Runestone's Question Bank, Exam Generator, and Log Data

Barbara J. Ericson
University of Michigan, School of Information
Ann Arbor, Michigan, USA
barbarer@umich.edu

Bradley N. Miller
Luther college / Founder Runestone Interactive
Minneapolis, Minnesota, USA
brad@runestoneinteractive.com

ABSTRACT

Runestone is an open-source platform for interactive ebooks. It serves over 120,000 registered learners and has an average of 350,000 page views a day. There are ebooks for secondary computer science as well as for undergraduate computing courses: CS1, CS2, data science, web programming, and more. The platform supports executable and editable examples in Python, Java, C, C++, HTML, JavaScript, Processing, and SQL. It also includes code visualizers/steppers for Python, Java, and C++ code. Runestone supports instructional material: text, videos, and images as well as typical practice problems with immediate feedback such as multiple-choice, fill-in-the-blank, and matching questions. Runestone also has unusual features such as audio tours of code, clickable areas, adaptive Parsons problems, and a unique practice tool. This paper highlights the new exam generation feature which also allows A/B testing, explains the log data that is collected and is available for analysis, and describes plans for future development including making the smart content reusable.

Author Keywords

on-line learning, ebooks, adaptive learning, practice tools, intelligent ebooks, Parsons problems, ACOS, LTI

INTRODUCTION

Brad Miller started creating the Runestone ebook platform in 2011 with a goal of *"democratizing textbooks for the 21st century"*. Brad wanted ebooks to be free so that everyone could learn computer science, even if they could not afford a $200 textbook. The first feature he added to the platform was the ability run and edit code. He also integrated a code visualizer / stepper [4]. Over the years many more typical interactive features with immediate feedback were added including multiple-choice questions, fill-in-the blank questions, and matching.

UNIQUE INTERACTIVE FEATURES

Runestone has several unique interactive features, some of which (clickable area problems, adaptive Parsons problem, and a spaced practice tool) we have reported on before [2, 8, 3]. Due to space constraints we will only discuss the newest features here, which are the ability to generate exams from a question bank and A/B testing.

EXAM GENERATOR

The exam generator allows an instructor to specify one or more questions for an exam that utilize a question bank. Questions are selected from the question bank and presented to the student as they take the exam. Taken to its extreme this could provide a unique exam for each student. However an instructor can also specify a list of equivalent questions and the exam generator will pick a random question for each student.

To create an exam the instructor writes questions using the `.. selectquestion::` directive in Runestone. This directive supports the following options:

- `:fromid: id1, id2, id3, ...` Each id refers to an existing question in the question bank. Runestone randomly selects an id for each student.
- `:proficiency: prof1, prof2, ...` If the questions in the question bank are tagged with a proficiency (competency), Runestone will randomly select a question from the question bank that is tagged with the specified proficiency.
- `:min_difficulty: 1.25` Specifies the minimum difficulty for the questions to be selected.
- `:max_difficulty: 5.0` Specifies the maximum difficulty for the questions to be selected.
- `:not_seen_ever:` Ensures that the student is only presented with a question they have not seen before.
- `:autogradable:` Only selects questions that can be graded automatically.
- `:ab:` Allows A/B testing and is followed by a unique experiment id

Currently the questions in the Runestone question bank come from two sources: book authors and instructors. The entire corpus of questions is available from a GitHub repository and contains approximately 19,000 questions covering a range of topics including introductory programming in Python, Java, and C++, web programming, and database design. Each question is stored in a file in restructuredText format. However, there are many duplicate questions and many questions that were contributed by instructors that may not be ready for use. A database of questions can be populated by simply running a runestone build from the top level of the QuestionBank folder.

Seventh SPLICE Workshop at SIGCSE 2021 "CS Education Infrastructure for All III: From Ideas to Practice", SPLICE'21, March 15-16, 2021, Virtual Event

Each question includes metadata to facilitate question selection by the exam generator. The metadata includes difficulty, topic, proficiency (competency) tested, the number of students that have attempted the question, the number of students that got the question correct, the percentage of students that got the question correct on their first try, and the mean number of attempts for the student to get the question correct.

For questions such as multiple choice and fill-in-the-blank we compute a difficulty score on a scale from 1 to 5 by re-scaling the percent of students that got the question correct on their first try. For coding questions with unit tests we re-scale the mean attempts to get the question correct to the 1 to 5 scale.

Currently we use the topic meta data as a proxy for the competency that the question tests. Since Runestone books use a common chapter / subchapter structure we use the chapter label and subchapter label to form a topic. For example, the chapter on conditionals has a subchapter labeled *LogicalOperators*. Using the select question directive one can specify a proficiency of "conditionals/logicaloperators". This is not meant to be a long term solution for competencies. It would likely be better to have a formal taxonomy and tag each question with one or more competencies from the taxonomy.

A/B TESTING

A/B testing allows researchers to run experiments in which learners are automatically assigned to one of two conditions (A or B). This is accomplished with the *selectquestion* directive and *ab* option followed by a unique experiment id. The instructor must provide two questions in the *fromid* option for each quesion in the experiment. The first question will be used for condition A and the second for condition B.

LOG DATA

Runestone logs every user interaction in the ebook. Each log entry includes the date and time (timestamp), user identifier (sid), event, data about the event (act), item identifier (div_id), course identifier (course_id), base course identifier (base_course), chapter, subchapter, and institution identifier (anon_institution). It logs page views, answers to any of the practice problems, video plays, audio tour plays, and every code edit/run and block move in a Parsons problem. Part of a log file is shown in Figure 1. All of the code that is executed or saved is also available in a separate log.

Researchers can request an anonymous log file from Brad Miller or wait for a snapshot of the anonymized Runestone log file data base to be integrated into the SPLICE data set

FUTURE WORK

Our plans for future work include further development of the question bank and exam generation system, making anonymous log file data and the question bank available to the SPLICE research community, support for both in-person and remote Peer Instruction, and new research on Parsons problems.

The question bank is in its infancy and needs a team of editors to eliminate or improve bad questions. We need to develop a taxonomy of competencies for Python, Java and C/C++ and

timestamp	sid	event	act	div_id
8/21/17 23:54	27	Audio	Line-by-line Tour	lcfc1
8/21/17 23:54	27	Audio	play	lcfc1
8/21/17 23:54	27	Audio	closeWindow	lcfc1
8/21/17 23:57	23	video	play	ants
8/21/17 23:57	25	mChoice	answer:3:correct	q2_2_1
8/21/17 23:57	25	mChoice	answer:1:correct	q2_2_2
8/21/17 23:58	25	mChoice	answer:3:no	q2_2_3
8/21/17 23:59	25	mChoice	answer:0:correct	q2_2_3
8/22/17 0:02	26	page	view	index.html
8/22/17 0:03	26	page	view	JavaBasics/\
8/22/17 0:03	25	activecode	edit	lcfc1
8/22/17 0:04	25	parsonsMove	start|0_1_0-5_0-6_0-2_3_0-4_0|-|c0	thirdClass
8/22/17 0:04	25	parsonsMove	move|5_0-6_0-2_3_0-4_0|0_1_0|c0	thirdClass
8/22/17 0:04	25	parsonsMove	move|5_0-6_0-4_0|0_1_0-2_3_0|c0	thirdClass
8/22/17 0:04	25	parsonsMove	move|5_0-6_0|0_1_0-2_3_0-4_0|c0	thirdClass
8/22/17 0:04	25	parsonsMove	move|6_0|0_1_0-2_3_0-4_0-5_0|c0	thirdClass
8/22/17 0:04	25	parsonsMove	move|-|0_1_0-2_3_0-4_0-5_0-6_0|c0	thirdClass
8/22/17 0:04	25	parsons	correct|-|0_1_0-2_3_0-4_0-5_0-6_0|c1-s	thirdClass

Figure 1. Part of a log file from Runestone that has been anonymized.

then annotate the questions with the appropriate competencies. We also need to build visualization tools to help instructors verify that their exams are covering the competencies. Research also needs to be done on students perceptions of the fairness of competency-based generated exams.

We will make anonymous log files available to the SPLICE community and will also develop a representation of the questions in the question bank using JSON that is compatible with ACOS to encourage question reuse.

We will also leverage the question bank to add support for both in-person and remote Peer Instruction. In Mazur's Peer Instruction students read material before lecture and take an assessment based on the reading either before or at the beginning of lecture [1]. In lecture the instructor displays a difficult multiple-choice question that contains distractors (incorrect answers) based on common misconceptions. The students answer the question individually (vote), then discuss their answers with neighboring students (peers), and then answer (vote) individually again. Finally, the instructor shows the result of the two votes and leads a discussion of the question [1]. Peer Instruction improves student engagement, retention, and learning over traditional lecture [7, 6, 1]. We will use the question bank to identify good questions for Peer Instruction. A good question for Peer Instruction is one that about 40-60% of the students get wrong on the first vote [5]. If students are remote and synchronous they will use a chat-style interface to discuss the Peer Instruction question. If students are asynchronous we plan to serve them the saved chat from a synchronous learner.

An end of course student survey from a course taught by Ericson during Fall 2019 that used both Parsons problems and write code problems as lecture exercises found that 78.3% of students agreed or strongly agreed that they found the mixed-up code (Parsons) problems in lecture practice helpful for learning. However, 36.2% of the students would rather write the code from scratch than solve a Parsons problem. We are adding the ability to choose to write the equivalent code with unit tests rather than a presented Parsons problem. We have also been exploring ways to cluster student written code in order to generate Parsons problems.

REFERENCES

[1] Catherine H Crouch and Eric Mazur. 2001. Peer instruction: Ten years of experience and results. *American journal of physics* 69, 9 (2001), 970–977.

[2] Barbara Ericson. 2019. An Analysis of Interactive Feature Use in Two Ebooks. In *iTextbooks@ AIED*. 4–17.

[3] Barbara J Ericson and Bradley N Miller. 2020. Runestone: A Platform for Free, On-line, and Interactive Ebooks. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 1012–1018.

[4] Philip J Guo. 2013. Online python tutor: embeddable web-based program visualization for cs education. In *Proceeding of the 44th ACM technical symposium on Computer science education*. 579–584.

[5] Nancy Kober. 2015. *Reaching students: What research says about effective instruction in undergraduate science and engineering.* National Academies Press.

[6] Leo Porter, Dennis Bouvier, Quintin Cutts, Scott Grissom, Cynthia Lee, Robert McCartney, Daniel Zingaro, and Beth Simon. 2016. A multi-institutional study of peer instruction in introductory computing. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. 358–363.

[7] Cynthia Taylor, Jaime Spacco, David P Bunde, Andrew Petersen, Soohyun Nam Liao, and Leo Porter. 2018. A multi-institution exploration of peer instruction in practice. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. 308–313.

[8] Iman YeckehZaare, Paul Resnick, and Barbara Ericson. 2019. A Spaced, Interleaved Retrieval Practice Tool that is Motivating and Effective. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. 71–79.