Open IDE Action Log Dataset from a CS1 MOOC

Juho Leinonen
University of Helsinki
juho.leinonen@helsinki.fi

ABSTRACT
Studying students’ time-related behavior is a common research topic within educational data mining. One way to study time-related behavior is by collecting log data from the use of pedagogical tools or systems. In computer science in particular, such data has often been collected from students’ use of integrated development environments (IDEs). In this work, we present and openly release a dataset containing IDE logs from an introductory programming MOOC. The dataset contains information on when actions in the IDE were performed in relation to deadlines over the different parts of the course. One exceptional aspect of the dataset is that part of the logs have been gathered at the keystroke level, allowing for fine-grained insight into the learning process. In addition to the IDE logs themselves, the dataset has information on whether students included in the data passed the course. This can facilitate further research that analyzes how time-related behavior correlates with performance in introductory programming courses.

Keywords
open data, IDE logs, log data, integrated development environment, CS1, keystroke data, programming process data, learning analytics, educational data mining

1. INTRODUCTION
One recurring research theme in educational data mining is studying student time-related behavior such as earliness of work [4, 6, 12, 15, 17], procrastination [9, 10, 18], and time-on-task [13, 14, 21]. Studies in this area typically utilize logs gathered from pedagogical tools or learning management systems. In computer science educational data mining (CSEDM) in particular, studying logs from integrated development environments (IDEs) has been common [8].

In this work, we present and make publicly available a dataset [16] containing IDE logs from an introductory programming massive open online course (MOOC). The logs contain information on actions students perform in the IDE while working on course exercises. Notably, the data does not contain source code etc. and can be thought of as “metadata” related to the students’ learning processes. Such data has potential, for example, in developing time management based interventions, for example, by following the IDE-based learning analytics process model by Hundhausen et al. [7].

The data presented here is partly at the keystroke level, which allows for a fine-grained look into students’ learning [11]. This data contributes to the growing body of open IDE log datasets [1, 3, 5], which can both support future research utilizing IDE logs and help replicate earlier findings; lack of replication has been noted as a major challenge in computing education research [8].

2. CONTEXT
The context of the data is a massive open online course (MOOC) on introductory programming (CS1). The programming language used in the course is Java. The course is split into seven parts, where each individual part introduces a few new programming concepts to the students. The concepts covered in the course are typical CS1 topics, beginning with variables, conditionals, loops and lists, which are followed by methods, parameters, classes and objects.

The course uses online materials. In the online material, concepts are introduced with examples that focus on how the concepts can be used in programs. The introduction of the concept (“theory part”) is followed by a couple of exercises where students are expected to practice the concept. Each part has a few larger exercises at the end of the material that typically combine earlier concepts and the new concepts introduced in the respective part of the course. The course uses the many small exercises approach [2, 22], where students work on anywhere from ten to thirty exercises in each part of the course.

Exercises in the course are completed using an integrated development environment (IDE) with the custom Test My Code -plugin [23]. While exercise descriptions are found in the online material, students can download exercise templates directly into the IDE. The templates typically contain boilerplate code, and in some cases some starter code. The exercises are accompanied by instructor-created unit test suites that students can test their program against directly in the IDE. When students believe their code to be
Each part of the course has a deadline after which students can no longer submit their program for evaluation nor get the points for the exercises of that part. They can continue working on the exercise after the deadline if they so desire; however, events that occur after the deadline are not included in the data. The first four parts of the course had the same deadline to allow students to join the course late. For the fifth, sixth and seventh part of the course, the deadlines were one, two and three weeks later (compared to the deadline for the first four parts), respectively. The course was released 42 days before the deadline of the first four parts, so there are students who worked close to when the course was released and others who worked closer to the deadlines. Since the course is a MOOC, some of the students who started later might have only heard about the course after it was released, so starting closer to the deadlines might not indicate procrastination.

In addition to the exercises, there were three online exams. The online exams were completed in the IDE similar to course exercises with the exception that there was a time limit of two hours to complete all exercises of the exam from the time when the student started the exam. The three exams were situated at different points in the course. The first exam was at the beginning of the third part and covered topics from parts 1 and 2. The second exam was at the beginning of part 5 and covered the previous four parts. The third exam was at the end of part 7 and covered all course topics.

Passing the course was determined based on both points received from the exercises (55% of the total points) and points received from three online exams (45% of the total points). Students could receive 8% of the points by completing every exercise in each of the seven parts of the course (except for the first part, where they could get only 7% of the points). The three exams were worth 10%, 15% and 20% of the points. Students had to receive a total of 51% or more of the total course points in order to pass the course. In addition, students were required to receive at least half of the points from the last exam in order to pass the course, regardless of any other points received. Some students might not have many IDE logs even if they passed the course as it is possible they partly worked offline in which case the logs were never sent to the server, or they might have disabled logging intermittently.

The IDE used in the course collects data from the students’ programming process. Only data from students who consented for their data to be used for research is included in the dataset. Additionally, students could turn logging on and off at will within the IDE.

<table>
<thead>
<tr>
<th>Event type</th>
<th>Event count</th>
</tr>
</thead>
<tbody>
<tr>
<td>text_insert</td>
<td>19,652,100</td>
</tr>
<tr>
<td>text_remove</td>
<td>5,582,299</td>
</tr>
<tr>
<td>text_paste</td>
<td>166,601</td>
</tr>
<tr>
<td>focus_gained</td>
<td>1,427,004</td>
</tr>
<tr>
<td>focus_lost</td>
<td>1,383,954</td>
</tr>
<tr>
<td>run</td>
<td>323,749</td>
</tr>
<tr>
<td>submit</td>
<td>52,602</td>
</tr>
<tr>
<td>total</td>
<td>28,588,309</td>
</tr>
</tbody>
</table>
programming snapshot format as the ProgSnap format requires sharing the source code (“CodeStateID” is a required column for ProgSnap2), which we are unable to share at this time due to privacy concerns.

There are seven types of events in the data which are as follows.

- **text_insert** – text was added into the IDE (at the keystroke level)
- **text_remove** – text was removed from the IDE
- **text_paste** – text was pasted into the IDE
- **focus_gained** – the IDE gained focus (student clicked into the IDE)
- **focus_lost** – the IDE lost focus (student clicked away from the IDE)
- **run** – student executed their program
- **submit** – student submitted their program

The data has a total of 28,588,309 rows (i.e., logged IDE actions) from a total of 473 students. The mean number of events per student is 60,440 (with a standard deviation of 61,228 and a median of 50,810). The number of each type of event in the data is outlined in Table 1. A few example data rows are shown in Figure 1 and the distribution of logged actions per student is visualized in Figure 2.

The passed.csv has one row for each student. The only columns are a student identifier and “passed” where the value is either **True** (student passed the course) or **False** (student did not pass the course).

Both IDE_logs.csv and passed.csv can be found here: https://doi.org/10.5281/zenodo.6903968

4. POTENTIAL RESEARCH QUESTIONS

There are many research questions that could be studied utilizing this type of data. As the data does not contain the source code or source code related information such as compilation status or correctness, the main type of research that the data is useful for is related to students’ behavior in IDEs. Here, we outline a few potential research questions and areas where such data could be used.

One stream of research that can utilize this type of log data is studies related to student time-on-task. Recent works have, for example, examined how the granularity of data affects time-on-task estimates [13] and how time-on-task estimates derived from log data correlate with students’ performance [14]. Potential research questions that remain are, for example, how individual students’ times-on-task differ between exercises – is it always the same students who spend a lot of time on exercises, or are the exercise-specific differences between students in how much time they spend on the exercise?

Another common area of research in computing education where the data could be used is studying student time management such as how early (or late) students work on exercises, and procrastination. Prior work has found correlations between starting early and performing better in the course [6, 12]. Future work in this area could study, for example, whether time management related behavior varies across the course and how well can log-derived time management metrics predict student performance in the course. Additionally, one could try clustering the data to try identify whether there are clear student subpopulations with different time-related behavioral patterns, for example, try replicate prior work studying “tinkerers”, “movers”, and “stoppers” [19].

Lastly, a relatively under-explored aspect of student work is examining at what time of the day students work on exercises, and how this correlates with performance in the course. A recent study by Zavgorodniaia et al. [24] clustered students based on when they were working in an IDE and found clusters that roughly correspond to different “chronotypes”, i.e. people’s propensity to sleep at certain times of day and correspondingly, be alert at other times.

There are two main limitations related to the data. First, there is the possibility of self-selection bias as only data from students who both chose to allow action logging within the IDE and who consented to their data being used for research is included. It is possible that students who chose to do so differ from other students in the course. Second, the data does not include the source code or e.g. an AST representation of it. As part of our future work, we are exploring ways of sharing source code and code states while preserving the privacy of the students who have supplied the data.

Altogether, there are multiple research directions where IDE action logs could be utilized. Releasing this dataset openly hopefully facilitates future research in this area.
5. REFERENCES


