R. (Ed); Graham, Steve (Ed); Urdan, Tim (Ed); Bus, Adriana G. (Ed); Major, Sonya (Ed); Swanson, H. Lee (Ed), (2012). APA educational psychology handbook, Vol 3: Application to learning and teaching. , (pp. 451-473). Washington, DC, US: American Psychological Association, viii, 668 pp.

[9] . D. Bitzer, P. Braunfield, W. Lichtenberger, PLATO: An Automatic Teaching Device IEEE Trans. Educ. 4, 157 (1961).

[10] Smith, S., and Sherwood, B.A. Educational uses of the PLATO computer system. Science 192 (1976).

[11] The 20-year History of ITS. http://www.aect.org/edtech/ed1/19/19-04.html.

[12] Page, E. B. (1966). The imminence of grading essays by computer. *Phi Delta Kappan*, January, 238-243.

[13] Burstein, J., Kukich, K., Wolff, S., Lu, C., and Chodorow, M. (1998). Enriching automated essay scoring using discourse marking. *Proceedings of the Workshop on Discourse Relations and Discourse Markers*, Annual Meeting of the Association of Computational Linguistics, August, Montreal, Canada.

[14] Dikli, S. (2006). An overview of automated scoring of essays. The Journal of Technology, Learning, and Assessment 5/1. Retrieved from http://escholarship.bc.edu/jtla/

[15] Murray, T. (1999). Authoring Intelligent Computer Systems: An analysis of the state of the art. International Journal of Artificial Intelligence in Education, 10, 98-129.

[16] Marcke, K. van. (1998). GTE: An epistemological approach to instructional modeling. Instructional Science, 26, 147-191.

[17] Ainsworth, S., Major, N., Grimshaw, S., Hayes, M., Underwood, J., Williams, B., & Wood, D. (2003). REDEEM: Simple intelligent tutoring systems from usable tools. In T. Murray, S. Blessing, & S. Ainsworth (Eds.) Authoring Tools for Advanced Learning Environments (pp. 205-232). Dordrecht, the Netherlands: Kluwer Academic Publishers.

[18] Shimic, G., & Devedzic, V. (2003). Building an intelligent system using modern Internet technologies. Expert Systems With Applications, 25, 231-246.

[19] Graesser, A. C., Lu, S., Jackson, G. T., Mitchell, H., Ventura, M., Olney, A., & Louwerse, M. M. (2004). AutoTutor: A tutor with dialogue in natural language. *Behavioral Research Methods, Instruments, and Computers*, 36, 180-193.

[20] Kurt VanLehn. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. Educational Psychologist, 46(4):197–221, 2011.

[21] Chi, M. T. H., Siler, S., & Jeong, H. (2004). Can tutors monitor students' understanding accurately? Cognition and Instruction, 22, 363–387

[22] Jeong, H., Siler, S., & Chi, M. T. H. (1997). Can tutors diagnose students' understanding? In M. G. Shafto & P. Langley (Eds.), Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society (p. 959). Mahwah, NJ: Erlbaum.

[23] Putnam, R. T. (1987). Structuring and adjusting content for students: A study of live and simulated tutoring of addition. American Educational Research Journal, 24(1), 13–48.

[24] Chi, M. T. H., Roy, M., & Hausmann, R. G. M. (2008). Observing tutorial dialogues collaboratively: Insights about human tutoring effectiveness from vicarious learning. Cognitive Science, 32, 301–342.

[25] Graesser, A. C., Person, N., & Magliano, J. (1995). Collaborative dialog patterns in naturalistic one-on-one tutoring. Applied Cognitive Psychology, 9, 359–387.

[26] Chi, M. T. H., Siler, S., Jeong, H., Yamauchi, T., & Hausmann, R.G. (2001). Learning from human tutoring. Cognitive Science, 25, 471–533.

[27] Core,M. G., Moore, J. D., &Zinn, C. (2003). The role of initiative in tutorial dialogue. Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL) (pp. 67–74). Morristown, NJ: Association of Computational Linguistics.

[28] Evens, M., & Michael, J. (2006). One-on-one tutoring by humans and machines. Mahwah, NJ: Erlbaum.

[29] Samuel Alexander, Abdolhossein Sarrafzadeh and Stephen Hill, Easy with Eve: A Functional Affective Tutoring System, Workshop on Motivational and Affective Issues in ITS. 8th International Conference on ITS 2006, pp 5-12, 2006

[30] T. Reinhardt, N. Pillay, Analysis of spreadsheet errors made by computer literacy students, Proceedings of the IEEE International Conference on Advanced Learning Technologies, 2004, pp. 852–853.

[31] Gheorghiu R, Vanlehn K (2008) XTutor: an intelligent tutor system for science and math based on excel. In: Woolf BP et al. (eds) Intelligent tutoring systems: 9th international conference, ITS 2008. Springer, Germany.

[32] Baker, R., Corbett, A., & Koedinger, K. (2004). Detecting student misuse of intelligent tutoring systems. In Intelligent tutoring systems (pp. 531–540).

[33] Yoo, J., Yoo, S., & Rusek, S. A Question Generator for an Online Tutoring System, Proceedings of the ED-MEDIA conference, (Montreal, Canada, 2005), 1820-1825.

[34] Lentini, M. Nardi, D., & Simonetta, A. (2000). Self-instructive spreadsheets: an environment for automatic knowledge acquisition and tutor generation. International Journal of Human-Computer Studies, 52(5), 775-803. doi:10.1006/ijhc.1999.0363.