Writing effective programming questions for Automated Grading using Bloom's Taxonomy

Abstract

Large introductory programming courses are experiencing an increase in class sizes, as they often see the enrollment of hundreds of students. Automated grading has become crucial in supporting courses of this size by assisting instructors in reducing grading time and course costs. However, there is an issue that sometimes novice programmers are frustrated by autograded systems as they don't provide enough feedback for complex questions. We have designed a study that will examine if gradually increasing question complexity, based on Bloom's Taxonomy, for auto-graded exercises will lead novice programmers in non-CS majors to learn more effectively and efficiently. Additionally, we will survey students to measure student perception on automated grading as well as their self-efficacy regarding their programming skills. Examining perception is important because providing a positive student experience is critical for cultivating an effective learning environment.

Question difficulties are rated in regards to the revised Bloom's Taxonomy. To measure effective learning, we will analyze student exam scores after the completion of a lab that consists of automatically graded programming exercises. To measure efficient learning, we plan to analyze the time spent on each exercise, number of attempts made before submitting an exercise, and the following exam score.

We conjecture that scaling the question complexities from lower to higher orders of thinking will produce the most effective and efficient student learning. We expect to see that this modified progression of exercises will result in higher student self-efficacy and more positive opinion of the exercises and the automated grading itself. The three research questions we seek to answer are as follows:

- **RQ1:** Does scaling exercise difficulty on Cody Coursework impact student performance?
- **RQ2:** Is student learning efficiency affected by scaling exercises on Cody Coursework?
- **RQ3:** How do students perceive their learning experience when question difficulty on an AGT is scaled?

Background

As AGTs have become an essential component in computer science education, they have also been subjected to numerous studies. The bulk of these studies cover changes in student performance after the integration of an AGT as well as student perception of AGTs. A considerable amount of

There are numerous Autograders available for beginner programmers and instructors. A few popular examples include CodingBat (Parlante), Practice-it (Bowden), and Gradescope (Singh). The AGT we utilize is Cody Coursework; Cody Coursework is unique in the fact that it grades programs written in the proprietary language MATLAB, owned by MathWorks. It is also web-based, similar to CodingBat and Practice-it, but Cody Coursework allows for instructors to write their own exercises and test cases.

AGTs have multiple applications in computer science classrooms, and typically are used in a way that complements the instructor's taste. The three most common include: an in-class active learning supplement(Benotti, Pettit), as a laboratory grading platform (Pettit, Barr), and as assigned homework(Benotti, Pettit).

AGTs have been shown to **benefit student performance** in several regards. Courses that have implemented AGTs have experienced reduced dropout rates (Pettit). In the case of two Argentinian Universities, UTN and UNC, they experienced an early drop-out rate decrease from 28% to 14% and 58% to 35% respectively (Benotti). Those students had successfully passed the first exam on the first try or on the re-test, as the failure rate had remained relatively unchanged. The improvements in student retention and passing rates were attributed to allowing students to learn at their own pace. Classrooms that have implemented AGTs in their coursework have also experienced an overall increase student grades (Pettit, Cohenour). In the case of Oregon University, the first semester that integrated CodeLab (Barr) saw the average class GPA rise from 2.0 to 2.2 on a 4-point grading scale.

In a large meta study (Pettit) that analyzed AGTs in computer science education, they had deemed that **student perception** was ambiguous. However, it is important to note that many of the negative-opinion papers, and some of the positive-opinion papers are over ten years old, and may contain outdated perceptions. We intend to study student perception of AGTs to assist in providing a current look at student perceptions of AGTs.

Students who rated the tool positively often praised the opportunity to receive feedback before the final submission of the assignment (Benotti, Selby), which encouraged them to make

multiple attempts to strive for a higher grade. This increased exposure to the material may have positively impacted their overall understanding of the material as well.

On the other hand, students who had rated the AGTs negatively almost unanimously disliked the level of precision required from their programs (Pettit). The finicky nature of some AGTs would penalize students who may have been close to the correct solution, leading the students to still receive a low grade (Pieterse). Some students also showed skepticism of the grading without human intervention.

Bloom's Taxonomy is one of the most commonly used models that describes a learner's level of understanding of a topic based on cognitive domains. It has been effective in assisting instructors in various fields in structuring coursework, homework assignments, and assessments (Bruyn, Dorodchi, Errol, Selby). The taxonomy has also been widely used in computer science education to provide a way for instructors to accurately compare programming question complexities across several topics (Dorodchi). Lastly, it has also been used to teach computational thinking while using programming as a tool (Selby).

We wish to combine these areas of research by extending the learning concepts proposed by the revised Bloom's Taxonomy to the domain of AGTs. Specifically, we analyze if scaling question difficulty from the lower orders to the higher orders of Bloom's taxonomy within AGTs will increase student performance and perception; to our knowledge this is the first study of its kind.

Methods

We are implementing our study into the lab sessions of an introduction to MatLab course at North Carolina State University. This course, similar to many other programming courses, consists of a large lecture session of around 250 students, with 8 weekly laboratory sections made up of about 20-30 students each. The lab sessions are 2 hours and 45 minutes, and begins with review session led by a teaching assistant. After review, students are asked to complete 3-5 exercises on Cody Coursework in which they can work with a partner. These exercises conclude lab activities. Throughout the course, there are formative exams in the weeks following labs 4 and 8, and a final summative exam after lab 10.

For our study, we divide the lab sections into 4 experimental sections, and 4 control sections. We are imposing a special treatment to labs 4 and 8: experimental sections will solve 4 exercises that progressively become more complex in regards to Bloom's Taxonomy, whereas the control sections will solve the same questions, but unordered. After the students solve these exercises, they will then complete a short post-lab survey measuring their perceptions of their

self efficacy, the exercises themselves, and Cody Coursework. We will be collecting data regarding the number of pretests the students have ran, the time it takes for them to complete the exercises, and we will also be implementing questions in the formative exams following labs 4 and 8 that target the topics covered in labs 4 and 8. The time taken and number of pretests compared to correctness of target exam questions will measure learning efficiency. Correctness of targeted exam questions and correctness of check-in question will measure learning effectiveness.

Following the final lab, lab 10, the students will also complete and end of semester survey, which is more in-depth than the post lab surveys and includes more questions about the students' perceptions of Cody Coursework, followed by the exercises, followed by their self efficacy. This final survey, along with the previously discussed post-lab surveys will measure student perceptions.

We will be comparing data between the control and experimental sections based on lab and the lab's following exam. We will not be comparing data between labs 4 and 8 to draw our conclusions. This applies to all the metrics we are using to measure learning efficiency, learning effectiveness, and student perception.

References

Barr, V., & Trytten, D. (2016). Using Turing's Craft Codelab to Support CS1 Students as they Learn to Program. *ACM Inroads,7*(2), 67-75. doi:10.1145/2903724

Benotti, L., Aloi, F., Bulgarelli, F., & Gomez, M. J. (2018). The Effect of a Web-based Coding Tool with Automatic Feedback on Students Performance and Perceptions. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education - SIGCSE 18*. doi:10.1145/3159450.3159579

Bowden, G., & Maguire, K. (n.d.). Practice IT [Computer software].

Bruyn, E. D., Mostert, E., & Schoor, A. V. (2011). Computer-based testing - the ideal tool to assess on the different levels of Blooms taxonomy. *2011 14th International Conference on Interactive Collaborative Learning*. doi:10.1109/icl.2011.6059623

Cohenour, C., & Hilterbran, A. (2017). Automated Grading of Access® Databases Using the Matlab® Database Toolbox. *2017 ASEE Annual Conference & Exposition Proceedings*. doi:10.18260/1-2--27647

Dorodchi, M., Dehbozorgi, N., & Frevert, T. K. (2017). "I wish I could rank my exams challenge level!": An algorithm of Blooms taxonomy in teaching CS1. *2017 IEEE Frontiers in Education Conference (FIE)*. doi:10.1109/fie.2017.8190523

Parlante, N. (n.d.). CodingBat [Computer software]. Retrieved August 2, 2018, from http://codingbat.com/java

Pettit, R., Homer, J., Holcomb, K., Simone, N., & Mengel, S. (n.d.). Are Automated Assessment Tools Helpful in Programming Courses? *2015 ASEE Annual Conference and Exposition Proceedings*. doi:10.18260/p.23569

Pieterse, V., & Liebenberg, J. (2017). Automatic vs Manual Assessment of Programming Tasks. *Proceedings of the 17th Koli Calling Conference on Computing Education Research - Koli Calling 17*. doi:10.1145/3141880.3141912

Selby, C. C. (2015). Relationships. *Proceedings of the Workshop in Primary and Secondary Computing Education on ZZZ - WiPSCE 15*. doi:10.1145/2818314.2818315

Singh, A., Karayev, S., Awwal, I., & Abbeel, P. (n.d.). Gradescope [Computer software]. Retrieved August 2, 2018, from https://gradescope.com/

Thompson, E., Luxton-Reilly, A., Whalley, J. L., Hu, M., & Robbins, P. (2008). Bloom's Taxonomy for CS Assessment. *Australian Computing Education Conference (ACE2008),78*(10th). Retrieved July 5, 2018, from http://infotech.scu.edu.au/~ACE2018/