A Tutoring System Discovering Gaps in the Current Body of Students' Knowledge

Sylvia Encheva[1] and Sharil Tumin[2]

[1] Stord/Haugesund University College, Bjørnsonsg. 45, 5528 Haugesund, Norway
sbe@hsh.no
[2] University of Bergen, IT-Dept., P. O. Box 7800, 5020 Bergen, Norway
edpst@it.uib.no

1 Introduction

Most learning management systems do not support a flexible navigational structure and an open assessment mechanism allowing incorporation of new pedagogical ideas. These learning management systems do not allow server side scripting for dynamic presentation of content either. Our idea is to develop a tool follows and supports each student's learning process by modelling a tutoring system based on classroom teaching.

This paper focuses on a tutoring system discovering gaps in the current body of students' knowledge. A framework for building new courses or updating existing ones by choosing learning objects developed at universities that are members of a federated learning system is also provided. The aim of this framework is twofold. First assisting a lecturer in collecting learning objects closest to the lecturer's vision on what a subject should contain and how the content should be presented. Secondly, present a student with content, tailored according to student's individual learning preferences.

Let us consider a lecturer affiliated with an educational institution that is member of a federated learning system. Suppose the system is able to provide a large number of learning objects (LOs) upon the lecturer request. A lot of time and efforts can be spared if the system can first filter and rank those LOs according to the lecturer's preferences. Another important issue is how to build a course supporting student's individual learning preferences. The system can help both the lecturer and the students by presenting each student with a LO chosen from the related set of selected LOs but tailored according to the student's individual learning styles and preferences. Identifying a student's style and then providing instruction consistent with that style contributes to more effective learning [3].

2 Related Work

Expert and theoretical knowledge about the use of technology for assessment is offered in [2].

Please use the following format when citing this chapter:
Encheva, Sylvia, Tumin, Sharil, 2006, in IFIP International Federation forInformation Processing, Volume 204, Artificial Intelligence Applications and Innovations, eds. Maglogiannis, I., Karpouzis, K., Bramer, M., (Boston: Springer), pp. 442–449

An intelligent system for assisting a user in solving a problem was developed in [7]. A personalized intelligent computer assisted training system is presented in [8]. A model for detecting student misuse of help in intelligent tutoring systems is presented in [1]. An investigation of whether a cognitive tutor can be made more effective by extending it to help students acquire help-seeking skills can be found in [5]. Evidence that when used appropriately, on-demand help can have a positive impact on learning was found in [9].

We focus on a different aspect of use of intelligent tutoring systems which is an attempt to first detect lack of prior knowledge of each student and then fill in the gaps.

3 Tutoring System

A course work is made of a set of subjects. Each subject contains an ordered set of related topics. Each topic contains a set of definitions, statements, examples, hints and tests. The tests within a topic are grouped into levels of difficulties and are used to asses students' level of knowledge about a particular topic.

For each topic, a student is initially presented with a Web page containing theory (definitions and statements) and supporting examples, where it is explicitly stated which part is required and which part is recommended for additional reading, to the current curriculum. The student is expected to study the presented material.

The student is then asked to take a test in the form of multiple choice question tests and provide a level of confidence in each answer. This test assumed that the student has learned and understood the presented materials in that particular topic.

From the result of the test, the system will guide the student through an automatic tutoring session if the student fails the test or presents the student with the next topic. The automatic tutoring session guides the student downwards (lower level of difficulties) according to the student mistakes and upwards again using the same path without a human tutor within the current topic.

Suppose the current topic was designed with three levels (A,B,C) in descending order of difficulties 1. After failing the test on level A, the automatic tutoring session brings the student to level B for reading suitable theory, hints and solving new examples. A new multiple choice question test for level B is then presented to the student. The automatic tutoring session checks the result of the level B test. If the student fails the test B then the automatic tutoring session brings the student down to the next (easier) level (level C). If the student passed the test B then the automatic tutoring session brings the student up again to next (more difficult) level (level A).

These iterations of tests between levels terminate when the student passes the multiple choice question test at the top most level (level A) or the student chooses to break out of the current automatic tutoring session. The materials and the multiple choice question tests presented at each level are dynamically

produced depending on student's responses to the given tests. At any level the automatic tutoring session distinguishes wrong answers caused by miscalculation, lack of knowledge, and misconception. By automatically tracking the students' paths in response to theirs tests results, the automatic tutoring session helps the students to learn from their mistakes.

The automatic tutoring session keeps a record of all students' interactions with the system, providing data about level of knowledge of each student. This information can be used to compare the performance of each student against the whole class, compare the overall performance of different classes against each other, and to keep statistics of usage of particular learning units (statement, problems, hints, and examples). This helps content developers (teachers, lecturers, tutors) to improve on certain aspect of learning units by including new materials.

The automatic tutoring session is the heart of our tutoring system employing several intelligent agents working in concert which respond to students real time interactions to the system Web based interface. The agents interact with each other, record students' knowledge states and status and based on rules stored in the database provide students with personalized adaptive learning experience.

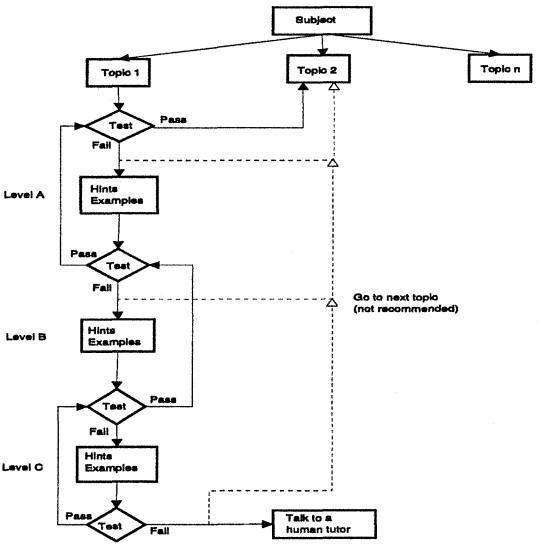

Fig. 1. Automated tutoring session

4 System Architecture

The system framework is composed of three main components: Web server, database and agents implemented on a Linux server.
Web server: Apache Web server with mod_python (Python interpreter). Dynamic HTML pages are created by server-side scripts (SSS) written in Python in response to students' interactions with the system using a Web browser. Having Python interpreter embedded in the Web server improves performance and reduces response time thus increases user satisfaction. Python is used to program the whole system including database connectivity and the agents.
Database : SQLite is used as a relational database. SQLite is a small *C* library that implements a self-contained, embeddable, zero-configuration Structured Query Language (SQL) database engine. SQLite is small and fast compare to other open source database engines for example PostgreSQL and MySQL. A complete database is stored in a single disk file that supports databases up to 2 terabytes (241 bytes) in size. Database files can be freely shared between machines with different byte orders. It is easy to set up a distributive multi-databases system using SQLite. A Python module pysqlite is used as a DB-API 2.0-compliant database interface for SQLite databases from within Python program.

Each subject meta-data is stored in its own database. Each student's automatic tutoring session data has its own database. We used native file system file structure to structure subjects and students' classes. One database is used for user administration and global references to others operational databases. This provides us with a flexible and expendable multi-databases system.
Agents: Independent agents written in Python provide the system support for students' automatic tutoring session. The Web server scripts and agents communicate with each other using XML-RPC (remote procedure call) over HTTP. The communication sub-system is also implemented in Python using xmlrpclib module. The system is supported by user, test and diagnostic agents that will be discussed later on in this paper.

By using three open source software - Apache, SQLite and Python we are able to implement a machine based tutoring system for teaching mathematical courses for higher education.

4.1 Agents

Agents are free running programs, capable of mutual interaction. The interaction can be in the form of message passing (XML-RPC) or producing changes in their common environment (data saved into or fetched from databases).
User agent: In order to use service provided by the system a person must register as a user to the system. A registered user will be given identity (user identification and password) and authority (student or teacher) within the system. The user agent is responsible in providing users' authentication and authorization data to other agents and keeping track of user current session, knowledge state

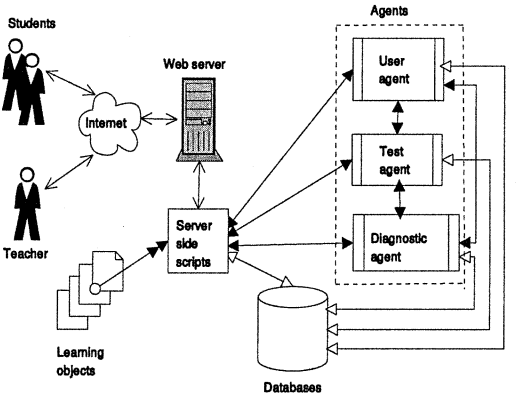

Fig. 2. System architecture

and working profiles. The user agent records students learning progress through the system. All learning material subscriptions and test results for a particular student are saved into the student personal database. Data saved in this database is used for the student's automatic tutoring session. These data also provides the student's audit-trail. Such audit-trail data can be used for billing purposes and course planning.

Test agent: This agent is equipped with a pedagogically crafted scheme with a set of questions and answers for each topics and levels. The agent responds to a particular student test results and his/her audit-trail to provide the student with personalized automatic tutoring session. The student can then subscribe to those learning materials suggested by his/her automatic tutoring session. The agent also calculates scores, show result status and keeps track of assessments taken by each student. After each assessment the test agent sends summarized information to diagnostic agent.

Diagnostic agent: Each automatic tutoring session is determined by pedagogical requirements. Each automatic tutoring session is structured using dependencies among learning materials, level and relationships between tests options, and inference rules triggered by a particular student interaction with the system. These requirements, relations and rules are crucial to the effectiveness of the system as a learning aid. Expert tutors experienced in the subject of learning are employed to define the pedagogical requirements of the diagnostic agent. This agent is the most difficult to implement and the success of the system depends on its implementation.

In order to support an adaptive learning environment these agents need a large collection of LOs, in particular - hints, examples and tests. A federation of organizations sharing LOs can in principal provide such collection.

5 Framework for Building New Courses

Suppose universities $U_1, U_2, ..., U_n$ are members of a federated learning system. A lecturer, affiliated with university U_j is developing a new course.

What is known for this course is the following - students will obtain degree (D) in for example engineering, mathematics, statistics or physics; it is at bachelor, master or Ph.D. level (L); it is an introductory, intermediate or advanced course level (C); table of contents of the subject (S).

Let us assume that it is an advanced course in differential equations for engineering students on a master program in fire safety.

Figure3 illustrates how an automated system for building courses using LOs in a federated learning system can assist a lecturer. A lecturer sends a quarry to the system about existing LOs in an advanced (C) course in differential equations (S) for engineering students (D) on a master program (L).

(1) Select universities (U_D) offering degree in engineering.
(2) Select universities (U_L) offering master degree in engineering, (U_L $\subseteq$ U_D).
(3) Select universities (U_C) offering advanced courses, (U_C $\subseteq$ U_L).
(4) Select universities (U_S) offering courses in the desired subject, (U_S $\subseteq$ U_C).
(5) Select universities (U_LOs) offering courses that contain LOs, wanted by the lecturer, (U_LOs $\subseteq$ U_S).
(6) Collect the wanted LOs that belong to the universities in the set U_LOs.
(7) Choose the most suitable LOs.
(8) Consider whether the course is complete.

If one of the sets U_D, U_L, U_C, U_S, U_LOs is empty or the list of obtained LOs is incomplete, the lecturer will be asked to send a new quarry. In our example it could be a degree in physics, mathematics or statistics. If the system search exhausts all possibilities in this federated system the lecturer is advised to consider other options like search among other systems or develop the missing LOs.

Intelligent Diagnostics- It is based on each individual's learning styles and preferences. An intelligent agent will choose the most appropriate LOs for the course. An agent is first checking whether all definitions and statements required are included in the suggested LOs. Another agent determines whether the level of difficulties assumed for the new LOs corresponds to the level of difficulties of the suggested LOs. The unfolding model (shortest distance) is used while comparing different LOs. The level of difficulties is judged based on the included theory, examples and assessment tests. A questionnaire is put to the students for determining their individual learning preferences.

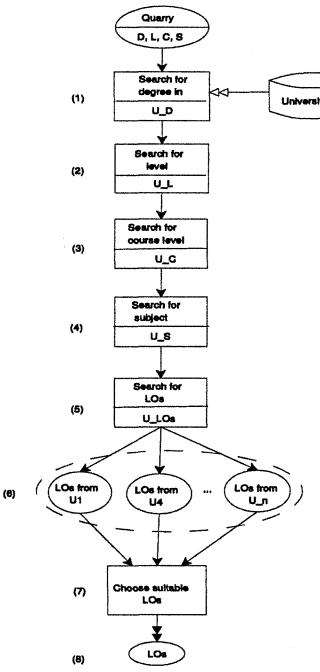

Fig. 3. Search process for LOs among federated universities

Stack Profiler - In the recommendation on how to proceed, a student can choose to subscribe to one or more suggested LOs. The student's LO subscriptions are placed in a stack-like structure in the student profile data. Initially, the profile stack contains a sequential ordering of LOs in a given subject. A student can choose to skip any presented LOs and go to the next one at any time.

Policy - The curriculum of each subject at every university should be described using a set of agreed upon metadata presented in a standard structure in a database.

LO Caching - If LOs in a course are connected with hyperlinks, the course builder risks to end up with some dead links during the semester. If all LOs in a course are cached on a local server, the course builder is sure that all LOs are going to be available to the students through the entire semester. The owners of the LOs have no control over the amount of students and number of times those LOs are used. However, the owners of the LOs can include f. ex. 1×1 pixel gif picture in every LO. Thus the owners will get information from log files for the number of times a LO has been used and by how many different users.

6 Conclusion

The system is designed to enhance the specific features of each user, without increasing the differences between users in what concerns the level of understanding or the ability to creatively use the acquired knowledge.

The paper describes also a framework for building new courses or updating existing ones by choosing learning objects developed at universities that are members a federated learning system. The aim of is to assisting a lecturer in collecting learning objects closest to the lecturer's vision on what a subject should contain and how the content should be presented, and to present a student with contents, tailored according to student's individual learning preferences.

References

1. Baker R S, Corbett A T, Koedinger K R (2004) Detecting student misuse of intelligent tutoring systems. Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg New Jork 3220: 531–540
2. Conole G, Crewe E, Oliver M, Harvey J (2001) A toolkit for supporting evaluation. The Association for Learning Technology Journal 9: 38–49
3. Jonassen D H , Grabowski B L (1993) Handbook of Individual Differences, Learning, and Instruction. Mahwah, N.J. Erlbaum.
4. Hron A, Friedrich H F (2003) A review of web-based collaborative learning: factors beyond technology. Journal of Computer assisted Learning 19: 70–79
5. Koedinger K R , McLaren B M, Roll I (2004) A help-seeking tutor agent. Proceedings of the Seventh International Conference on Intelligent Tutoring Systems, ITS 2004, Springer-Verlag, Berlin 227–239
6. Lepper M R, Woolverton M, Mumme D, Gurther G (1993) Motivational techniques of expert human tutors: Lessons for the design of computer-based tutors. In S.P. Lajoie, S.J. Derry (Eds.): Computers as cognitive tools. LEA, Hillsdale, NJ 75–105
7. Liu C, Zheng L, Ji J, Yang C , Li J, Yang W (2001) Electronic homework on the WWW. First Asia- Pacific Conference on Web Intelligence 540–547
8. Pecheanu E, Segal C, Stefanescu D (2003) Content modeling in Intelligent Instructional Environment. Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin Heidelberg New Jork 3190: 1229–1234
9. Renkl A (2002) Learning from worked-out examples: Instructional explanations supplement self- explanations. Learning and Instruction 12: 529–556