AN AI SYSTEM FOR COACHING NOVICE PROGRAMMERS

An Undergraduate Research Scholars Thesis

by

GILBERT CRUZ, JACOB JONES and MEAGAN MORROW

Submitted to the Undergraduate Research Scholars program
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Research Advisor: Dr. Bruce Gooch

May 2016

Major: Computer Science and Engineering

TABLE OF CONTENTS

ABSTRACT

An AI System for Coaching Novice Programmers

Gilbert Cruz, Jacob Jones and Meagan Morrow
Department of Computer Science and Engineering
Texas A&M University

Research Advisor: Dr. Bruce Gooch
Department of Computer Science and Engineering

Rapidly giving students meaningful feedback is a key component in an effective educational experience. A common problem in modern education is scalability, as class size increases the ability of the instructor to rapidly provide meaningful feedback decreases. Our team of undergraduates are preparing Java language platform for beginning programmers. The team has done background reading and had discussions on meaningful feedback for novice programmers over the last year. As a result we are building an online Artificial Intelligence (AI) system capable of providing insightful narrative based coaching to beginning programmers. We will then evaluate the system to insure that it meets the following criteria: it generates a unique narrative response for every input, response is generated in real time, the system is deployable online.

SECTION I

INTRODUCTION

Background

First year Computer Science carries the highest cognitive load of any academic disciple [1 - 6]. Students have to learn the problem the solving skills of Engineers in addition to a new form of language. Compounding this problem is the widespread use of Mathematics pedagogy. Exercises and tests use a “fill in the blank” model where students are given nonworking computer code or a specification and expected to generate a solution.

Dr. Gooch found more effective methods of teaching introductory computer science by incorporating cognitive linguistics research [7 - 12]. The most effective result is changing the Exercise format from “fill in the blank” to creative writing. Dr. Gooch gave students working code and educational scaffolding based on industry practice. Students extend the starter code to increase functionality or efficiency. Then test their code using data they are responsible for creating.

Current auto grading software uses a Compile, Correct, Copied model [13]. The software checks that student submissions compile, run, produce correct output, and do not constitute plagiarism. Such software does not provide meaningful feedback [14-15].

Objective

Our team will design, build and evaluate a system to provide narrative based feedback and coaching in real-time. The system will first perform the standard Compile, Correct, Copied checks. If the submission passes a set of metrics will be computed that compare the submission to a code database including; the starter code, the students previous work, model code, and submissions from previous classes. Feedback in the form of a unique narrative will be generated that details the effects of the students modifications (speed, functionality, memory usage, etc.), tips on how they might improve both the functionality and readability of their code, how is their work is novel when compared with their peers, and an evaluation of how their coding ability is improving.

SECTION II

METHODS

Materials

In order to reach multiple goals and manage our project development, our team implemented a project management task known as scrum. This allowed for the building and testing of the AI system to have a designated timeline. Our team also used our personal past assignments as a sample material. These assignments are located in a database and will be used as a testing aid when implementing our AI. To create these databases, we used a database manage system known as MariaDB. For the generation of individualized and unique reports, our team will use the KPML system[20]. This system is a platform that uses grammar engineering for text generation based on sample sentences in which we have provided to the software.

Procedures

As a basis of this study, we used a bottom up design process shown to be useful in Computational Linguistics [18], Applied Computing [17] and Machine Learning [16]. Our team created two databases that hold information that will be used as either the input or the output. The first database, the input, is a collection of old coding assignments. We will compare the students turned in code with the database to check for similarity between their code and already turned in assignments. This is an effort to instantly check for plagiarism, whether a student has copied an old assignment or has turned in a similar assignment as their peer. We will also use the first database to see how a student has improved over time in a statistical manner. The second

database, the output, is a collection of feedback text we have coined as “mad libs.” This text will give students personalized feedback after they have turned in their assignment. The feedback is based on their personal information and how well they did on their assignment.

The team has also created an all-encompassing script that executes everything in order and neatly package the different outputs to be passed to the database. We have written a Java program to accomplish this task using Unified Code Count [19] and Maven. This Java code is called and Unified Code Count is executed along with a newly created Maven project. Once an output is generated, the Java code compiles all the useful statistics into a final output for the database. We are able to delete unnecessary files and statistics once a final output is created. Figure 1 shows the design of the processes used in creating the AI.

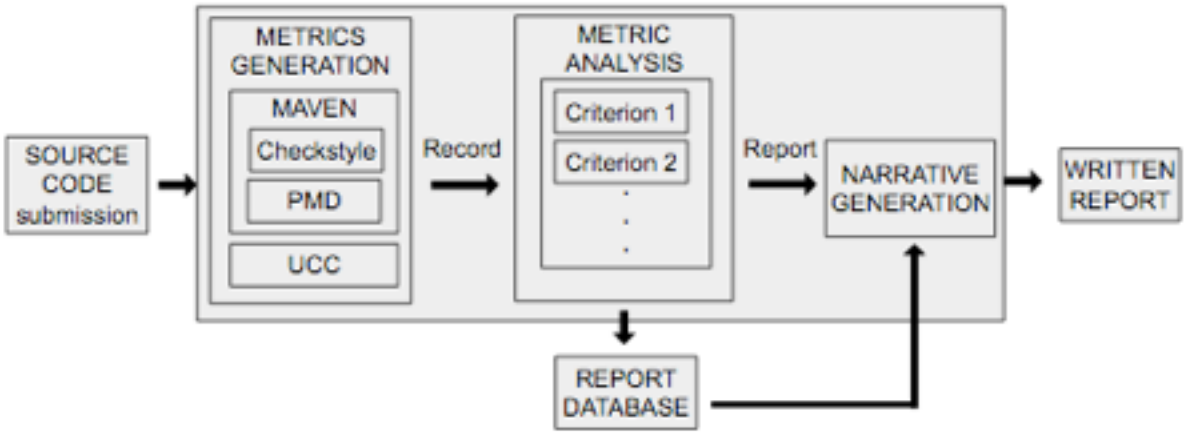

Figure 1: This figure encompasses a break down of the system our team used when creating our AI. It also demonstrates how each component of the design is related and interconnected.

CHAPTER III

RESULTS

Our team has created Java code that can remotely gain accesses and inject SQL code into our database. This code is imbedded in our report that runs when we call to make a report. The report uses the Java code to access the tables in our database of the student information. The final program uses a CSV file that is generated with the help of our entire research team. The program will use the CSV file as an input file to generate a report for the user allowing them to know where they have improved and in what areas they need improvement based off of the specific data and statistics generated from the analysis programs. The purpose of the program that is being built is to avoid making the report seem like it was automatically generated by a computer, but rather make it seem as if another person generated it for them. Every student's progress is stored for the purpose of creating new reports without making them sound too familiar to previous generated reports. The finalized report is designed in an easy to read manner that will hold specific detail that is relevant to the students as individuals. This type of result aids in coaching a student based off of their personal improvement rather than in comparison to fellow students. Figure 2 demonstrates the metrics and criteria use to generate the report.

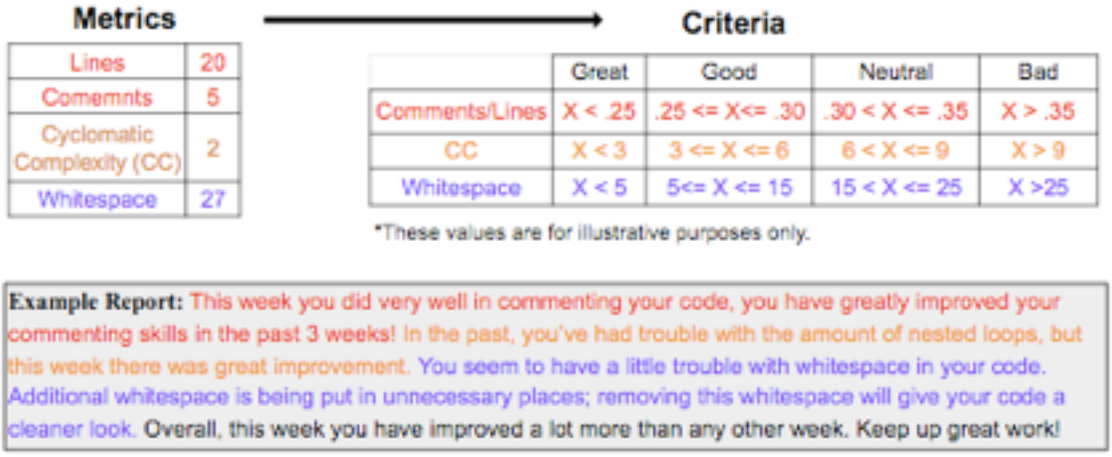

Figure 2: This figure is a demonstration of possible metrics that are generated in the statistical analysis programs, the tentative criteria guidelines for illustrative purposes, and an example report designed based on the given metrics and criteria.

REFERENCES

[1] Tuovinen, Juhani E.; Sweller, John A comparison of cognitive load associated with discovery learning and worked examples. Journal of Educational Psychology, Vol 91(2), Jun 1999, 334-341.

[2] John Sweller Cognitive load theory, learning difficulty, and instructional design. Learning and Instruction Volume 4, Issue 4, 1994, Pages 295–312.

[3] Pea, Roy D., and D. Midian Kurland. “On the cognitive effects of learning computer programming.” New ideas in psychology 2.2 (1984): 137-168.

[4] Jenkins, Tony. “On the difficulty of learning to program.” Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences. Vol. 4. 2002.

[5] Pennington, Nancy. “Stimulus structures and mental representations in expert comprehension of computer programs.” Cognitive psychology 19.3 (1987)

[6] Mayer, Richard E. “Different problem-solving competencies established in learning computer programming with and without meaningful models.” Journal of Educational Psychology 67.6 (1975): 725.

[7] Soloway, Elliot, and James C. Spohrer. Studying the novice programmer. Psychology Press, 2013.

[8] Gomes, Anabela, and António José Mendes. “Learning to program-difficulties and solutions.” International Conference on Engineering Education–ICEE. Vol. 2007. 2007.

[9] Du Boulay, Benedict. “Some difficulties of learning to program.” Journal of Educational Computing Research 2.1 (1986): 57-73.

[10] Lahtinen, Essi, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. “A study of the difficulties of novice programmers.” ACM SIGCSE Bulletin. Vol. 37. No. 3. ACM, 2005.

[11] Mayer, Richard E. Teaching and learning computer programming: Multiple research perspectives. Routledge, 2013.

[12] Tan, Phit-Huan, Choo-Yee Ting, and Siew-Woei Ling. “Learning difficulties in programming courses: undergraduates’ perspective and perception.” Computer Technology and Development, 2009. ICCTD’09. International Conference on. Vol. 1. IEEE, 2009.

[13] Ala-Mutka, Kirsti M. “A survey of automated assessment approaches for programming assignments.” Computer science education 15.2 (2005): 83-102.

[14] Ihantola, Petri, et al. “Review of recent systems for automatic assessment of programming assignments.” Proceedings of the 10th Koli Calling International Conference on Computing Education Research. ACM, 2010.

[15] Morris, Derek S. “Automatic grading of student’s programming assignments: an interactive process and suite of programs.” Frontiers in Education, 2003. FIE 2003 33rd Annual. Vol. 3. IEEE, 2003.

[16] Mary Elaine Califf and Raymond J. Mooney. Bottom-up relational learning of pattern matching rules for information extraction. J. Mach. Learn. Res., 4:177–210, 2003.

[17] Byeong-Mo Chang, Kwang-Moo Choe, and Roberto Giacobazzi. Abstract filters: improving bottom- up execution of logic programs by two-phase abstract interpretation. In Proceedings of the 1994 ACM symposium on Applied computing, pages 388–393. ACM Press, 1994.

[18] Gregor Erbach. Bottom-up earley deduction. In Proceedings of the 15th conference on Computational linguistics, pages 796–802. Association for Computational Linguistics, 1994.

[19] http://sunset.usc.edu/ucc_wp/

[20] http://www.fb10.uni-bremen.de/anglistik/langpro/kpml/readme.html