Creating Session Data from eTextbook Event Streams

Samnyeong Heo[1], Mohammed F. Farghally[1], Mostafa Mohammed[2] and Clifford A. Shaffer[1]

[1]Department of Computer Science, Virginia Tech, Blacksburg VA, USA 24061
[2]Department of Computer Science, Assiut University, Assiut, Egypt

Abstract

As more students interact with online learning platforms and eTextbooks, they generate massive amounts of data. For example, the OpenDSA eTextbook system collects clickstream data as students interact with prose, visualizations, and interactive, auto-graded exercises. A large class with a lot of system usage could easily generate in excess of a million events during a semester. Ideally, instructors and system developers can harness this information to create better instructional experiences. But in its raw event-level form, it is difficult for developers or instructors to understand student behaviors, or to make testable hypotheses about relationships between behavior and performance. In this paper, we describe our efforts to break raw event-level clickstreams first into sessions (a continuous series of work by a student) and then to meaningfully abstract the events into higher-level descriptions of that session. We first bundle events into related activities, such as the events associated with stepping through a given visualization, or with working a given exercise. Each such group of events defines a state. A state is a basic unit that can characterize the interaction log data, and there are multiple types including reading, slideshows, and exercises state. We hope that this process of creating semantic-level abstractions of work sessions can be a useful step in the analysis pipeline to give us better insight into students' learning behaviors.

Keywords

eTextbooks, work sessions, clickstream data, student analytics

1. Introduction

Online educational resources are becoming increasingly sophisticated. Within CS, there is widespread use of programming environments both for large and small programs (projects vs. exercises), interactive exercises, and eTextbooks. One goal of these systems is to increase the level of student interaction over paper textbooks, lectures (live or video), and paper homework with slow feedback cycles. They can also reduce grading burdens for instructors, as the online activities are typically graded automatically.

If we want to increase the level of student engagement with the material, monitoring the actual amount of engagement and the details of the type of engagement are likely to be important. Fortunately, these systems make it easy to collect detailed event-level data about how students interact with the system, including the time distribution of the interactions (such as a little, a lot, many small sessions, a few large sessions, etc.), and the behavior (do students read the

Fifth Workshop on Intelligent Textbooks, July 2023, Tokyo, Japan
hsn1017@vt.edu (S. Heo); mfseddik@vt.edu (M. F. Farghally); mkosman@aun.edu.eg (M. Mohammed); shaffer@vt.edu (C. A. Shaffer)

material and then work the assessment, or do they go to the assessment and only look at the material as necessary to get credit?).

On the other hand, the basic unit of data collection is not these higher-level semantic actions, but rather an event such as a button click, typing a character or unit of text, opening or leaving a page. A large class with a lot of system usage could easily generate in excess of a million events during a semester. How can instructors or researchers hope to make practical use of this fire hose of data?

In this paper, we describe our efforts to organize the event click stream from use of an eTextbook system, first into individual "sessions" of use, and then into higher levels of abstraction for the activity in the session.

2. Event Streams

Many researchers have used event stream data to assess students' detailed learning processes and progress to figure out how students were interacting within the infrastructure. For example, Baker et al [1] analyzed these data to help students achieve better performance in a class. Successful results from previous research and analysis open up new techniques and opportunities for instructors to tailor their educational tools for learners. Tax et al. proposed Mining Local Process Models (LPM) [2] to discover frequent behavior patterns in event logs, typically between three and five types of events. Local process model mining focus on tracing partial behaviors focusing on frequently occurring patterns. Carter et al. [3, 4] proposed the Normalized Programming State Model (NPSM). The goal is to gain a better idea of students learning process and to benefit from advanced predictions of students' course performance. The authors suggested a holistic predictive model of student performance which characterizes student's programming activity in terms of different states, and derived a formula that can predict students' performance in a programming course. The authors investigated the relationship between the paths that students take through the programming states defined by the NPSM and their overall course achievement. They examined the correlation between the frequency of the most common transition paths and students' overall performance in a CS2 course.

OpenDSA [5] is an eTextbook used in a number of courses in the Computer Science department at Virginia Tech and other institutions around the globe. It supports various computer science-related topics including Data Structures and Algorithms, and provides algorithm visualizations, interactive exercises such as small code writing problems, and proficiency exercises. The platform collects a stream of learning process data and stores these in its database. This includes event-level data such as when a user clicks a button, switches between pages, or gets a score on an exercise. Additional information on the OpenDSA architecture can be found in [5].

Later in this paper, we discuss various behaviors that can be recognized from event streams. We analyze data from at two different courses, that we label Data Structures and Algorithms (DSA), and Formal Languages and Automata (FLA). The DSA course typically has hundreds of students, and the FLA theory course typically has about 80 students. Most of the students in these courses are undergraduates. In this paper, we primarily take our examples from log data for a (FLA) course given in Fall 2020 and Spring 2021 and a Data Structures and Algorithms (DSA) course given in the same semesters.

Since there are some qualitative differences in the eTextbook materials for these courses, looking at the two lets us make some comparisons. The DSA materials are primarily a mix of prose, visualizations, programming exercises, multiple-choice question banks, and proficiency exercises that require manipulating a graphical representation of a data structure [5]. In contrast, the eTextbook for the FLA course [6] relies heavily on the pedagogical instructional approach known as Programmed Instruction, with variations known as Computer-Assisted Instruction (CAI), and the Keller Plan [7, 8]. We implement this in the form of slideshows, known as PI Framesets (and hence, a single slide is called a PI Frame). A PI frame generally contains only a couple sentences of information. Many of the PI frames (ideally, over half) also contain a question of some sort. These might be multiple choice (where there is one correct selection), True/False, or a more difficult multiple choice variant where several of the possible choices have to be selected to be correct. Often the questions are easy, with the primary purpose of forcing attention to the material through some user interaction. Occasionally the questions force the student to work harder to reflect on the information in the slideshow.

3. Abstracting Session Data

We find that higher-level understanding is hampered as much as helped by the overwhelming mass of clickstream data to be analyzed. In order to get to the level of understanding student's semantic-level behavior, we need to organize and abstract the event-level data first. We find that a key organizing unit of behavior is the "session". A session is a period of use of the eTextbook. But recognizing which events form a session is not trivial.

The ability to group raw events into sessions, and then to abstract those sessions into a series of semantically meaningful actions should be a useful step toward answering many questions about student use of eTextbooks. Here are a few examples.

- What kinds of study patterns can we observe? Specifically, what kinds of transitions within modules can we observe?
- Which study behaviors will yield the best performance in a student's overall grade in a course?

While the purpose of this paper is not to show answer such research questions, our techniques for session analysis should be a useful step in a general analysis pipeline.

3.1. Defining Sessions

Study sessions are defined to be a continuous stream of interactions without a significant time gap. For example, one study session can start when a student opens the module and end when the student closes it. There are exceptional cases we need to address when considering study sessions. If a student leaves the module open for a while, meaning that we do not find a closing window interaction, do we consider that the student keeps studying, or are they away from the keyboard? If we set the time limit too short for deciding that the student is no longer working, for example to 1 minute, the student might have been reading materials on the screen without generating recognizable events. If we set the gap threshold too long, multiple days of records

can be considered as a single study session. We start with a value that we selected (and some discussion below to support this choice), but our analysis software can easily redefine sessions using another threshold.

Figure 1 shows the number of instances based on the time differences between two consecutive events of a student. The figure disregards delta times less than 60 seconds due to its massive volume, and it is convincing that a student was actively interacting if generated events are recorded with less than a 60 second gap.

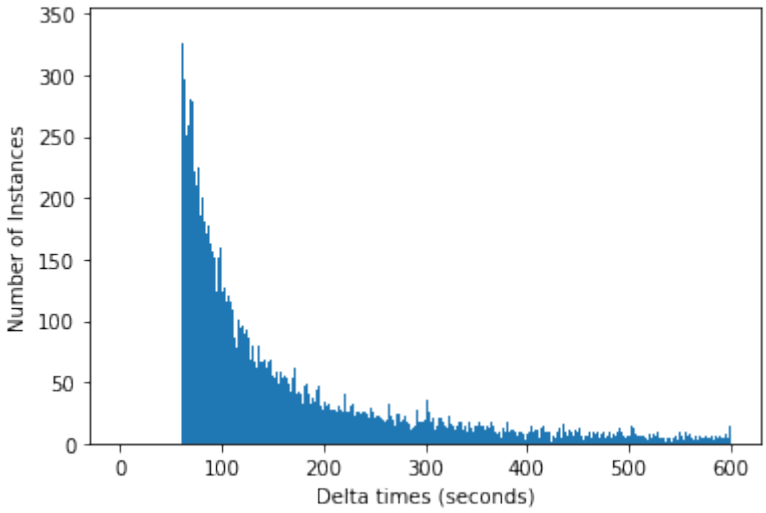

Figure 1: Histogram of delta time between two consecutive events of all interaction data in FLA for Spring 2021. Delta times less than 60 seconds are not shown in the graph due to massive volume.

Delta time (for gaps > 60 seconds) has the 25th quartile at 88 seconds, 50th quartile at 163 seconds, and 75th quartile at 599 seconds, or about 10 minutes. We compared the results of defining sessions with thresholds of 10 minutes and one hour, and checked how sensitive the results are to these values. We find that the results are nearly identical for 10 minutes or one hour (or anything in between), so we choose the threshold as 10 minutes in all analyses.

Before we cab gemerate user-sessions for a course, we first need to pre-process the raw event data. In the case of OpenDSA, events of various types are stored in several database tables, depending on the type and source. This is probably typical for many educational systems. Ideally, all of the event types can be extracted for each individual user and placed in a consistent timestamp order. In our case for the Formal Languages course, we might have a variety of event types, such as those from proficiency exercises (shown in Figure 2) and programmed instruction framesets (Figure 3). Both tables contain (anonymized) user IDs, timestamps, exercise names, and other necessary information for the analysis.

After collecting the data from the various tables for a given course, we use a custom python script to merge them into a single spreadsheet sorted by user ID and then by event time. Now we have a massive sorted raw stream of data with all information about the students' interactions. The output raw data of the Formal Languages Spring 2021 course alone generates about 1,640,000 rows with 12 columns for 65 students.

	id	user	question	type	time_done	section_id	sec_ex_id
2508	8680925	812	CharacterizeLang1p	attempt	2021-02-01 18:19	143543.0	242940
2509	8680936	812	CharacterizeLang1p	attempt	2021-02-01 18:21	143543.0	242940
2510	8680940	812	CharacterizeLang1p	attempt	2021-02-01 18:21	143543.0	242940
2511	8680944	812	CharacterizeLang1p	attempt	2021-02-01 18:21	143543.0	242940
2512	8680947	812	CharacterizeLang1p	attempt	2021-02-01 18:21	143543.0	242940
2514	8680958	812	CharacterizeLang2p	attempt	2021-02-01 18:22	143543.0	242941
2515	8680961	812	CharacterizeLang2p	attempt	2021-02-01 18:22	143543.0	242941
2516	8680962	812	CharacterizeLang2p	attempt	2021-02-01 18:22	143543.0	242941
2517	8680965	812	CharacterizeLang2p	attempt	2021-02-01 18:23	143543.0	242941
2518	8680968	812	CharacterizeLang2p	attempt	2021-02-01 18:23	143543.0	242941

Figure 2: Example of odsa_exercise_attempts from Formal Languages

	id	user	frame	question	correct	created_at
1637	101248	812	LanguagesFS	0	0	2021-01-22 1:37
1638	101249	812	LanguagesFS	0	1	2021-01-22 1:37
1639	101250	812	LanguagesFS	1	1	2021-01-22 1:38
1640	101251	812	LanguagesFS	2	1	2021-01-22 1:38
1643	101254	812	LanguagesFS	3	1	2021-01-22 1:39
1645	101256	812	LanguagesFS	4	1	2021-01-22 1:40
1649	101260	812	LanguagesFS	5	1	2021-01-22 1:40
1650	101261	812	LanguagesFS	6	0	2021-01-22 1:40
1651	101262	812	LanguagesFS	6	1	2021-01-22 1:40
1652	101263	812	LanguagesFS	7	1	2021-01-22 1:40

Figure 3: Example of pi_attempts from Formal Languages

3.2. Abstracting Sessions

Given a definition to determine events that comprise a session, we next would like to abstract these into semantic-level behavior. We would like to recognize behaviors such as reading content, navigating through slideshows, or solving exercises. Carter et al. [3] developed the Normalized Programming State Model (NPSM), which characterizes students' programming activity in terms of the dynamically changing syntactic and semantic correctness of programs. Inspired from the way Carter's study defined programming activities as being in one of a small number of states, we characterize a series of events on the same content element as a state, such as document state, proficiency exercise state, and frameset state. Each event type is classified as part of one of a few pre-defined states.

With raw events split into sessions, the next step toward semantic information is to combine

consecutive similar events within the session. We bundle the consecutive events for a single interface element (a state) into a single row, and we keep track of the total time spent on the events, the number of events, an event description, and the type of exercise. For example, a series of clicks within a slideshow can be bundled into a single line, showing summary information like the total time elapsed to complete these events and a total number of slideshow events recorded consecutively.

Figure 4 shows an output of abstracted data after processing the raw event stream data from Formal Languages in Spring 2021. The figure shows that user id 812 starts the first study session on January 21st by opening Module 01.03 and then moving on to the LanguagesFS frameset. In the short span of 4 seconds, the user generates one event associated with the window state and three events associated with the frameset state. On January 22nd, the user starts the second study session by opening the same module, working on the same slides for 43 seconds, then has a window close followed by a window open 72 seconds later. Following that, the user spends 288 seconds interacting with the LanguageFS slide, generating 27 slide events. We can grasp the idea of how frequently and how diligently students study over the semester by looking at the abstracted interaction data. Of course, we can always refer back to the raw event stream data when a deeper analysis is required, but the abstracted interaction data gives instructors a higher level of understanding as to how students behave.

session	user	Event	Event Description	Start time	End Time	Action Time	Type	# events
2	812	document event	"User loaded the...	2021-01-22 01:37:37	2021-01-22 01:37:37	NaN	NaN	1
2	812	window event	"User closed or ...	2021-01-22 01:37:37	2021-01-22 01:37:37	NaN	NaN	1
2	812	document event	"User loaded the...	2021-01-22 01:37:41	2021-01-22 01:37:41	NaN	NaN	1
2	812	Window open	"User looking at...	2021-01-22 01:37:42	2021-01-22 01:37:42	Reading time: 1....	NaN	1
2	812	FF event	Attempted to sol...	2021-01-22 01:37:43	2021-01-22 01:38:26	43.0 seconds	LanguagesFS	4
2	812	Window close	"User is no long...	2021-01-22 01:38:30	2021-01-22 01:38:30	Away time: 72.0 sec	NaN	1
2	812	Window open	"User looking at...	2021-01-22 01:39:42	2021-01-22 01:39:42	Reading time: 12...	NaN	1
2	812	FF event	Attempted to sol...	2021-01-22 01:39:54	2021-01-22 01:44:42	288.0 seconds	LanguagesFS	27
2	812	Window close	"User is no long...	2021-01-22 01:44:46	2021-01-22 01:44:46	Away time: 18.0 sec	NaN	1
2	812	Window open	"User looking at...	2021-01-22 01:45:04	2021-01-22 01:45:04	Reading time: 36...	NaN	1
2	812	FF event	Attempted to sol...	2021-01-22 01:45:40	2021-01-22 01:47:41	121.0 seconds	GrammarIntroFS	9

Figure 4: Abstracted interaction data from Formal Languages

3.3. Data Compression

One side effect of abstracting session data is that it greatly reduces the size of the information represented. We characterize an approximate compression ratio between the number of rows in event-level data and the number of rows in abstracted session-level data by presenting statistics about how big these data sets are in terms of the number of rows. Table 1 shows the number of rows for both raw and abstracted data, the compression ratio as well the total number of sessions abstracted. The compression ratio was measured by calculating the percentage decrease.

$$\text{Compression Ratio} = \frac{\text{\# of rows in raw data} - \text{\# of rows in session data}}{\text{\# of rows in raw data}} * 100$$

About 90 percent of raw data is compressed into abstracted data for the Formal Languages class, and 68 percent is compressed for the Data Structures course. We hypothesize that the variability in the number of sessions depends on how steadily students study in one sitting. The DSA material has more prose interspersed with exercises. The FLA material has most of the information presented in Programmed Instruction framesets (slideshows). One basic behavior that we observe is that students are more likely to attempt to complete a frameset in a single session than they are to read the prose and do the associated exercises in a single session.

Table 1
Compression ratio statistics between a raw event stream data and abstracted data: number of rows.

Class	Raw data	Abstracted data	Compression Ratio	Total # of Sessions
FLA Fall 2020	763,320	101,437	86.7%	3442
FLA Spring 2021	1,638,155	126,412	92.3%	3490
DSA Fall 2020	447,153	140,332	68.6%	5242
DSA Spring 2021	298,138	92,969	68.8%	2543

4. Session Behaviors

Even the session-level data abstractions presented in Section 3 provide rather a lot of information for an instructor to absorb. In this section we consider ways to utilize abstracted session data by characterizing sessions into patterns of behavior.

We propose four example session behaviors that at least some students are likely to exhibit while studying materials in OpenDSA, and discuss how to recognize those behaviors from the session data. At this point we are not prepared to make judgements about relationships between student performance in the class and any of these behaviors, we are just demonstrating how abstracted session data can be used to recognize behavior patterns. A behavior is defined as a stream of transitions from one state to another, and each state represents a collection of associated events, such as a window transition state or exercise state. Researchers can use this sort of session classification process to find dissimilarities in overall performance measures between groups of students according to their study behavior.

Activity Sessions Behavior For our first example, we propose a behavior that would naturally occur if a student opens a module and interacts with exercises or other activities such as slideshows. We count the number of transitions from document state to window state and followed by FF state (for PI framesets) or PE state (for other exercise types) for each student during the whole semester. We can easily detect transitions between states for each unique user ID from abstracted interaction data. For example, when a particular user triggers a document event followed by opening a module event, we look to see if the user transitioned from the window event to the FF event or PE event. If no other events are triggered between transitions, we can assume that the user followed the pattern of jumping straight to an exercise. This might be in contrast to opening a module and reading prose if there is any before trying to solve exercises. Behaviors that we are not considering and therefore not counted include when a

student closes a module without interacting with exercises (we only consider modules that have at least one exercise) or jumps between modules rather than completing one module. Basically, we attempt to find how often the students engage with modules in a way that is possibly learning behavior.

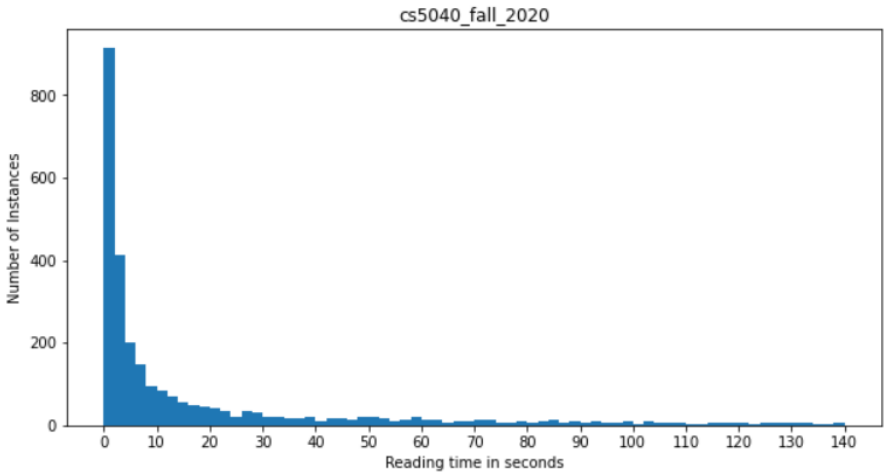

Figure 5: Histogram of reading prose time for CS5040 Fall 2020

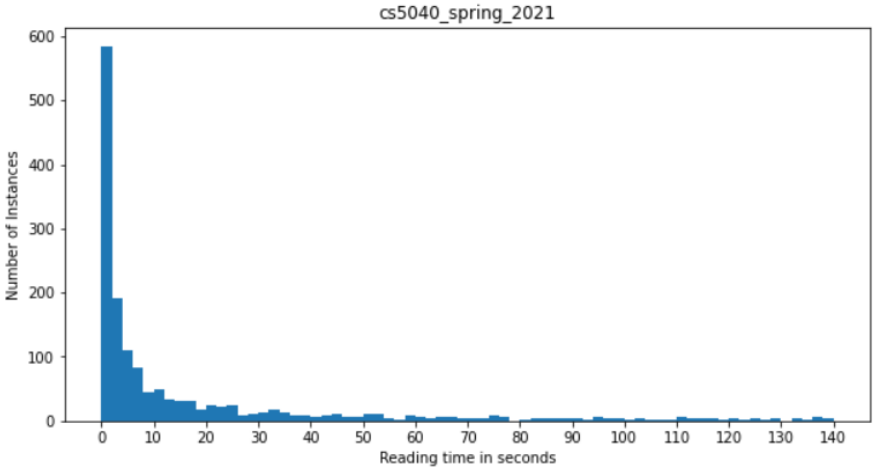

Figure 6: Histogram of reading prose time for CS5040 Spring 2021

Reading Prose Behavior Examining only the number of sessions in which students engage with exercises and slideshows is inadequate to evaluate students' behaviors, so we also consider time. A transition made from a document state to a window state or other states does not guarantee that a student was reading prose because there are other factors that we need to look for, such as students stepping away from the computer while keeping the module open.

We note that using scrolling events would help to figure out which section of a module the individual was paying attention to so that researchers may investigate interaction logs with more detailed information. Unfortunately, this feature was not available in OpenDSA when we collected these data. Instead, we consider the estimated reading time, which is how long students spend time in the text section before jumping into solving exercises or slideshows.

Another version of Reading Prose behavior come from Fouh et al. [9], who proposed that the student reads the slide thoroughly if time spent on a each slide is between 8 and 15 seconds. Less than 8 seconds means that students did not read the text on the slide and advanced to the next slide, and greater than 15 seconds is that the student stepped away from the computer while the slideshow was up. Though this research focus was explicitly on time spent in each slide in a slideshow, we can borrow the idea of setting a threshold and investigating how long students stayed in reading contents.

If a student makes a transition from module opening to PE event or FF event (the two exercise types) within a given time range, we conclude that the person read the prose content. Any transitions that happened in less than or greater than the threshold are not counted as we specifically look for if individuals were actually reading. We draw two histograms of reading prose time of both semesters of DSA classes depicted in Figures 5 and 6. We identify three clusters by setting two dividing points as seen in the figures; 1) Not-reading behavior where the reading time is less than 15 seconds, 2) Reading behavior where the reading time is in between 15 and 120 seconds, and finally, 3) Stepped Away behavior beyond 120 seconds which is assumed that students leave the modules open and generate no further activities.

To validate the accuracy of the time threshold for the Reading Prose behavior, we investigated several modules independently. We used the top six most-used modules by students that contain both prose and exercises. Figure 7 shows the load-to-exercise time (time difference between the initial module opening to the first exercise event) histogram for a module on Shell Sort. In addition to the histogram of the Shell Sort module, all other histograms have shown a cluster that falls between 15 seconds to 120 seconds. Given the consistency of how the clusters were formed in different modules with various lengths of prose, we determined to use that time range as the threshold for the Reading Prose behavior. According to this figure, most students are not reading the explanation for Shellsort prior to jumping to work the exercise, as most of them spent less than 15 seconds between opening the module and doing the first exercise event.

In our abstracted interaction data (see Figure 4), the column named “Action Time" shows the time difference between two consecutive events. If the column value contains the “Reading time" string, it indicates the time between opening the module and clicking on the exercise. We can take advantage of the value to count the number of times students read prose before moving on to the exercise states, which is considered to occur when the “Reading time" is in the threshold range.

Credit Seeking Behavior Our third behavior example is called “Credit Seeking". Previous studies [9, 10] identify such credit-seeking behaviors that differ from and possibly are in conflict with learning behaviors. These include jumping straight to the exercises without reading the text, clicking through slideshows as quickly as possible, and skipping to the end of slideshows.

Our third behavior focuses on recognizing credit-seeking behavior, especially the not-reading

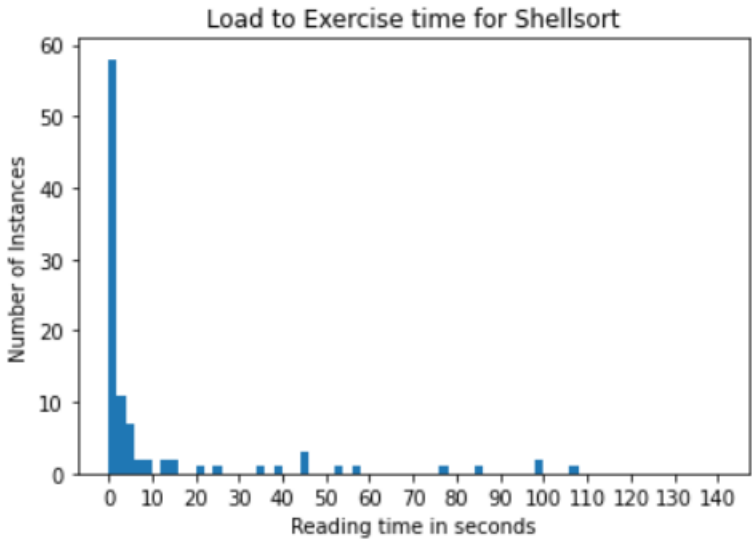

Figure 7: Histogram of reading prose time for Shell Sort Module

patterns. In contrast to our second activity type "Reading Prose", we are interested in finding students who did not read a majority of the text and skipped directly to the exercises. Instructors may give assignments to complete an OpenDSA module. To be specific, students earn credits from completing slideshows and solving exercises, not from reading the material. While instructors typically believe that reading the material is essential to learning, spending time on this might conflict with a student's goal of getting credit with the minimal effort.

Referring back to the histograms from Figures 5 and 6, we use 15 seconds for DSA data as our threshold to identify credit-seeking behaviors whether students skip directly to the exercises. A similar approach from the reading prose behavior is used. We measure the total time a module was opened before a transition was triggered to engage with exercises. If it is less than the threshold of 15 seconds, we consider that as a credit-seeking pattern and record the number of patterns that occurred for each class student.

Note that our rule for recognizing "credit seeking" behavior is not exactly the inverse of "reading prose" behavior, because in both cases we rule out sessions over our threshold that we assume indicates the student is away from the computer. We cannot predict which behavior students are following in those cases. Future analysis that includes scrolling events can hope to distinguish this more accurately.

PI Credit Seeking Behavior We also propose a pattern of behavior related to completing PIFrames, which is PI Credit Seeking behavior. Modules for the FLA text tend not to have prose, but instead have nearly all content contained in Programmed Instruction framesets. PIFrame slides typically contain a brief text statement supported by visual components. Sometimes a frameset deals with complex material. Although PIFrames have been proved to be an excellent instructional method [6], a student may find the topic unattractive, especially for mathematical material. Students might be tempted to skip to the next slide without adequately reading and

understanding the presented context. Since PIFrames do not allow skipping without answering the question on the slide, such students are motivated to just guess the answer. The chances are higher if the framesets are graded based on completion, meaning if students reach the end of the frameset then they earn full credit. Baker et al. [11] suggests that students who engage in the gaming system behavior learn only 2/3 as much as students who do not engage in such behavior.

We use a threshold of 8 seconds to determine the number of times students skipped a slide without reading a text statement by iterating through the interaction data and count the number of times the frameset events for the same exercise are recorded within 8 seconds. Instead of just counting raw occurrences of PI Credit Seeking behavior, we also consider the amount of effort the students put into doing PIFrames. Otherwise a student might be recognized as having the most credit seeking behavior if the person simply interacted with PIFrames more than anybody else in the class. To prevent such cases, we consider the ratio of the total number of PIFrame related events and how many of those events are identified as a credit seeking behavior.

5. Conclusion and Future Work

We were able to achieve some success in abstracting session descriptions by simply joining together similar events. For the Spring 2021 Formal Language course, we were able to extract a total of 1,638,155 rows of raw interaction data from the database and abstracted that to 126,412 rows of bundled events, making up 3490 total sessions. This is a reduction below 10%, and also represents breaking the events into something more manageable by human analysts. The average session contains 470 raw and 36 abstracted events. With some effort, human analysts can recognize various patterns of behaviors in the sessions, and from the numbers of sessions for an individual.

While hands-on interaction with the session data can hopefully lead to meaningful hypotheses worthy of testing about what is successful and unsuccessful behavior, the actual testing requires more automation. Our next step was to automatically recognize and compare specific session-level behaviors. We hope to use clustering algorithms to automatically classify session types as described in Section 4. From these clusters we can hope to correlate them to course outcomes, and human analysts can hope to make semantic meaning out of the session clusters. This information can then lead to design changes in the eTextbook presentation or advice to students that can improve learning outcomes.

Adding scrolling data to the event stream might also significantly aid understanding of the session behaviors. Interaction events within exercises are under the control of the exercise creator, and so we are able to capture those events relatively easily. Scrolling is a browser-level event, and so less accessible to eTextbook creators. However, we are are now experimenting with open-source libraries for extracting scrolling events, and will make these part of session analysis in our future work.

References

[1] R. S. J. d. Baker, E. Duval, J. Stamper, D. Wiley, S. B. Shum, Educational data mining meets learning analytics, in: Proceedings of the 2nd International Conference on Learning Analytics and Knowledge, LAK '12, 2012, p. 20.

[2] N. Tax, N. Sidorova, R. Haakma, W. M. van der Aalst, Mining local process models, Journal of Innovation in Digital Ecosystems 3 (2016) 183–196.

[3] A. S. Carter, C. D. Hundhausen, O. Adesope, The normalized programming state model: Predicting student performance in computing courses based on programming behavior, in: Proceedings of the Eleventh Annual International Conference on International Computing Education Research, ICER '15, 2015, p. 141–150.

[4] A. S. Carter, C. D. Hundhausen, Using programming process data to detect differences in students' patterns of programming, in: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, SIGCSE '17, 2017, p. 105–110.

[5] E. Fouh, V. Karavirta, D. A. Breakiron, S. Hamouda, S. Hall, T. L. Naps, C. A. Shaffer, Design and architecture of an interactive eTextbook — The OpenDSA system, Science of Computer Programming 88 (2014) 22–40.

[6] M. Mohammed, C. A. Shaffer, S. H. Rodger, Teaching formal languages with visualizations and auto-graded exercises, in: Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, SIGCSE '21, 2021, p. 569–575.

[7] B. Lockee, D. Moore, J. Burton, Foundations of Programmed Instruction, Handbook of Research on Educational Communications and Technology (2004) 545–569.

[8] F. S. Keller, Good-Bye, Teacher..., Journal of Applied Behavior Analysis 1 (1968) 79–89.

[9] E. N. Fouh Mbindi, Building and evaluating a learning environment for data structures and algorithms courses, Ph.D. thesis, Virginia Tech, http://hdl.handle.net/10919/51951, 2015.

[10] D. A. Breakiron, Evaluating the Integration of Online, Interactive Tutorials into a Data Structures and Algorithms Course, Master's thesis, Virginia Tech, http://hdl.handle.net/10919/23107, 2013.

[11] R. Baker, A. Corbett, K. Koedinger, Detecting student misuse of intelligent tutoring systems, in: 7th International Conference on Intelligent Tutoring Systems, volume 3220, 2004, pp. 531–540.