Coding Trajectory Map: Student Programming Situations Made Visually Locatable

Yuta Taniguchi, Tsubasa Minematsu, Fumiya Okubo and Atsushi Shimada
Kyushu University, Japan
taniguchi.yuta.941@m.kyushu-u.ac.jp

ABSTRACT: Understanding student activities in programming exercise is vital for teachers to support students. Because teachers cannot always pay attention to every student's source code, they have difficulties in identifying students who are having trouble with errors and understanding how students write source code. The representative approach of using dashboard systems, however, has limitation on the depth of understanding of students' situations due to the lack of intuitive presentation of the processes in which source code contents evolve. This study aims to provide beneficial information to teachers about the temporal changes of student coding situations. Based on a dataset of source code snapshots taken frequently, we propose a feedback tool for teachers, called Coding Trajectory Map. We conducted an experiment for human evaluation of the tool, whose result shows that teachers can better understand the learning situations of students with the tool.

Keywords: Learning process, Learning situation, Programming, Visualization, Teacher support

1 INTRODUCTION

In novice programming education, many students struggle to write and debug programs (Robins et al., 2003). The difficulties in programming often lower learner motivation which is an important factor of success. Thus, supporting students in difficult situations is crucial. However, identifying such students is hard because help-seeking is known as a difficult metacognitive skill and, in addition, teachers cannot always pay attention to every student's learning situation. Learning analytics dashboards arises to overcome this challenge also in this field (Diana et al., 2017; Fu et al., 2017), which present to teachers an overview of their students' situations through statistics, performance indicators, and visualization, and so on. However, the depth of understanding via prior dashboards is limited because the temporal changes of source code content are less likely to be considered. For example, students' trial and error and coding strategies could be observed only from how source code changes over time, and it is hard to realize from statistical numbers or a single snapshot of source code. The research question in this study is "How can we help teachers obtain insights into student situations when we have a class-wide dataset of student source code snapshots?" Based on the research question, we aim to make student learning situations visually and intuitively locatable making use of a mass of snapshots that were taken frequently. Here, we mean by "locatable" that a learning situation is linked to a specific position on a visual map, where the distance between two positions intuitively reflects the difference in corresponding situations at source code level. In this poster paper, we present a method to create such a visual map, called Coding Trajectory Map, and report the result of a preliminary evaluation experiment on the usefulness of maps for teachers.

Creative Commons License, Attribution - NonCommercial-NoDerivs 3.0 Unported (CC BY-NC-ND 3.0)

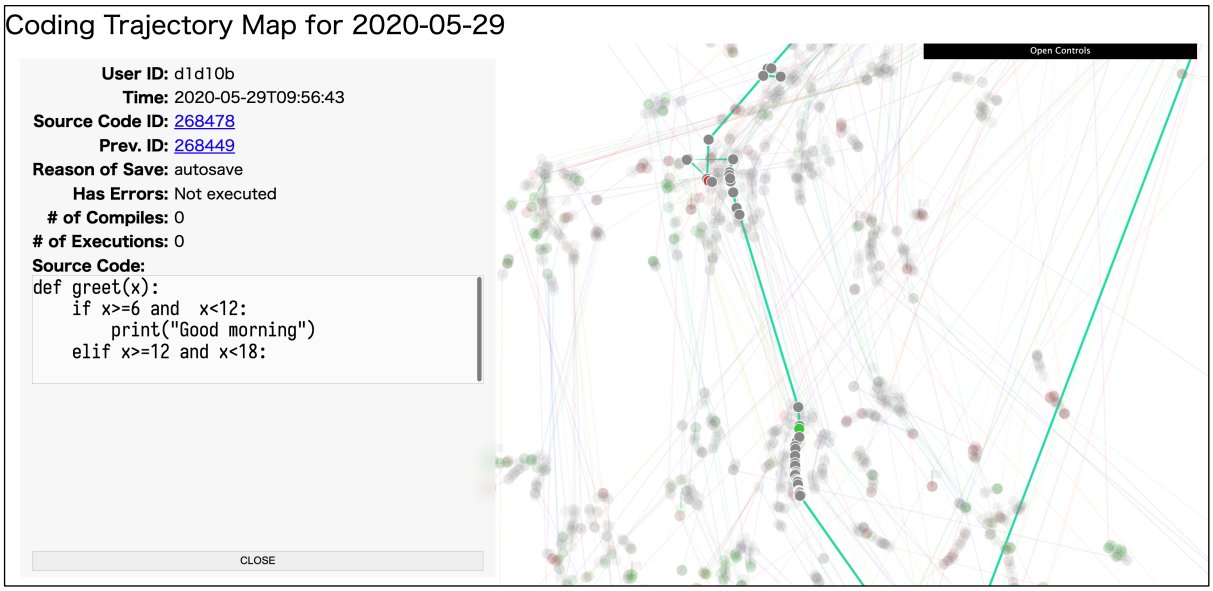

Figure 1: Coding Trajectory Map showing the distribution of source code on the right-hand side and the information of a selected source code snapshot on left-hand side.

2 METHODS

Dataset: We collect a set of source code snapshots in an introductory Python programming course offered in 2020 for first-year students at our university. We used our own online programming environment, called WEVL, to collect snapshots. With WEVL, a snapshot is taken three seconds after a student stops typing or when a student runs a program. In our dataset, about 84% of snapshots were taken by stopping typing.

Visualization: The proposed Coding Trajectory Map shows on a 2D plane a set of source code snapshots. The positions of snapshots are determined with t-SNE algorithm together with edit distance. First, edit distance is measured for every pair of snapshots in a dataset and a distance matrix is formed. An edit distance between a pair of source code is the minimum total number of adding, deleting, or substituting character-wise operations required to transform one into another. Intuitively, this is roughly the same as how much keyboard typing is necessary for the transformation. Second, the t-SNE algorithm is applied to the distance matrix, and we obtain 2-dimensional coordinates for each snapshot. The algorithm computes those coordinates trying to maintain the original distances, i.e., edit distance, as much as possible in the output space. Therefore, it is expected on the map that similar source code snapshots are placed nearby and the distance tell us approximately how much typing effort is required to reach a certain source code starting from one. Furthermore, because our snapshots were taken frequently, it is also expected that a series of snapshots from a student would be like a string of beads when visualized. Figure 1 shows an example of Coding Trajectory Map. On the right, each snapshot is shown as points, snapshots from the same student are connected by a line. On the left, the information of a snapshot that a user selected on the map is provided.

Evaluation: We asked 7 university teachers to evaluate the Coding Trajectory Map tool in the context of reflecting past classes in the same course as described above. Those teachers had experiences of teaching programming before. We provided instructions on how to use the tool in both text and video formats. Teachers were asked to learn the usage of the tools and then to use the tool themselves. We

Creative Commons License, Attribution - NonCommercial-NoDerivs 3.0 Unported (CC BY-NC-ND 3.0)

Table 1: The result of questionnaire for evaluating the proposed tool. (N=7)

ID	Question	Mean	SD
Q1	The tool is useful for understanding the process of each student's exercise in detail by checking the content and status of the source code at any given time, including the presence or absence of runtime errors.	4.43	0.53
Q2	The tool is useful for understanding what kind of code was struggles for students.	4.29	0.76
Q3	The tool is useful for understanding each major and minor programming activity in the class.	4.14	0.69
Q4	The tool can help you identify students to whom you should pay attention.	3.86	1.07
Q5	The tool helps the teacher to reflect on the lesson.	4.00	0.82

also asked them to fill out a questionnaire with 5-point Likert scale to measure the degree of agreement on each question, where 1 stands for strongly disagree and 5 stands for strongly agree.

3 RESULTS AND DISCUSSION

Table 1 shows the result of the evaluation questionnaire. The tool was given the high score of 4.43 regarding the question Q1, which suggests the tool successfully provided the details of the individual coding activities. In terms of source code, we can say that the teachers thought the tool is helpful for finding one that is problematic to students (Q2). Concerning coding activity patterns, it is suggested that common patterns and abnormal activities could be easily identified by teachers (Q3). As for students, we can expect the tool may help us to notice students whom we should pay attention to (Q4) although the score is not very high. Overall, it is shown that Coding Trajectory Map has functions useful to find important code, students, and activities in programming exercise, and teachers agreed that the tool is useful to reflect on classes (Q5). We conclude that we successfully presented the tool which could be beneficial for gaining insights into problematic source code or major and minor coding activities from frequent source code snapshots. While this study focused on teacher support, this kind of feedback would be also useful for students because they will be able to relatively position their own situations on the map. As a future study, we are planning to evaluate the proposed tool in the context of student support.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number JP21K17863.

REFERENCES

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer science education, 13*(2), 137–172.

Diana, N., Eagle, M., Stamper, J., Grover, S., Bienkowski, M., & Basu, S. (2017). An instructor dashboard for real-time analytics in interactive programming assignments. In *Proceedings of the seventh international learning analytics & knowledge* (p. 272–279).

Fu, X., Shimada, A., Ogata, H., Taniguchi, Y., & Suehiro, D. (2017). Real-time learning analytics for C programming language courses. In *Proceedings of the seventh international conference on learning analytics & knowledge* (pp. 280–288).

Creative Commons License, Attribution - NonCommercial-NoDerivs 3.0 Unported (CC BY-NC-ND 3.0)