ISBN: 978-989-8704-08-5 © 2014

USING CASE-BASED REASONING TO IMPROVE THE QUALITY OF FEEDBACK PROVIDED BY AUTOMATED GRADING SYSTEMS

Angelo Kyrilov & David C. Noelle
Electrical Engineering and Computer Science, University of California, Merced
5200 N Lake Road, Merced, California, USA

ABSTRACT

Information technology is now ubiquitous in higher education institutions worldwide. More than 85% of American universities use e-learning systems to supplement traditional classroom activities while some have started offering Massive Online Open Courses (MOOCs), which are completely online. An obvious benefit of these online tools is their ability to automatically grade exercises submitted by students and provide immediate feedback. Most of these systems, however, provide binary ("Correct/Incorrect") feedback to students. While such feedback is useful, some students may need additional guidance in order to successfully overcome obstacles to understanding. We propose using a Case-Based Reasoning (CBR) approach to improve the quality of feedback Computer Science students receive on their programming exercises. CBR is a machine learning technique that attempts to solve problems based on previous experiences (cases). The basic idea is that every time the instructor provides feedback to a student on a particular exercise, the information is stored in a database system as a past case. When student experience similar problems in the future, knowledge contained in past cases is used to guide the students to a solution. While the system will provide detailed feedback automatically, this feedback will have been previously crafted by human instructors, leveraging their pedagogical expertise. We describe a system of this kind, which is currently under development, and we report results from a preliminary experiment.

KEYWORDS

Case-based reasoning, e-learning, e-assessment, immediate feedback.

1. INTRODUCTION

Higher education institutions around the world are rapidly adopting information technology to help students. According to Chen et al. (2010), more than 85% of universities in America make use of Learning Management Systems (LMSs), such as Moodle and Sakai. Such systems allow students access to course materials, discussion forums, and other resources. LMSs facilitate the online submission of assignments or quizzes, in addition to managing enrollments and grades. Researchers have found that using LMSs to supplement the traditional classroom environment has been beneficial to students (Chen et al, 2010).

Some institutions have recently started providing Massive Online Open Courses (MOOCs), where the entire learning experience is online. Organizations like EdX and Coursera offer online courses from leading universities. All instruction is done using pre-recorded video lectures, and communication with instructors is facilitated through online forums. Quizzes and assignments are submitted online and are graded either by fellow students (peer grading) or automatically (e-assessment), with feedback often provided instantly.

Immediate feedback has been shown to facilitate deeper learning (Epstein et al, 2002) and is generally regarded as an advantage. Most automated grading systems, however, only provide binary feedback, in the form of "Correct/Incorrect". In the case of an incorrect solution, these automated graders are not able to provide guidance to the student. Some intelligent tutoring systems attempt to provide richer feedback, but this approach is not trusted by many educators (Van Lehn, 2011).

In this paper, we propose a methodology for improving the quality of feedback provided by automated grading systems. The proposed framework is built upon the principles of Case-Based Reasoning (CBR). While the idea of using CBR for e-learning applications is applicable to a wide range of academic disciplines, we focus on improving the quality of feedback given to undergraduate Computer Science students on solutions to programming exercises.

The rest of the paper is organized as follows: Section 2 introduces the reader to the Case-Based Reasoning methodology and provides examples of systems where CBR has been successfully applied. Section 3 provides the details on our proposed e-learning system. Section 4 presents the results of preliminary experiments, and Section 5 contains a discussion and concluding remarks.

2. BACKGROUND AND RELATED WORK

Case-based reasoning is a machine learning technique that uses knowledge from past experiences (or cases) in order to find solutions to current problems. Since case-based reasoning is a lazy technique, it builds up its knowledge base by simply storing past cases into a database. A case is defined as a description of a problem along with a solution to the problem. For example, the technical support department of an Internet service provider may employ a CBR system. A typical case in their database would have a description of the symptoms, say “Modem LED blinking rapidly”, and a solution, for example, “Try restarting the modem”.

A new problem being encountered is called a “query”, presenting a description of the problem symptoms. The CBR system uses its knowledge base to try to find a solution to the query. There are typically four steps in the process.

1. Retrieve: In this step, the CBR system finds all cases whose problem descriptions are similar to the query. The most critical component here is the metric used to measure the similarity between the query and the past cases.
2. Reuse: Using the solution(s) retrieved in the previous step, construct a solution to the query.
3. Revise: Evaluate the quality of the solution provided in the last step.
4. Retain: Based on the outcome of the revise step, decide whether to store the query and the solution as a case in the database, or not.

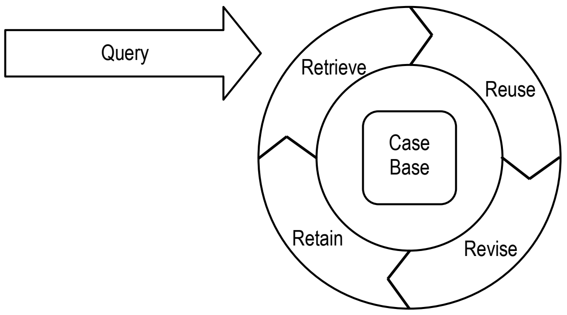

Figure 1. The case-based reasoning methodology, adapted from López (2013)

Figure 1 illustrates the steps of the CBR process. For more details on the methodology, see López (2013). CBR systems have been successfully applied in customer support centers, as shown in Allen (1994), while Jo et al. (1997) used a CBR approach to predict bankruptcy in Korea. Begum et al. (2010) surveyed a number of CBR systems used in the health sciences and found that these systems are being used for a wide variety of tasks, such as diagnosis, treatment planning, and training of medical personnel.

CBR has also been applied to the educational sector, although it has not received as much attention as it has in other domains. Jonassen and Hernandez-Serrano (2002) proposed a CBR system to support problem solving using stories, and Ballera et al. (2013) proposed a CBR system to personalize the e-learning experience of students by sequencing the topics being presented. Wiratunga et al. (2011) presented RubricAce, a CBR system designed to assist instructors who use rubrics for grading students’ work. RubricAce suggests feedback comments to instructors once they have assigned grades according to a rubric. Instructors then decide how to use the suggested feedback to provide summative evaluations to students. One of our goals is to provide high quality immediate formative feedback, to the degree that such is possible. To our knowledge there have been no previous attempts to use CBR in order to improve the quality of immediate feedback provided to students by e-learning systems. In the next section we describe a framework for achieving this.

3. PROPOSED SYSTEM

We now present the details of our proposed methodology for improving the quality of feedback generated by automated grading systems. We assume that a system for providing "Correct/Incorrect" feedback already exists. In the Computer Science domain, it is easy to implement a system that will distinguish correct from incorrect programming assignment submissions. Upon creating a programming exercise, the instructor provides test cases that are used to verify the correctness of student-submitted solutions. This is the "Grading System" module in Figure 2. When a student submits his/her first solution to a given exercise, the code is sent to the grading system and if it is determined to be correct, the student is given positive feedback. If, however, the solution is incorrect, the CBR component of the system comes into play.

First, the system searches the case base for past cases that are similarly incorrect to the current submission. Two programs are similar with respect to incorrectness when they both contain the same bugs, or suffer from the same problems. A possible metric for establishing incorrectness similarity is to treat two submissions as similar if they both fail on a common set of test cases. This metric was used in the preliminary experiment described in Section 4. Other possible metrics, such as static code analysis, are be briefly discussed in section 5 and will be further explored in future studies.

The past cases that are retrieved will contain pedagogically appropriate guidance on how to resolve the problems that are being experienced. Pedagogical appropriateness is determined by instructors who generate the feedback. They may generate different kinds of feedback for different students, experiencing the same problem. The system can use student-specific as well as submission-specific features to determine when a given kind of feedback is to be generated.

Before the feedback is presented to the student, the system determines whether human intervention is necessary, which would be when the system is unable to retrieve similar cases, or when the student has repeatedly failed to rectify a particular problem, indicating that the feedback received has not been helpful. If it is determined that intervention is needed, a human instructor will manually examine the feedback and update it as necessary before it is given to the student. If intervention is deemed unnecessary, the system-generated feedback is passed directly to the student, who uses it to update and resubmit his/her solution.

The resubmitted solution is graded again. If it is correct, it shows that the feedback that was generated by the system was, indeed, helpful to the student, and the entire submission history of this interaction with the student is used to update the case base. If the resubmitted solution is still incorrect, it is pushed through the CBR process repeatedly, including potential human intervention by members of the teaching team, until the student succeeds at the exercise.

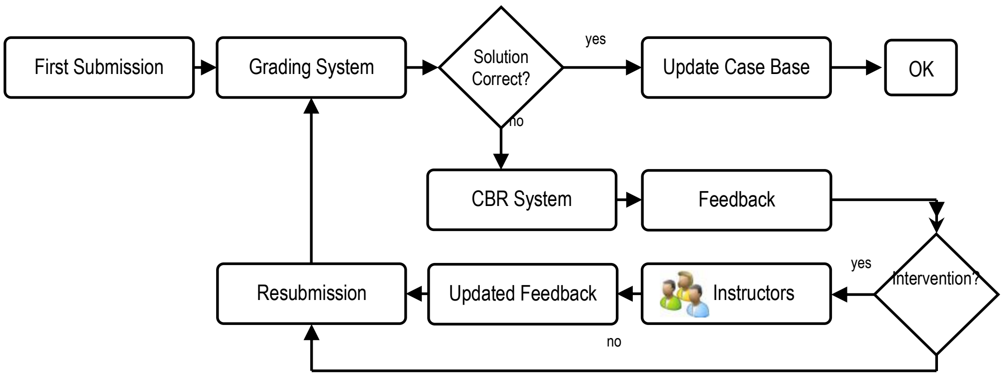

Figure 2. A flowchart of the proposed system

4. PRELIMINARY RESULTS

At our institution we use an online grading system to automatically evaluate programming assignments completed by undergraduate students on a weekly basis. The correctness of each student submission is determined by compiling and running their code against predefined test cases. We have collected submission data from several undergraduate computer science courses over the past 2 years.

As a proof-of-concept we have implemented a simple CBR system in order to gauge the potential impact of the methodology. We use a simple incorrectness similarity measure, where two programs are declared similarly incorrect if they fail on a common subset of the test cases.

For the purposes of this experiment, we clustered submissions based on the incorrectness similarity measure above in order to determine whether instructor-generated feedback for one member of a cluster will be applicable to the rest of the members. Cases in which the previously generated feedback is not applicable are called "interventions". We randomly sampled 3 out of the 82 exercises assigned in an undergraduate computer science course with 60 enrolled students. The exercises are labeled 1-3 in Table 1. Exercise 1 asked students to read in an integer *n* and sum up all integers from 1 to *n*. Exercise 2 involved reading from a text file and counting the occurrences of a given word. Exercise 3 was a simple OOP exercise where students had to create a "circle" class that has radius and area. Table 1 summarizes the results of the experiment.

Table 1. Results of clustering experiment

Exercise	Num. of incorrect submissions	Clusters by system	Clusters by instructor	Interventions needed
1	123	28	9	3
2	180	13	9	1
3	149	14	11	0

Despite the simple incorrectness-similarity metric used in this experiment, Table 1 shows that significant labor savings can be had. For example, the third exercise would have required 149 separate interactions between an instructor and a student, but the system reduced the number of required human-generated guidance passages to 14, resulting in a savings of over 90% with regard to instructor time. In addition, the system allowed all students who submitted incorrect solutions to be assisted in a timely manner, with most students receiving appropriate feedback instantaneously, all while requiring the instructor to do roughly 10% of the work that he/she would have had to do without the system.

5. DISCUSSION AND CONCLUSIONS

Our proposed e-learning system makes use of a case-based reasoning mechanism to improve the quality of feedback by looking at previously encountered incorrect submissions and the steps that were taken to resolve them. Once an instructor has determined the pedagogically appropriate feedback to remediate a particular misunderstanding, and that information is stored in the case base, students who share that misunderstanding will receive useful guidance immediately. In this way, our proposed system combines the rapidity of feedback delivery present in modern e-learning systems with the integration of human teaching expertise, which is used to design rich and detailed feedback.

A metric that captures similarity with respect to incorrectness is an essential component of our CBR system. This metric allows the system to find previously submitted incorrect solutions that have similar flaws. Identifying a good similarity metric for this process is not easy. While one might imagine assessing similarity by performing some form of direct comparison of source code, we have found such approaches to contain hidden problems. First, surface features of source code, such as the names of identifiers, are typically irrelevant to the nature of the errors in a program. Minimally, some analysis of the logical structure of the source code samples must be done. There are existing tools for performing comparisons of the logical structure of two source code samples, but we have yet to find a way to leverage these tools for identifying common errors. For example, the popular MOSS system for detecting plagiarism in computer programming assignments does an excellent job of identifying pairs of programs that pass a threshold of similarity, suggesting a common origin for the two programs (Schleimer, S. et al, 2003). Upon careful examination of this system, we discovered that the kinds of comparisons needed to detect plagiarism are nothing like those needed to detect common errors. Consider two programs that take very different approaches to solving a problem (e.g., an iterative approach versus a recursive approach), but make a common error in some small section of code, such as failing to initialize a critical variable. In this case, MOSS would rate the two programs as highly dissimilar, despite the fact that they share a common error. Conversely, consider two programs that are virtually identical, except for the fact that one makes a particular error in one line of code while the other does not. In this case, MOSS would flag the two programs as highly similar, despite the fact that they do not share an error. If the logical structure of source code is to be used as part of the similarity measure for the proposed CBR system, it will need to be carefully crafted to focus on similarities in terms of common errors. One possible approach would involve allowing human instructors to annotate source code submissions, identifying the sections that are most relevant for the detected error.

In addition to exploring approaches to similarity focusing on such a static analysis of code, we intend to continue to investigate similarity metrics based on the common failure of programs on carefully crafted arrays of test cases. Possibilities include methods for automatically checking that individual test cases are discriminative of particular errors, comparing the output generated by test cases in more detail, and applying additional machine learning methods to learn to discriminate between error types based on a variety of both static and dynamic features. Since some errors may mask the presence of others, we are also designing methods for maintaining a hierarchy of potential errors, keeping the similarity metric from being deceived by such masking.

Computers are ubiquitous in higher education institutions, and they are increasingly being used to improve the quality of the education being offered. E-learning systems have the ability to offer students immediate feedback, but it is often of the form of “Correct/Incorrect”. While such binary feedback is helpful, students may need additional guidance. We propose a case-based reasoning approach in order to try to improve the quality of feedback generated by automatic grading systems. We conducted a proof-of-concept experiment to determine the usefulness of CBR in a Computer Science e-learning system. Despite using a very simple incorrectness-similarity metric, our results are promising. Research is currently underway to improve the incorrectness similarity metric and to test the complete system in live classroom environments.

REFERENCES

Allen, B., 1994. Case-Based Reasoning: Business Applications. *In Communications of the ACM*, Vol. 37, No 3, pp 40-42.

Ballera, M. et al, 2013. Personalizing and Improving e-Learning System Using Roulette Wheel Selection Algorithm, Reinforcement Learning and Case-Based Reasoning Approach. *Proceedings of The Fourth International Conference on e-Learning (ICEL 2013),* Ostrava, Czech Republic, pp. 184-193.

Begum, S. et al, 2010. Case-Based Reasoning Systems in the Health Sciences: A Survey of Recent Trends and Developments. *In IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 41, No. 4, pp. 421-434.

Chen, P. et al, 2010. The impact of Web-based learning technology on college student engagement. *In Computers & Education,* Vol. 54, pp. 1222-1232.

Epstein, M. et al, 2002. Immediate Feedback Assessment Technique Promotes Learning and Corrects Inaccurate First Responses. *In The Psychological Record*, Vol. 52, No. 2, Article 5.

Jo, H. et al, 1997. Bankruptcy Prediction Using Case-Based Reasoning, Neural Networks, and Discriminant Analysis. *In Expert Systems With Applications,* Vol. 13, No. 2, pp. 97-108.

Jonassen, D. and Hernandez-Serrano, J., 2002. Case-Based Reasoning and Instructional Design: Using Stories to Support Problem Solving. *In ETR&D,* Vol. 50, No. 2, pp. 65-77.

López, B., 2013. *Case-Based Reasoning: A Concise Introduction*. Morgan & Claypool Publishers, San Rafael, USA.

Schleimer, S. et al, 2003. Winnowing: Local Algorithms for Document Fingerprinting. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* San Diego, USA.

Van Lehn, K., 2011. The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems. *In Educational Psychologist*, Vol. 46, No 4, pp. 197-221.

Wiratunga, N. et al, 2011. RubricAce: A Case-based Feedback Recommender for Coursework Assessment. *Proceedings of the Sixteenth UK Workshop on Case-Based Reasoning (UKCBR 2011),* Cambridge, United Kingdom