What's Most Broken? A Tool to Assist Data-Driven Iterative Improvement of an Intelligent Tutoring System

Mononito Goswami*
Delhi Technological University
New Delhi, India
mononitog@hotmail.com
+91-8800592994

Shiven Mian*
IIIT-Delhi
New Delhi, India
shiven15094@iiitd.ac.in
+91-8130562083

Jack Mostow
Carnegie Mellon University
Pittsburgh, PA, USA
mostow@cs.cmu.edu

Abstract

Intelligent Tutoring Systems (ITS) have great potential to change the educational landscape by bringing scientifically tested one-to-one tutoring to remote and under-served areas. However, effective ITSs are too complex to perfect. Instead, a practical guiding principle for ITS development and improvement is to fix what's most broken. In this paper we present SPOT (Statistical Probe of Tutoring): a tool that mines data logged by an Intelligent Tutoring System to identify the 'hot spots' most detrimental to its efficiency and effectiveness in terms of its software reliability, usability, task difficulty, student engagement, and other criteria. SPOT uses heuristics and machine learning to discover, characterize, and prioritize such hot spots in order to focus ITS refinement on what matters most. We applied SPOT to data logged by RoboTutor, an ITS that teaches children basic reading, writing and arithmetic.

INTRODUCTION

An Intelligent Tutoring System (ITS) is a computer system that enables learning in an effective and meaningful manner by providing personalized instruction to learners. Prior research has demonstrated that ITSs take extensive time to author, with reported estimates of 200-300 hours of development per hour of instruction (Aleven et al. 2006). As a result, many authoring tools such as the Cognitive Tutor Authoring Tool (CTAT) have been built to make ITS development more efficient. But, despite our growing understanding of human cognition and the tutor authoring process, developing effective tutoring systems with limited development resources remains hard. In this situation, a practical guiding principle towards ITS development and improvement is to fix what is most broken.

The design of RoboTutor (RoboTutor 2015) was inspired by this principle. RoboTutor is an ITS developed as an open-source Android application that teaches children aged 7-10 in developing countries basic reading, writing, and arithmetic. In this paper, we address the question: is there a way to use data from RoboTutor to automate discovery of design issues and highlight them? To best address this issue, we must answer the following subquestions pertaining to the development and design process of RoboTutor:

- **Reliability:** How often and under what conditions does RoboTutor crash or hang? How fast does a child recover from the crash or hang?
- **Recognition**: How accurately does RoboTutor recognize both written and spoken input?
- **Usability:** How easily and efficiently can children operate RoboTutor? Which activities do they find hard to navigate?
- **Engagement:** When are the children disengaged the most, and why?

To answer these questions we present Statistical Probe of Tutoring (SPOT): an Educational Data Mining tool intended to help ITS developers identify 'what's most broken' – i.e, 'hot spots' with respect to design criteria such as software reliability, student engagement etc. SPOT uses metrics to identify and predict the occurrence of undesirable events, and trains a decision tree to discover hot spots. A hot spot is a subtree with a high proportion of undesirable events.

The intuitions that inspired this approach are as follows:

- Within a decision tree, undesirable events in the same subtree are likely to have the same underlying cause.
- The feature combination associated with a subtree - that is, the sequence of tests from the root to the subtree - characterizes when the undesirable events tend to occur and may reflect this underlying cause.
- Screen capture videos of a random sample of undesirable events in the subtree may shed further light on a certain cause and inspire ideas for how to address it.

Henry et al (Henry, Lin, and Park 2017) designed a UI/UX for SPOT and prototyped it using simulated data. Here we describe an implementation that uses automated decision tree learning and runs on real data from RoboTutor. We briefly describe the SPOT workflow and its findings on RoboTutor data in the next sections.

Methodology

Data Collection

RoboTutor uses a variety of activities such as story reading, math activities etc. to teach children. The dataset used by SPOT to discover hot spots is derived from student attempt

*authors contributed equally

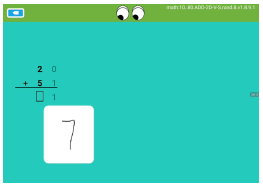

Figure 1: RoboTutor mis-recognizing 7 as 1 because of absence of middle stick - instance provided by SPOT

performance logs recorded from RoboTutor's beta field testing sites in Tanzania. We also extract activity and item-level aggregates from these performance logs. SPOT also uses screen capture videos of RoboTutor recorded using AZ Screen Recorder[1], used by developers to further examine the obtained hot spots.

Approach

To automate discovery of design issues, SPOT uses a work flow summarized as follows:

1. Developers specify a design criterion, and pick metric(s). A design criterion is any property of an Intelligent Tutor that we wish to analyze. We classify design issues into one of these design criteria: Reliability, Recognition, Usability and Engagement. Using pre-defined metrics (heuristics), SPOT approximately labels each row of the data as: suspicious instance indicating a design issue, or a non-suspicious instance. For example: the rejection rate of responses in writing activities is a metric of writing recognition, and a suspicious instance is any written response other than the expected answer.
2. SPOT trains a decision tree on the labeled data and discovers the top N hot spots using the F_1 score for each subtree. SPOT uses features such as activity name, student input, expected answer, attempt duration etc, with different features for attempt, item and activity-level data. The F_1 score enables SPOT to discover hot spots in the decision tree and balances both the concentration and the number of suspicious instances in the hot spots.
3. SPOT automatically characterizes each of the hot spots by conjoining every test on the path from the root node of the tree to the hot spot. In our work, we use Decision Trees strictly for finding and characterizing hot spots, rather than classifying unseen data.
4. SPOT picks up a random sample of suspicious instances of a hot spot, and presents their corresponding screen capture videos scrolled to the start of the instances along with the hot spot characterization, to the developers. Developers examine the videos and characterization for clues to diagnose the design issue, and devise solutions for redesigning RoboTutor.

Figure 1 illustrates a screen shot from the screen capture videos for a recognition hot spot characterized as: 'expected answer = 7'. After examining the sample of screen capture videos from this hot spot, we realized that the writing recognizer often mis-recognizes 7 in the absence of its middle stick as 1, leading children to confuse between the two.

[1]http://bit.ly/azurl

Conclusions

Table 1 presents some key SPOT findings and corresponding design implications for RoboTutor. We evaluated SPOT and selected features qualitatively, since design changes are fundamentally subjective decisions on part of the developers.

Criterion	SPOT Findings	Design Implications
Reliability	Counting and Story Reading actvities have high crash rates.	Implement crash logging, Examine Counting and Story Reading activities for bugs
Recognition	Children confuse between number pairs like 1 and 7, 5 and 3.	Bias data sources of appropriate activities towards frequently confused digits for better practice
Usability	Children spend unusually long time in Story Reading activities	Add timeouts to story reading activities
Engagement	Children bail out of activities when given the same problems repeatedly	Children should not be given the same problems repeatedly

Table 1: SPOT Findings and Implications

We believe SPOT would benefit many tutor developers by helping them iteratively improve their tutors and save time. SPOT may be especially useful when user-testing in person is impractical, such as in situations where the users are far away, when children may behave differently when adults are present, and when hot spots are important to fix but too rare to observe in person. SPOT can also be used for other ITSs, since it relies on simple data sources such as log files and screen record videos, both of which are easy to record.

Acknowledgements

Special thanks to other members of the RoboTutor team, especially Evelyn Yarzebinski for data parsing and cleaning. This work was sponsored by the Robotics Institute Summer Scholars (RISS) program, Carnegie Mellon University.

References

Aleven, V.; McLaren, B. M.; Sewall, J.; and Koedinger, K. R. 2006. The cognitive tutor authoring tools (ctat): Preliminary evaluation of efficiency gains. In *International Conference on Intelligent Tutoring Systems*, 61–70. Springer.

Henry, G.; Lin, J.; and Park, C. Y. 2017. SPOT: Refining robotutor. *HCI senior capstone project presentation, Carnegie Mellon University*.

RoboTutor. 2015. RoboTutor. http://robotutor.org.