Detecting Struggling Students from Interactive Ebook Data: A Case Study Using CSAwesome

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

Hisamitsu Maeda
himaeda@umich.edu
University of Michigan
Ann Arbor, MI, USA

Paramveer S. Dhillon
dhillonp@umich.edu
University of Michigan
Ann Arbor, MI, USA

ABSTRACT

Ebooks on the Runestone platform contain instructional material (text, images, videos, and a code visualizer/stepper) and a variety of practice problem types (write code problems with unit tests, multiple-choice questions, mixed-up code problems, etc.). User interaction is timestamped and logged. This paper reports on analyses comparing student interaction data to midterm scores for CSAwesome: a College Board endorsed ebook for the Advanced Placement Computer Science A course. We also analyzed mixed-up code (Parsons) problem data in-depth since these are a newer type of practice. Our analysis found that the percent correct on the midterm was most negatively correlated with being in a larger class and most positively correlated with the percent correct on other multiple-choice questions. It was also positively correlated with several other activities including the percent correct on Parsons problems, active code, and the pretest. Interestingly, it was positively correlated with the number of videos viewed, but negatively correlated with the number of videos completed. Next, our analysis of adaptive mixed-up code (Parsons) problems, where the student can ask for help when stuck, found a positive correlation with the number of steps a user completed before asking for help and a negative correlation with the elapsed time before getting help. Looking closely at two Parsons problems, we found that solving each problem more efficiently, i.e., with fewer extra steps, correlated with higher midterm scores. This work could help instructors identify and support struggling students early in a semester and informs the redesign of the instructor dashboard.

CCS CONCEPTS

• **Applied computing** → **Interactive learning environments**; **Computer-assisted instruction**; **Distance learning**; **E-learning**.

KEYWORDS

struggling students, ebooks, Parsons problems, data mining

ACM Reference Format:
Barbara J. Ericson, Hisamitsu Maeda, and Paramveer S. Dhillon. 2022. Detecting Struggling Students from Interactive Ebook Data: A Case Study Using CSAwesome. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2022), March 3–5, 2022, Providence, RI, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3478431.3499354

SIGCSE 2022, March 3–5, 2022, Providence, RI, USA

ACM ISBN 978-1-4503-9070-5/22/03…$15.00
https://doi.org/10.1145/3478431.3499354

1 INTRODUCTION

Research has shown that interactive ebooks have better learning gains than static ebooks [20]. In addition, most students report that the interactive features help them learn and want to use interactive ebooks in future courses [18]. A 2013 working group predicted that traditional CS textbooks would be replaced by interactive ebooks [17].

Runestone is an open-source interactive ebook platform that has grown from serving one free ebook [18] in 2011 to over 30 free ebooks for computer science and math in 2021. It supports several languages, including Python, Java, C++, and SQL [12]. During the 2020-21 academic year, Runestone had 69,400 registered users and served an average of over two million page views a week.

Runestone interactive ebooks log timestamped user interactions[1]. For the analyses in this paper, we use the **CSAwesome** interactive ebook [6]. CSAwesome has been endorsed by the College Board for the Advanced Placement (AP) Computer Science A (CSA) course. Advanced Placement courses are taken by secondary students for college credit and/or placement. The AP CSA course is equivalent to a first course for majors (CS1) at the college level and covers object-oriented programming in Java. Our CSAwesome data is from a random sample of custom courses because there is too much data to dump it all into a log file. Instructors can create a custom course from any of the free ebooks on Runestone and have their students register for that custom course.

Ebooks provide rich interaction data that could be leveraged to improve instruction [25]. We may be able to harness this data to identify struggling students early in a course as well as their conceptional gaps and misconceptions. Our research questions were 1) what student interactions correlate with the midterm score, 2) what mixed-up code (Parsons) problem interactions correlate with the midterm score, and 3) is there a difference in results by class size? To answer these questions, we performed regression analyses at both the higher level based on the major types of activities and at the lower level with a more in-depth analysis of Parsons problem data. To our knowledge, no prior research has explored using log file data from Parsons problems to detect struggling students.

2 RELATED WORK

Our work draws on learning theories and research on detecting struggling students, interactive ebooks, and Parsons problems.

[1]Researchers can request an anonymized logfile from Brad Miller, the founder of Runestone.

2.1 Learning Theories

The prevalent view of learning is that people construct new knowledge based on what they already know and believe [1]. The ICAP framework hypothesizes that active learning leads to greater learning gains than passive learning [2]. Specifically, it hypothesizes that Interactive > Active > Constructive > Passive. The theory of the Zone of Proximal Development (ZPD) hypothesizes that learning is maximized when learners are challenged to do more than they could alone, but can accomplish with help from a person or system [21]. This implies that systems that adapt the difficulty of problems based on the learner's performance should improve learning outcomes. The CSAwesome ebook was designed to provide both active and adaptive learning [6, 7, 9, 13].

2.2 Detecting Stuggling Students

Costa et al. tried to detect struggling students in introductory programming courses using educational data mining techniques such as Naive Bayes classifiers, decision trees, multilayer neural networks, and support vector machines [3]. They were able to predict struggling students with 79% accuracy after 25% completion of an on-campus course from weekly activities and exams; however, they do not describe what the weekly activities were, and they only used data from one institution [3]. Watson et al. compared compilation behavior on weekly lab programming assignments in a Java course with other standard measures such as prior programming experience, self-efficacy, and lecture attendance [23]. They found that several measures of compilation behavior more strongly correlated with the final mark in the course than most of these, except for self-efficacy. Their research implies that weaker students had more trouble fixing compile-time errors than stronger students. However, they did not include any other data, such as the results from unit tests and their data was only from one course. Another problem is that compile-time data is not always available to instructors. Interactive ebooks provide a much finer-grained look at student progress since they include several types of practice: multiple-choice questions, write code problems, and mixed-up code (Parsons) problems and students typically work in ebooks several times a week.

2.3 Interactive Ebooks

In this paper, we are interested in studying the relationship between students' interaction in the ebook and their pretest and midterm scores. In a related study, Pollari-Malmi et al. [20] found an increase in use, motivation, and learning gains from the use of an interactive ebook versus an equivalent static ebook, which is consistent with the ICAP hypothesis. Ericson et al. [8] found that teachers who used more of the interactive features in an ebook had higher gains in confidence and higher scores on the final posttest, which is also consistent with ICAP.

2.4 Parsons Problems

Several researchers [4, 5, 11, 19, 22, 24, 26] have been exploring mixed-up code (Parsons) problems as both a lower cognitive load form of practice compared to writing code from scratch and as a summative assessment. In Parsons problems, learners must place mixed-up blocks of code in the correct order. They may also have to indent the blocks correctly. Parsons problems can also have distractors, which are code blocks that are not needed in a correct solution. Figure 1 contains distractors shown paired with the correct code block. Helminen et al. [16] visualized students' problem-solving process on Parsons problems using a graphical representation. They detected several different approaches to solving Parsons problems, including top-down, control structures first, and trial and error. Some learners got stuck in circular loops and repeated the same incorrect solution. Ericson et al. [10] found that more learners attempted Parsons problems than nearby multiple-choice questions in an interactive ebook. Du et al. [5] conducted a review of recent studies on Parsons problems.

There are two types of adaptation for Parsons problems on Runestone [7, 9]. In intra-problem adaptation, the problem can be dynamically made easier by removing a distractor or combining two blocks into one. Intra-problem adaptation is triggered by clicking a "Help" button. In inter-problem adaptation, the difficulty of the next problem is automatically modified based on the learner's performance on the last problem. The problem can be made easier by showing distractors paired with the correct code blocks or by removing distractors. It can be made more difficult by using all distractors and randomly mixing the distractor blocks in with the correct code blocks. Learners are nearly twice as likely to correctly solve adaptive Parsons problems than non-adaptive ones and report that solving Parsons problems helps them learn to fix and write code [7, 15]. A randomized controlled study provided evidence that solving adaptive Parsons problems takes significantly less time than writing the equivalent code and with similar learning gains from pretest to posttest [9].

3 DATASET

The log file used in our analysis comes from the CSAwesome interactive ebook on the open-source Runestone platform. This book was revised by Beryl Hoffman of Elms College and the Mobile CSP project in 2019 for the AP CS A course [6]. The log includes page views, video interaction, use of a program visualizer/stepper (CodeLens), and the results from interactive practice problems: multiple-choice, write code (activecode), and mixed-up code (Parsons) problems.

An activecode is a traditional programming exercise where students write/edit/run code in the ebook. Many activecode exercises have unit tests that students can use to verify that their code is correct. Students can also use a slider to view previous versions of the code. The "Show CodeLens" button on the activecode will display a program visualizer (CodeLens), allowing students to step through their code line by line and visualize the variables. It is a version of Guo's Python Tutor [14]. A Parsons problem provides mixed-up code blocks that the learner must place in the correct order [19] as shown in Figure 1. The Parsons problems in this ebook were adaptive. While some adaptive systems use selection adaptation in which the next problem is selected from a set of possible problems based on the learner's performance, this system modifies the difficulty of the current or next problem in the ebook based on the learner's performance.

We analyzed five categories of log file entries: page views, video (play, pause, and completion), activecode interaction (run, edit,

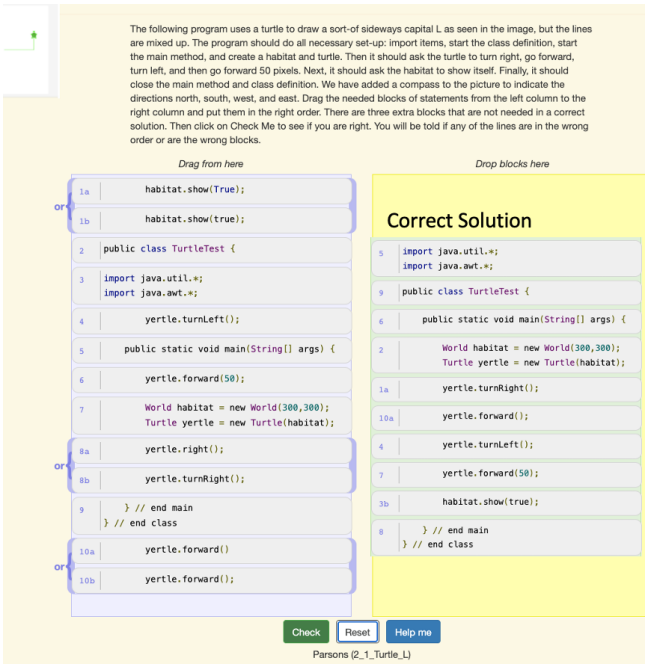

Figure 1: Example Parsons Problem with Paired Distractors

slide, and unit test), Parsons problem block moves and answers, and answers to multiple-choice questions. Overall, the log file contained data from 1,893 students in 57 custom classes. Most of these were high school classes, but some of them were college classes. This paper analyzed a subset of the log data from 505 students who took both the pretest and midterm in their course. The pretest and the midterm are both in the ebook and each has 20 multiple-choice questions. The summary statistics of the data are shown in Table 1. Over 37% of the log file activities were interaction with activecode. The AP CSA exam includes 40 multiple-choice questions and four free-response questions where the student must write Java code to solve a problem. The ebook is broken into ten content-based units and five practice units. At the end of each content-based unit, there are at least ten multiple-choice questions, mixed-up code (Parsons) problems, and write code problems with unit tests [6]. There are also a variety of practice problems throughout the unit subchapters.

We were interested in the effect of class size on students' performance. For this analysis, we divided the students into two groups: classes with less than 30 students, which is the typical maximum class size in high school, and classes with more than 30 students—224 students were in large classes, and 281 were in small classes.

4 METHODS AND RESULTS

This section analyzes the correlations between student interactions and the percent correct on the midterm. Figure 2 shows that the percentage correct on the midterm has a positive correlation with the percentage correct for each activity. The percent correct for each activity was calculated by the number of correct answers divided by the number of attempts for each type of problem. Multiple-choice problems have the highest correlation with the midterm score. This

Event Type	Count	Percentage
activecode	1,020,735	37.66%
page view	651,136	24.02%
Parsons move and answer	524,206	19.34%
multiple choice answer	261,321	9.64%
video	83,892	3.09%
other	169,363	6.25%
Total	2,710,653	100%

Table 1: Various types of events in our dataset.

could be because the midterm is a set of 20 multiple choice questions. There were also positive correlations with the percent correct on Parsons problems, active code, and the pretest. On the other hand, there was a negative correlation between the midterm score and the number of other activities (i.e., the percent of completed videos).

4.1 Regression analysis of student activities

Next, we employ a linear regression model to this data with the percentage of correct answers on the midterm exam as the dependent variable. The counts for each type of student activity were used as independent variables. Since the activity variables have a skewed distribution, we log-transformed them as $x \rightarrow log(x+1)$ and standardized them. Also, we add the class size as a dummy variable in our model to control for the variation in test scores based on the class size. The regression results are shown in Table 2.

The percent correct on the midterm was negatively correlated with being in a larger class, perhaps because there is less one-on-one interaction with the instructor and, as a result, lower learning outcomes. Interestingly, midterm results were also negatively correlated with the number of interactions with the CodeLens and the number of videos completed. This could potentially be because both of these activities were more likely to be used by struggling students who had lower midterm scores.

The midterm results were positively correlated with the percent correct on the pretest, percentage correct on other multiple-choice questions, the number of page views, and the number of videos played. Intriguingly, the midterm score is positively correlated with the number of videos played but negatively correlated with the number of videos completed. This suggests that stronger students might watch a video till they find what they need and then quit.

4.2 Analyzing Parsons Problem Data

In this section, we conduct an in-depth analysis of student interaction data on Parsons problems. As described earlier, these Parsons problems used both intra-problem and inter-problem adaptation. In intra-problem adaptation, if a student submits at least three incorrect solutions, they are notified that they can use a "Help" button to make the problem easier. Each time the student clicks the "Help" button, the ebook will remove a distractor block from the solution or combine two blocks into one, hence providing an implicit hint. This help may be why the percentage of correct Parsons problems was not statistically significant compared to the midterm score.

We pre-processed the log data to gather detailed information on the Parsons interactions. As can be seen from Figure 4, students

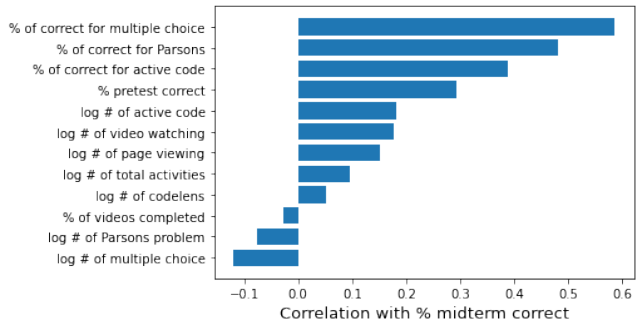

Figure 2: Correlation between percent correct on the midterm and the different activities.

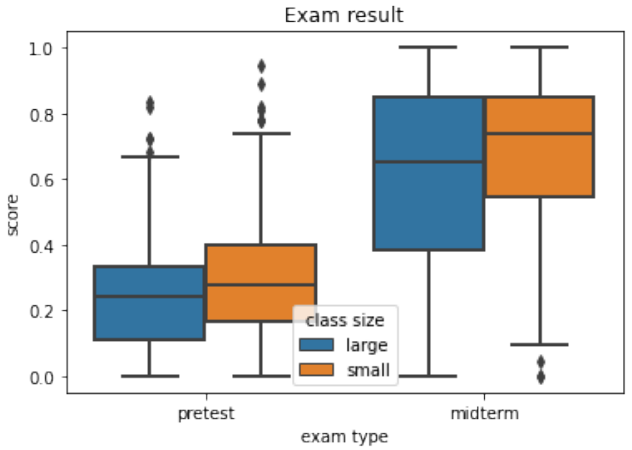

Figure 3: Results for pretest and midterm scores by class size.

Variable	Coef. (p-value)
Large Class (or not)	-0.362*** (0.00)
Percentage correct for pretest	0.1045*** (0.009)
Percentage correct for multiple choice	0.5366*** (0.000)
Number of activecode interactions	0.0901* (0.09)
Number of CodeLens interactions	-0.1403*** (0.002)
Number of page views	0.0931** (0.03)
Number of videos played	0.158*** (0.005)
Number of videos completed	-0.08** (0.03)
N	417
R^2	0.416

$^{***}p < 0.01$, $^{**}p < 0.05$, $^{*}p < 0.1$

Table 2: Regression results: The outcome variable is the midterm score and the independent variables are the student activities.

have to move from a state where all the blocks are jumbled to the state in which all the blocks are placed correctly. We define a step as a block move by either the student or system. We count the number of steps taken by the students and the number of failures incurred until a student finds the correct ordering. We also count the number of times a student got help (clicked the "Help" button) and the number of steps and time until a student asked for help. To tease apart the effect of "getting help," we add an interaction term with the "help flag" in our regression.

The regression result is shown in Table 3. As can be seen from the result, being in a large class is negatively related to the midterm score, as we found in our previous analysis. We also found that the number of steps before a student got help from the software was positively correlated with the midterm test results. Perhaps stronger student learners could figure out more of the problem before asking for help. On the other hand, the time to get support was negatively associated with the test score. This implies that students who took too long to get support scored poorly on the midterm. This has implications for providing help. Currently the help is student initiated, but this result suggests testing other options. In addition, students who received more support, i.e., received more help, did not score as well on the midterm. It is perhaps not surprising that weaker students take longer or need more help, but these indicators could be used to identify struggling students.

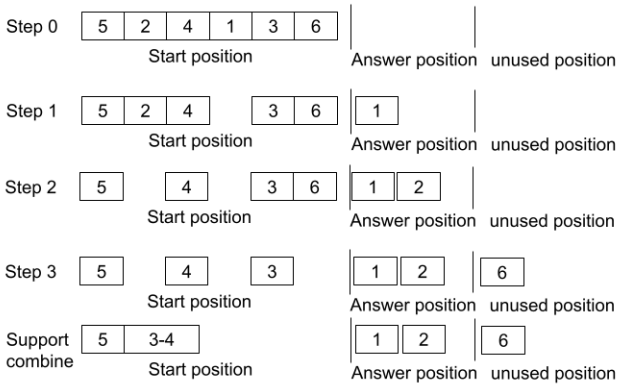

Figure 4: The process of solving a Parsons problem

Variable	Coef. (p-value)
Large Class (or not)	-0.0862*** (0.000)
Number of correct Parsons problems	0.0797* (0.09)
Number of incorrect submissions	-0.0136*** (0.000)
Number of times help is used	-0.0279** (0.03)
Number of steps before getting help	0.413* (0.07)
Elapsed time before getting help	-0.3472* (0.08)
N	402
R^2	0.141

$^{***}p < 0.01$, $^{**}p < 0.05$, $^{*}p < 0.1$

Table 3: Regression analysis of Parsons problem interaction data. The dependent variable is the midterm score and the independent variables are student activities on Parsons problems.

4.3 Deeper analysis of two specific Parsons Problems

Since the percent correct on Parsons problems positively correlates with the percent correct on the midterm, we further investigated two specific Parsons problems. In one problem in Unit two learners drew a sideways L with a turtle as shown in Figure 1. We analyzed the number of block moves and the time elapsed before the learner found the correct answer for each problem. We counted the number of steps it took from the initial state to the correct order, as discussed earlier. We define "extra steps" as the number of code-block moves beyond the required number of moves to a correct answer. For example, if a solution can be reached in just ten steps, and a student took twelve steps, this would be two extra steps. We also measured the amount of time it took students to reach the correct solution. Figure 5 shows a correlation between the number of extra steps to solve this Parsons problems versus the time to a correct solution for students in large and small classes.

Figure 6 compares the extra steps taken to solve Parsons problems by groups: 1) small class and less than median midterm scores, 2) small class and greater than median midterm scores, 3) large class and less than media midterm scores, and 4) large class and greater than median midterm scores. As can be seen, there is a significant relationship between students with higher than median midterm scores and the number of extra steps taken (t-statistic 5.096, $p < 0.001$). On the other hand, there is no relationship between class size and the number of extra steps taken (t-statistic 1.16, $p > 0.1$).

Figure 7 shows the result comparing the time by class size and midterm test score. Unlike the number of "extra steps," there is no significant association between students with higher than median midterm scores and the time taken by the learners (t-statistic 3.7, $p < 0.001$). Also, there is no relationship between class size and time (t-statistic 1.39, $p > 0.1$).

From this analysis, it appears that students who took fewer extra steps while solving this Parsons problem have better midterm scores. A similar trend is also observed in another problem from Unit 4, as shown in Figure 8.

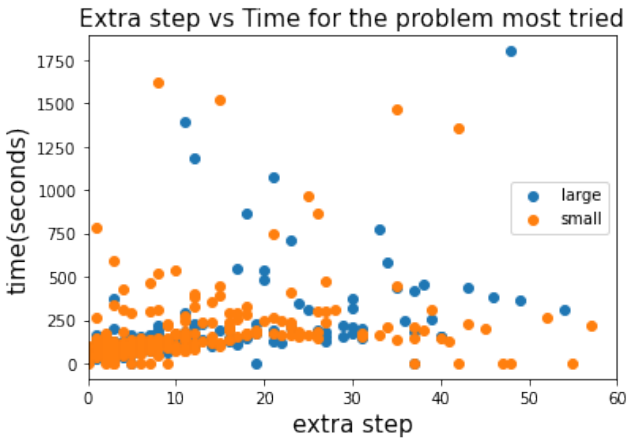

Figure 5: Relationship between extra steps and time

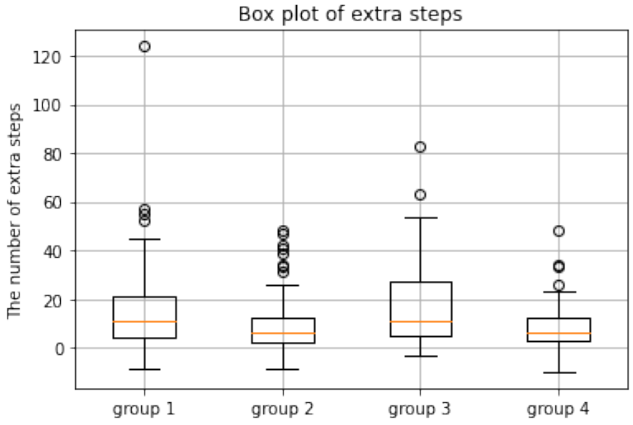

Figure 6: Comparison of number of extra steps while solving Parsons problems by groups: 1) small class and midterm score less than median, 2) small class and midterm score greater than median, 3) large class and midterm score less than median, and 4) large class and midterm score greater than median

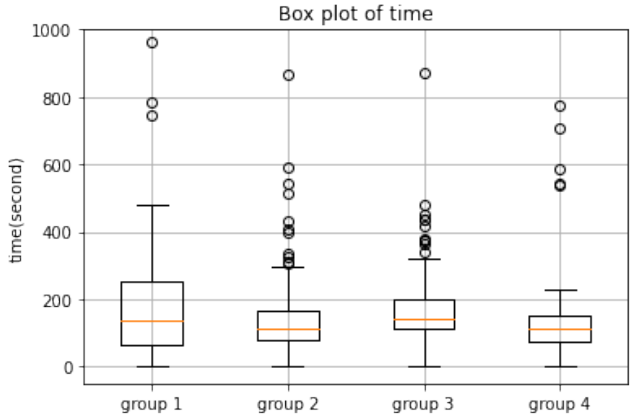

Figure 7: Comparison of Parsons problem completion time by groups: 1) small class and midterm score less than median, 2) small class and midterm score greater than median, 3) large class and midterm score less than median, and 4) large class and midterm score greater than median

4.4 Comparing the Percent Correct on Parsons and Write Code to the Midterm Scores by Groups

We compared the four groups 1) small class and midterm score less than the median, 2) small class and midterm score greater than the median, 3) large class and midterm score less than the median, and 4) large class and midterm score great then the median with respect to the percent correct for both Parsons problems, as shown in Figure 9, and write code (activecode) problems, as shown in Figure 10. There was a significant difference between groups that scored below and above the median for the percent correct on activecode problems (t

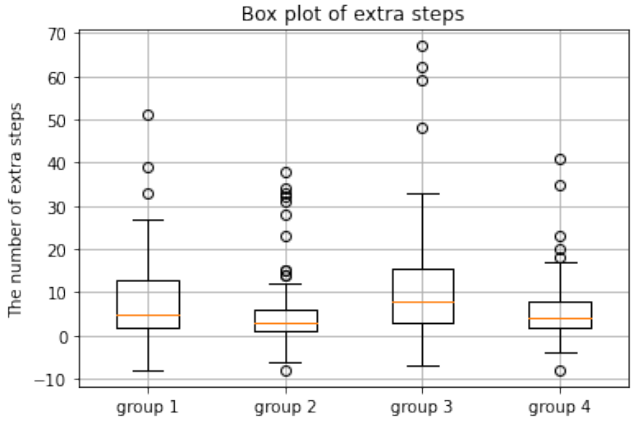

Figure 8: Comparison of the number of extra steps with midterm score and class size for a problem in Unit 4 by group.

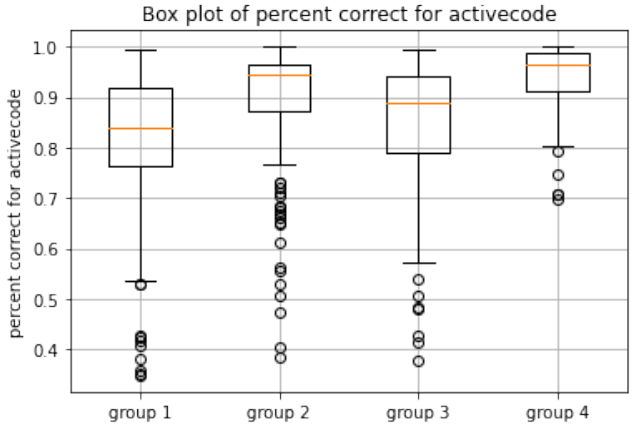

Figure 10: Comparison of percent correct on activecode problems by group.

statistic -6.5 p=2.3e-10) but not for Parsons problems (t statistic -2.19 p= 0.03). This could be because the adaptation to Parsons problems makes it easier for weaker students to solve the problem correctly.

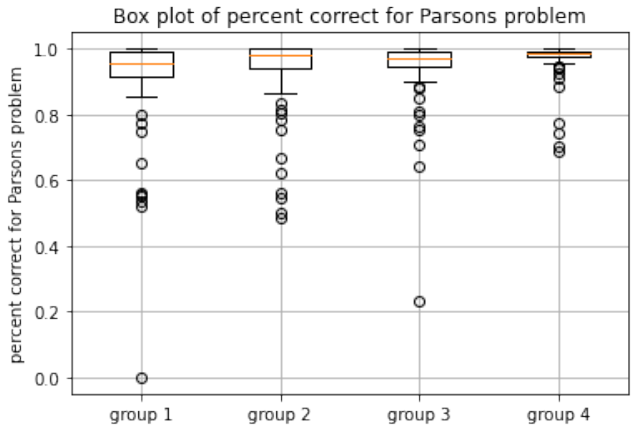

Figure 9: Comparison of percent correct on Parsons problems by group.

5 DISCUSSION

We performed several quantitative analyses on the clickstream data from the CSAwesome ebook to study the relationship between students' activities in the ebook and their midterm scores. In our regression analysis, the variable with the largest positive effect size was the percentage correct on other multiple-choice questions. On the other hand, the variable with the strongest negative correlation was the class size. It is not entirely surprising that the percent correct on other multiple-choice questions strongly correlates with the percent correct on the midterm or that students tend to do worse in larger classes than smaller ones. Several other positive and negative correlates are more surprising, such as the negative correlation with the number of videos completed, but the positive correlation with the number of videos played.

We also analyzed the learner interaction patterns on the Parsons problems since these are a newer type of practice. Specifically, we examined the correlation between the midterm score and the number of steps taken, time taken, and the frequency of "help." The results show a positive association between the number of correctly completed Parsons problems and the learners' midterm scores. However, there was a negative association between the midterm scores and the time taken by the student to get help on a Parsons problem. A closer investigation of two Parsons problems showed a negative correlation between the number of extra steps, time to solve that Parsons problem, and the midterm score.

6 LIMITATIONS

We only analyzed the data from one interactive ebook using the Java programming language. These results may not generalize to other ebooks or languages. We also only analyzed two Parsons problems in-depth. More work should be done to determine if the number of extra steps is consistently correlated with performance on the midterm. Also, the student data was from a random selection of custom courses on the Runestone platform. We did not have any additional information about these courses, such as which items were assigned, final grades, or student demographics.

7 CONCLUSION

In this paper, we mined the student interaction logs from the CSAwesome ebook to quantify the ability of such data to predict the students' educational outcomes (midterm scores). Our analyses uncovered key indicators that could potentially be used for the early identification of struggling students in a course. Such predictors could be used to design interventions to improve the learning outcomes of students. These correlates of students' midterm performance can also be used to enhance an instructor dashboard.

REFERENCES

[1] John D Bransford, Ann L Brown, and Rodney R Cocking. 2000. *How people learn*. Vol. 11. Washington, DC: National academy press.

[2] Michelene TH Chi and Ruth Wylie. 2014. The ICAP framework: Linking cognitive engagement to active learning outcomes. *Educational psychologist* 49, 4 (2014), 219–243.

[3] Evandro B Costa, Baldoino Fonseca, Marcelo Almeida Santana, Fabrísia Ferreira de Araújo, and Joilson Rego. 2017. Evaluating the effectiveness of educational data mining techniques for early prediction of students' academic failure in introductory programming courses. *Computers in Human Behavior* 73 (2017), 247–256.

[4] Paul Denny, Andrew Luxton-Reilly, and Beth Simon. 2008. Evaluating a new exam question: Parsons problems. In *Proceedings of the fourth international workshop on computing education research*. 113–124.

[5] Yuemeng Du, Andrew Luxton-Reilly, and Paul Denny. 2020. A Review of Research on Parsons Problems. In *Proceedings of the Twenty-Second Australasian Computing Education Conference* (Melbourne, VIC, Australia) *(ACE'20)*. Association for Computing Machinery, New York, NY, USA, 195–202. https://doi.org/10.1145/3373165.3373187

[6] Barbara Ericson, Beryl Hoffman, and Jennifer Rosato. 2020. CSAwesome: AP CSA Curriculum and Professional Development (Practical Report). In *Proceedings of the 15th Workshop on Primary and Secondary Computing Education* (Virtual Event, Germany) *(WiPSCE '20)*. Association for Computing Machinery, New York, NY, USA, Article 5, 6 pages. https://doi.org/10.1145/3421590.3421593

[7] Barbara Ericson, Austin McCall, and Kathryn Cunningham. 2019. Investigating the Affect and Effect of Adaptive Parsons Problems. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*. 1–10.

[8] Barbara Ericson, Steven Moore, Briana Morrison, and Mark Guzdial. 2015. Usability and usage of interactive features in an online ebook for CS teachers. In *Proceedings of the Workshop in Primary and Secondary Computing Education*. 111–120.

[9] Barbara J Ericson, James D Foley, and Jochen Rick. 2018. Evaluating the efficiency and effectiveness of adaptive parsons problems. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*. 60–68.

[10] Barbara J Ericson, Mark J Guzdial, and Briana B Morrison. 2015. Analysis of interactive features designed to enhance learning in an ebook. In *Proceedings of the eleventh annual International Conference on International Computing Education Research*. 169–178.

[11] Barbara J Ericson, Lauren E Margulieux, and Jochen Rick. 2017. Solving parsons problems versus fixing and writing code. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*. 20–29.

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

[13] Barbara J Ericson, Kantwon Rogers, Miranda Parker, Briana Morrison, and Mark Guzdial. 2016. Identifying design principles for CS teacher Ebooks through design-based research. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*. 191–200.

[14] 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.

[15] Carl C. Haynes and Barbara J. Ericson. 2021. Problem-Solving Efficiency and Cognitive Load for Adaptive Parsons Problems vs. Writing the Equivalent Code. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 60, 15 pages. https://doi.org/10.1145/3411764.3445292

[16] Juha Helminen, Petri Ihantola, Ville Karavirta, and Lauri Malmi. 2012. How Do Students Solve Parsons Programming Problems? An Analysis of Interaction Traces. In *Proceedings of the Ninth Annual International Conference on International Computing Education Research* (Auckland, New Zealand) *(ICER '12)*. Association for Computing Machinery, New York, NY, USA, 119–126. https://doi.org/10.1145/2361276.2361300

[17] Ari Korhonen, Thomas Naps, Charles Boisvert, Pilu Crescenzi, Ville Karavirta, Linda Mannila, Bradley Miller, Briana Morrison, Ross Rocky Rodger, Susan H, and A Shaffer, Clifford. 2013. Requirements and design strategies for open source interactive computer science eBooks. In *Proceedings of the ITiCSE working group reports conference on Innovation and technology in computer science education-working group reports*. 53–72.

[18] Bradley N Miller and David L Ranum. 2012. Beyond PDF and ePub: toward an interactive textbook. In *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*. 150–155.

[19] Dale Parsons and Patricia Haden. 2006. Parson's programming puzzles: a fun and effective learning tool for first programming courses. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*. 157–163.

[20] Kerttu Pollari-Malmi, Julio Guerra, Peter Brusilovsky, Lauri Malmi, and Teemu Sirkiä. 2017. On the value of using an interactive electronic textbook in an introductory programming course. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*. 168–172.

[21] Lev Semenovich Vygotsky. 1980. *Mind in society: The development of higher psychological processes*. Harvard university press.

[22] Wengran Wang, Rui Zhi, Alexandra Milliken, Nicholas Lytle, and Thomas W Price. 2020. Crescendo: Engaging students to self-paced programming practices. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 859–865.

[23] Christopher Watson, Frederick WB Li, and Jamie L Godwin. 2014. No tests required: comparing traditional and dynamic predictors of programming success. In *Proceedings of the 45th ACM technical symposium on Computer science education*. 469–474.

[24] Nathaniel Weinman, Armando Fox, and Marti A. Hearst. 2021. Improving Instruction of Programming Patterns with Faded Parsons Problems. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 53, 4 pages. https://doi.org/10.1145/3411764.3445228

[25] Hongxin Yan, Fuhua Lin, et al. 2020. Including Learning Analytics in the Loop of Self-Paced Online Course Learning Design. *International Journal of Artificial Intelligence in Education* (2020), 1–18.

[26] Rui Zhi, Min Chi, Tiffany Barnes, and Thomas W Price. 2019. Evaluating the Effectiveness of Parsons Problems for Block-based Programming. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. 51–59.