Technical Report

Course Number: CS 4982

Summary of Dept of Computer Science Seminar Series: Supporting Learning of Programming with Tutoring Systems & other Instructional Environments

Name: Eric Foote

ID Number: 3502094

Tuesday November 30 2021

Supervisor: Dr. P.Mahanti

Department of Computer Science

University of New Brunswick Saint John

Summary of Supporting Learning of Programming with Tutoring Systems & other Instructional Environments

The talk began with an with an introduction describing what are tutoring systems. They are defined as systems that detect what students know, how they feel and adapt instruction. Then there was an overview of some Artificial Intelligence (AI) opportunities. One particular opportunity is the design of student models, which is a machine learning problem, low-level data features are extracted then the model is created; to evaluate the model fit they look at student state. Another project is modelling creativity in educational environments. The third opportunity is tailored responses which uses applied dynamic decision networks to tailor example selection. Tailoring feedback based on student progress. The fourth and final opportunity outlined was educational data mining which is using Baysian parameter learning to analyze the utility of hints in a tutoring system. Then to use Natural Language Processing (NLP) to shed light on relationship between creativity and language proficiency. The talk then progressed to talking about an overview of tutoring systems. The current approach to building tutoring systems is to build basic tutoring systems with limited adaption, evaluate, revise and evaluate. Only adding advanced adaption techniques once learning from basic system is evaluated. This is a interdisciplinary approach between AI, human computer interaction and educational psychology. The talk then highlighted three specific projects: computer tutor design, instructional video design and programming and mindset. The target domain for all of this work is teaching programming. Computer tutor design is highlighted first with a look at a tool that is a foundational skill of programming: code tracing. Students who code trace do better than students who do not code trace. Many students do not code trace effectively. The long term research agenda of this project is to build an adaptive tutor to help students learn to code trace. The design criteria of this project is to support the way students code trace on paper and provide assistance for it. The first study was: how does assistance level impact learning and performance on code tracing problems? Assistance was operationalized in two forms: interface scaffolding (high vs reduced) and example presentation order (before or after problem). The methods for this study was 97 participants who studied 4 examples and solved 4 corresponding problems; learning measured by posttest – pretest. 2X2 between-subjects design, manipulating interface scaffolding and example / problem order. The hypothesis was high scaffolding > reduced scaffolding. The inferential statistics was marginal main effect of interface scaffolding. Additional analysis was mind the tutor log files to extract performance data (e.g. time, error, attempts) Used unsupervised learning to cluster students based on behaviours in the tutor. Identified 4 clusters via k-means. Checked for learning differences between the clusters. The key results of this study was 4 student profiles. Unproductively struggling students, Productive problem solvers, Productively-struggling cluster, and Dedicated problem solvers. The implications and next steps was high scaffolding was not as helpful as anticipated when all students were considered. A second code tracing tutor study was then outlined. The method was 45 participants studied 2 examples and solved 2 corresponding problems; learning measured by pretest → posttest gain. Participants verbalized mental processes; all sessions were transcribed and analyzed. The qualitative code scheme was human raters identified constructive verbalization related to code tracing in the transcripts. General reasoning, flow of execution, updating variable values. The summary of the key results was number of reasoning events during problem solving; number of reasoning events during example studying. No significant difference in learning. The implications was in both studies students significantly improved from pretest to posttest. The next step was to redesign the code tracing tutor to use CTAT tutor-building framework. A third study is getting underway soon which is the impact of example design. A takeaway from this part of the talk was programming is challenging for a variety of cognitive and effective reasons. The talk then left the world of code tracing tutor to move to instructional video design. The goal of this area is to evaluate the impact of instructor visual presence and presentation style in instructional videos. The method for this

study was 77 undergraduate students participated in this study. There was a between-subjects design with three conditions: narration only, monologue and dialog. The implications was no evidence that instructional video presentation style influenced learning and related psychological variables. The final area that the talk covered was programming and mindset. The goal of which is to evaluate if mindset can be changed by a brief intervention and if that will improve programming performance. The method for which was a between-subjects design: some participants got the mindset intervention, others got a reading on study skills. reading activity, programming tutorials, programming exercise. A conclusion of which is intervention increased programming actions. A general summary of the talk was tutoring systems and related learning interventions have potential to help student learning. A holistic set of intercentions / design choices is needed