Developing Program Grading Suggestion Mechanism by Reusing Teacher Grading Records

Chih-Yueh Chou, You-Chia Cheng
Department of Computer Engineering and Science, Yuan Ze University, Taiwan
cychou@saturn.yzu.edu.tw

Abstract: In this paper, we describe a work-in-progress research to develop a virtual grading assistant system for assisting teachers in grading programming assignments based on the output of student programs. The system provides teachers with an interface to grade student programs through marking token patterns of the program output and offering grades. The system records the teacher marking output token patterns and grades as teacher grading cases. The system will match the outputs of other student programs with the previous record of teacher grading cases to suggest grading to reduce the grading load. The paper presents the system prototype and proposes two research issues, which we will investigate.

Keywords: Virtual teaching assistant, computer assisted programming grading, token pattern matching

Introduction

Assessing and grading student programs are time-consuming and labor-intensive for teachers and teaching assistants, thus some computer systems are developed for assisting teachers in assessing student programs through assessing the program output and program code [1,2]. Most systems evaluate the function correctness of student programs by compiling and executing the programs with test inputs and comparing the output of student programs with that of the model program [3-5]. Generally, these systems evaluate the program through strict comparison of program output. It may arouse two limitations. First, a program with correct function but with minor different output form or order may be evaluated as wrong [6]. For instance, an assignment asks students to write a program to calculate and output the factorial result of an input integer number. If the input of a test case is 5 and the program output of the model program is "The factorial result of 5 is 120", a student program with the output "5! = 120" may be evaluated as wrong because the result of strict output comparison of these two outputs is different. Teachers usually setup rigid output format specifications to limit the student program output. Second, these systems evaluate programs either pass (correct) or failed (wrong), but teachers may grade programs within a range of grades from full mark to zero according to the correct ratio of the program output. Researchers proposed a program output comparison approach through token patterns to improve the flexibility of computer assessment [7].

Teachers can understand the meaning of program output and recognize the key token pattern, such as "120" in the factorial program when the test case input is 5, in program output, but computer assisted assessing systems are unable to understand program output and just compare the program output of student programs and that of the model program. Applying natural language processing techniques may solve this problem, but the system

development is complex. Researchers propose an approach of complementing machine intelligence and human intelligence to develop virtual teaching assistant systems [8-10]; that is, using human intelligence to reduce the complexity of system development and applying machine intelligence to reuse human intelligence. This study develops a computer assisted grading system, named ProgramHelper, to support teachers with an interface for marking token patterns on program output and giving corresponding grades (human intelligence). The system also records teacher marking token patterns and grades as grading cases for matching the output of other student programs and providing grading suggestion to reduce teacher load (machine intelligence).

1. System

The ProgramHelper system includes interfaces for teachers to assign and grade program assignments and an interface for students to submit programs; three databases to store program assignments, student submitted programs, and teacher grading cases; and three modules to compile and execute student programs, to match token patterns with the program output, and to suggest grades. Using ProgramHelper, a teacher assigns program assignments by describing program specification, offering inputs and outputs of test cases, and providing a model program. Students submit their programs for assignments and the system collects student programs for teacher grading. When a teacher wants to grade a student program, the system first compiles the program. If the program has compile errors, the system shows the error messages to the teacher. Otherwise, the system executes the program with the inputs of test cases and shows both the outputs of student programs and outputs of model program of all test cases to the teacher. The system will attempt to match the student program output with previous stored teacher grading cases to suggest grades. If the system did not contain any teacher grading cases or the student program output did not match with any teacher grading cases, the teacher needs to grade programs by himself/herself.

The system enables teachers to grade student programs by marking token patterns within the output of student programs and offering corresponding grades to these token patterns. The teacher also needs to assign the matching rule whether each token pattern should be matched at specific location or location-free so that the system can apply the rule to match the token pattern with the output of other student programs. For instance, if the input of a test case of factorial program is 5 and the output of a student program is “The factorial result of 5 is 120”. The teacher can mark “120” as a token pattern, offer 100 points to the token pattern, and assign the location-free matching rule. The system will store the token pattern, grade, and matching rule as a teacher grading case. When the teacher grades another student program and the output of the same test case is “5! = 120”, the system will found “120” matched and suggest 100 points. If a program aims to output all the values of a 2-subscribed array in two-dimensional form in order, the teacher needs to mark each value of the array as a token pattern, offer the corresponding grade, and assign each token pattern at specific location. For instance, the second array value is 4, has 5 points, and is located at the second token of the first row. If the system successfully found the output of the student program matched with some previous teacher grading cases, the system displays all matched teacher grading cases and matched token patterns of student program output in yellow background color, and suggests grades with the total of grades of all matched teacher grading cases. For instance, if the output of student program for calculating factorial contains the token pattern “120”, the system suggests 100 points based on the previous teacher grading case.

2. Summary and Future works

We developed a prototype system of reusing previous teacher grading cases to suggest grading based on the program output. The goal of the system is to frequently and correctly provide grading suggestions to reduce teacher load. The assumption is that the teacher grading cases on the program output can be reused for grading other student programs; that is, teacher grading cases are repeatedly occurred. We will apply ProgramHelper system to several program assignments to investigate the effectiveness of the system.

The ProgramHelper system is designed to assist teachers in grading student programs based on the program output. The system could be applied for other usages:

- Providing students with immediate feedback as formative assessment. After the system collected many teacher grading cases, the system can be used to immediately provide students with system grading when students submit programs. Although the system grading may not be exactly correct and teachers also require to confirm or to modify the grading to offer the final grade, the system grading can be a formative assessment.
- Finding out the programs which have correct functions but do not conform to the output format specifications. The system can compare the output of model program with that of student programs both through strict comparison and through token pattern comparison. If student programs pass token pattern comparison but do not pass the strict comparison, the programs have correct function but do not conform to the output format specification. The system can advise them to revise the program output format.

Acknowledgments

The authors would like to thank the support of the National Science Council (NSC 100-2511-S-155 -004 -MY3).

References

[1] Ala-Mutka, K. M. (2005) A survey of automated assessment approaches for programming assignments, *Computer Science Education*, 15 (2), pp. 83-102.

[2] Douce, C., Livingstone, D., & Orwell, J. (2005) Automatic test-based assessment of programming: A review, *ACM Journal on Educational Resources in Computing*, 5 (3), Article No. 4.

[3] Higgins, C., Hergazy, T., Symeonidis, P., & Tsinsifas, A. (2003) The CourseMarker CBA system: Improvements over Ceilidh. *Education and Information Technologies*, 8, 287 – 304.

[4] Cheang, B., Kurnia, A., Lim, A., & Oon, W.-C. (2003). On automated grading of Programming Assignments in an academic institution. *Computers & Education*, 41, 121-131.

[5] Joy, M., Griffiths, N., & Boyatt, R. (2005). The BOSS Online Submission and Assessment System, *ACM Journal on Educational Resources in Computing*, Vol. 5, No. 3, Article No. 2.

[6] Tang, C. M., Yu, Y. T., & Poon, C. K. (2009) Automated systems for testing student programs: Practical issues and requirements. *Workshop Proceedings of the 17th International Conference on Computers in Education (Workshop on SPECIAL)*, 132-136.

[7] Tang, C. M., Yu, Y. T., & Poon, C. K. (2009). An approach towards automatic testing of student programs using token patterns. *Proceedings of the 17th International Conference on Computers in Education (ICCE 2009)*, 188-190.

[8] Chou, C. Y., Chan, T. W., & Lin, C. J. (2003). Redefining the learning companion: the past, present, and future of educational agents. *Computers & Education*, 40(3), 255–269.

[9] Lin, C.J., Chou, C.Y., & Chan, T.W. (2008). Developing a computer-supported tutoring Interaction Component with Interaction Data Reuse. *Proceedings of the 9th International Conference on Intelligent Tutoring Systems (ITS 2008)*, 152-161.

[10] Chou, C. Y., Huang, B. H., & Lin, C. J. (2011) Complementary Machine Intelligence and Human Intelligence in Virtual Teaching Assistant for Tutoring Program Tracing. *Computers & Education*, 57 (4), 2303-2312.