Playing to Program: An Intelligent Programming Tutor for RUR-PLE

Marie desJardins and an Ensemble of Students
University of Maryland, Baltimore County
1000 Hilltop Circle
Baltimore MD 21250
mariedj@cs.umbc.edu

Abstract

Intelligent tutoring systems (ITSs) have proven their effectiveness in contributing to student learning. ITSs are automated programs that provide students with a one-on-one tutor, allowing them to work at their own pace, so they can spend more time on their weaker areas of the subject matter. RUR–Python Learning Environment (RUR-PLE), a virtual environment to help students learn to program in Python, provides an interface for students to write their own Python code and then be presented with a visualization of that same code [CITE]. The RUR-PLE system provides a sequence of learning lessons for students to explore. We have extended RUR-PLE to provide an intelligent tutoring system interface that consists of three components: (1) a student model that tracks student understanding, (2) a diagnosis module that provides tailored feedback to students, and (3) a problem selection module that guides the student's learning process. In this paper, we describe the basis RUR-PLE system and our extensions, and present the results of a user study in which we evaluated the effectiveness of our three ITS modules.

1. Introduction and Related Work

ADD introduction and motivation.

2. Infrastructure

ADD description of concept map, how we built/tested it (i.e., justification for these concepts and why they're connected as they are, and how we instantiate and track it for a specific user (presumably using some kind of Bayesian updating). Idea: validate/finalize it by some testing process on a group of students (i.e., is it in fact the case that students in general need to understand concept X before they can apply concept Y) – use a problem suite (where each problem includes known concepts) to measure these dependencies.

3. Pre-test and Problem Selection

We designed a pretest to administer to students that will evaluate their understanding of various concepts. This pretest consists of twenty multiple-choice questions each of which assesses student understanding across a variety of concepts. The concepts we include in our evaluation are loops, function calls, conditionals and variables. As the students answer various questions correctly or incorrectly, we are able to predict their level of understanding of the various concepts from the accuracy of their responses. We designed pretest problems at a variety of difficulty levels in order to detect fine grained levels of comprehension.

The problems we presented to the students were more complex, as they involved writing programs to solve problems. These tasks also incorporated different concepts at different levels of difficulty.

4. Experimental Design

Twenty-four participants were recruited for the study. Participants were UMBC students who had completed at least one computer science major class but have not taken any 300 level computer science classes. The participants were asked to take a pretest (described above). The participant were presented with 10 programming problems to solve using a PUFFL. The participant were able to skip a problem if they had trouble solving it.

5. Conclusions and Future Work

ADD: What have we contributed? What are the takeaway lessons? What would we work on next?

References