https://doi.org/10.36315/2025v1end056
Education and New Developments 2025

ANALYSIS FOR CHARACTERISTICS OF LOW-ACHIEVERS AND CONSTRUCTION OF A CLASSIFICATION MODEL USING PROGRAMMING DATA

Takuya Hasegawa[1], **& Yosuke Ito**[2]
[1]Hyogo University of Education (Japan)
[2]Naruto University of Education (Japan)

Abstract

Since 2018, programming has become a compulsory component of information education in Japanese senior high schools. Programming proficiency should be increased by improving instruction for difficult to understand content. This study aims to construct a model to classify students into subgroups at risk of low proficiency using machine learning, focusing on the types and number of token occurrences in student-created programs. Students created programs using a newly developed web-based coding system. When an operational event such as character input or execution occurs, the type of event, time, and current program are sequentially recorded as programming data. The student-created programs were divided into tokens and the occurrences of these tokens was analyzed. A classification model was constructed using machine learning to detect subgroups based on the number of occurrences of each token. It was found that low achievers could be classified with an accuracy of approximately 70% using this model. This has important implications for providing focused instruction for weaker students.

Keywords: *Programming, machine learning, classification model, low-achiever, token.*

1. Introduction

Educational activities in Japanese senior high schools are conducted according to nationally standardized curriculum guidelines. The current curriculum guidelines were announced in 2018, when the subject area of informatics was revised, and a new subject “Informatics I” was established and made compulsory. This subject includes programming, requiring the development of skills related to the ability to select and correctly express an algorithm, consider the efficiency of an algorithm, create a program, check the operation of the created program, and, correct defects (MEXT, 2018).

These skills require extensive instruction, particularly for students weak in programming. Our previous research indicates that the degree of similarity between student-created and exemplar programs is related to proficiency level (Hasegawa 2024a). The purpose of this study is to construct a model for detecting low achievers through machine learning, using the number of occurrences of each token in the student-created programs.

2. Teaching practice

In 2024, three lessons with a binary search program as a teaching unit were taught to 135 second-grade students in four classes at a private senior high school in Kyoto, Japan. Prior to this teaching practice, the students had learned fundamental programming skills such as branching, iteration, and arrays. They had also taken a class on creating a linear search program. JavaScript was used as the programming language. In the first lesson, the teacher explained the binary search algorithm. In the second lesson, an exemplary program was presented with an explanation by the teacher, after which each student attempted to create a binary search program independently. The third lesson reviewed key points to reinforce each student’s understanding of binary search programs.

A written examination was conducted to assess student proficiency after the last lesson. Of the six total questions, one was on understanding the algorithm, four on completing the program, and one on improving the program. Each question was allotted one point. The mean score was 3.6±1.6 (n=132). Given the importance of identifying weaker students and provide them with early assistance, we classified the students into top, middle, and lower groups according to their scores on the written examination. The actual classification results showed that students with a score of three were classified into the middle and lower groups. All students with a score of 3 were classified in the middle group, because more students were classified in the middle group than in the lower group. Finally, 35 students were classified as low achievers. 97 students were classified into the non-lower group, which was the sum of the middle and upper groups.

3. Programming data

The students used a web-based coding record system newly developed by Hasegawa (Hasegawa 2024b) for their study. A block diagram of the system is shown in Figure 1. Students can create programs in JavaScript and execute them by accessing the system through a web browser. The system detects operational events when a student enters characters to create or execute a program. Immediately after the detection, the type of operation event, the current timestamp, and the entire program being created are sequentially recorded as "programming data" into the system's database. A total of 58,817 records were obtained from 126 students for 20 minutes of creating programs in this teaching practice.

The time-series mean changes in the cumulative event count and number of characters in the program obtained as programming data for the lower and non-lower groups are shown in Figures 2(a) and (b), respectively.

Figure 1. Block diagram of a web-based coding record system.

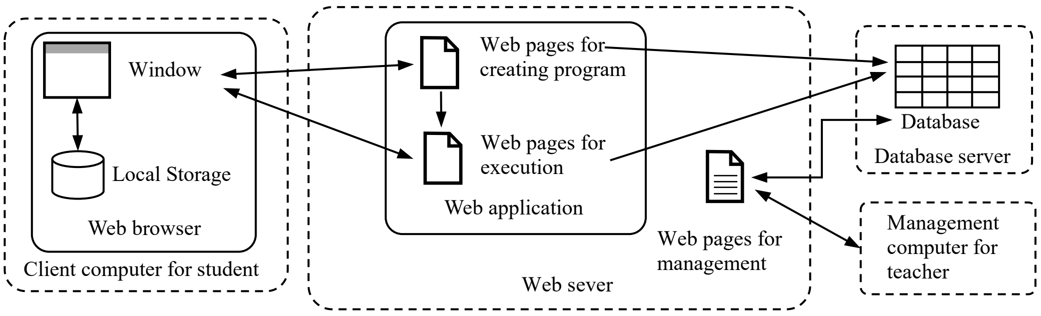
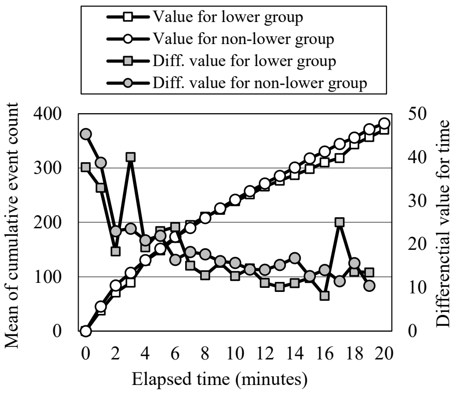
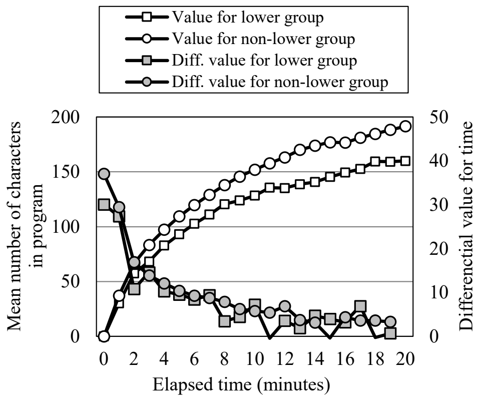


Figure 2. Time-series mean changes in programming data for the lower and non-lower groups.



(a) Cumulative event count



(b) Number of characters in program

As shown by the difference values, the cumulative number of events for both groups increased with time. However, the increase in the mean number of characters in the programs for both groups was suppressed after approximately 2 minutes. One possible reason for the time-series changes shown in Figure 2(b) is that the characters were deleted to modify the program. The mean number of characters in the programs at 20 minutes was 160.0 for the lower group and 191.6 for the non-lower group. The difference between the two means was significant at the 0.05 level of significance. The mean number of program execution events was 2.0 for the lower group and 2.2 for the non-lower group. Only three students in the non-lower group completed the program.

4. Classification model

We propose a model that classifies students into lower or non-lower groups by employing the programming data. First, we extracted the programs created by 126 students at 20 minutes. These programs are divided into tokens. Subsequently, space, string and numeric literal tokens, and tokens that did not appear in the answer program were removed. The number of occurrences of each token is then determined. We obtained the mean number of occurrences of each token in the programs for the lower and non-lower groups, and applied a t-test to verify whether these means differed significantly. There were significant differences in the tokens such as '/', '<=', 'Math', 'floor', 'while' and others. The results suggest that it may be possible to classify students into the lower group based on the number of occurrences of tokens.

A classification model was constructed by applying machine learning, where the explanatory variable was the number of occurrences of each token, and the objective variable was the lower and non-lower groups. Five algorithms were selected to investigate the suitability of machine learning for this classification: (M1) logistic regression, (M2) decision tree, (M3) support vector machine, (M4) random forest, and (M5) gradient boosting (Géron, 2019). Hyperparameters were adjusted in the machine learning model to optimize recall, which is an indicator of the proportion of models correctly predicted as positive among actual positive instances. A high recall value indicates that the model successfully detected most of the lower-group data without omission.

A five-fold cross-validation was performed to evaluate the generalization performance for 126 records. The recall of the models ranged from 0.74 to 0.77, while the precision fell within the range of 0.70 to 0.78. Here, precision indicates the proportion that actually belong to the positive class among the instances that the model predicts to be positive. M1 achieved the highest recall and precision among all the models. Although it is generally difficult to correctly classify the minority lower group using statistical models, the low achievers belonging to the lower group could be classified with approximately 70% accuracy using the machine learning model.

5. Conclusion

In this study, we proposed using the number of token occurrences in student-created programs to detect low achievers in programming education in senior high schools. A machine-learning model was constructed to classify students into lower and non-lower groups using the programming data. The findings suggest that it is largely possible to identify students with lower proficiency using this method. Consequently, teaching staff can provide individualized instruction for such students.

For future research, we plan to analyze the temporal variation of the tokens in the program students are creating, to detect those likely to have weak programming-related mastery.

Acknowledgments

This research was partially supported by a Japanese Grant-in-Aid for the Encouragement of Scientists 24H02432.

References

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd Edition). O'Reilly Media.

Hasegawa, T., & Ito, Y. (2024a). Feature engineering for proficiency prediction based on programming learning process data. *Proceedings of the 39th Research Meeting of the Information Sub-Committee in the Japan Society of Technology Education* (pp. 67-70) (in Japanese).

Hasegawa, T., & Ito, Y. (2024b). Design and evaluation of a programming learning system with simple operation and sequential recording. *Information Education Journal of Naruto University of Education, 21*, 27-31 (in Japanese).

MEXT (Ministry of Education, Culture, Sports, Science and Technology). (2018). *Explanation of the courses of study for senior high schools, information edition* (in Japanese), Retrieved March 7, 2025, from https://www.mext.go.jp/content/1407073_11_1_2.pdf