A Socratic Tutor for Source Code Comprehension

Zeyad Alshaikh(✉), Lasagn Tamang(✉), and Vasile Rus(✉)

The University of Memphis, Memphis, TN 38152, USA
{zlshaikh,ljtamang,vrus}@memphis.edu

Abstract. Reported here are the findings of a comparative study on the effects of using a Socratic Intelligent Tutoring System for source code comprehension and learning computer programming. The result shows there are significant differences between the two groups where students who used Socratic Tutor ITS improved their knowledge by 45% in term of learning gain, developed a better understanding of concepts such as nested if-else and for loop, and improved their confidence level by 13%. Furthermore, the result of the Pearson product-moment correlation coefficient shows a positive correlation (r = 0.68) between feedback from the ITS and learning gain.

Keywords: Socratic method · Computer science education · Computer programming · Intelligent Tutoring System

1 Introduction

Introductory programming courses are difficult [10], frustrating [8], and often considered a major stumbling block for many students [15]. There is much evidence that drop-out and failure rates in introductory Computer Science courses such as CS1 and CS2 are high (30–40%) [3,12,16].

Intelligent Tutoring Systems have been proven to be beneficial solutions that can provide individualized, one-on-one instruction for all students [1], and improve the quality and effectiveness of computer programming instruction [14]. As a result, many ITS systems were developed as early as 1974 [7] to aid students on different programming phases [6,9,10,17,18].

In our case, we developed a dialogue-based intelligent tutoring system called Socratic Tutor to help novice programmers acquire deep and robust programming knowledge by engaging in source code understanding learning activities. The Socratic Tutor is inspired by the Socratic instructional strategy [4] in the form of a set of guiding questions meant to provide students a form of scaffolding by targeting key aspects of the instructional task. Furthermore, the developed system relies on self-explanation theories of learning [5] by implementing instructional strategies such as eliciting self-explanations through Socratic questioning.

Socratic Tutor ITS uses a natural language understanding (NLU) engine [2] to evaluate students' responses with respect by computing a semantic similarity score to model/benchmark correct answers and well-known misconceptions

I. I. Bittencourt et al. (Eds.): AIED 2020, LNAI 12164, pp. 15–19, 2020.
https://doi.org/10.1007/978-3-030-52240-7_3

created prior by experts. Therefore, the NLU engine enables the Socratic Tutor to immediately detect misconceptions and provide tailored feedback which was proven to have a positive benefit on learning [13]. The developed system provides help to students using a three-level feedback strategy where at level one the tutor explains briefly the target concept and gives the student a second chance to retry answering the original question. At levels two and three, the tutor asks questions in the form of multiple-choice and fill-in-the-blank questions.

This paper analyzes a comparative study of using the Socratic Tutor ITS for learning JAVA programming in Introductory to Computer Science courses (CS1 and CS2) focusing on arithmetic operations, nested $if - else$, $while$ loops, for loops, arrays, and class.

2 Research Questions

To understand Socratic Tutor's impact on students programming knowledge and other characteristics such as confidence, we have conducted a study focusing on the following research questions: (1) how much do students learn when using Socratic Tutor?, (2) how much do students learn on each targeted programming concept?, (3) how much does the Socratic Tutor have an impact on students' self-confidence?, and (4) what is the relationship between feedback and learning gains?

3 Method

Subjects who participated in the study were undergraduates ($n = 70$) enrolled in the Introductory to Computer Science course at a major 4-year Asian university. Half of the students were randomly assigned to a control group who used a scaled-down version of the Socratic Tutor system that only presents JAVA code examples and asks the participant to predict the output without providing any feedback or Socratic tutoring. The other half of the participants were assigned to a condition in which they used the Socratic Tutor. The Socratic Tutor asks to explain the code while trying to understand it and then predicts the output. After that, the tutor asks questions about the programming concepts used in the code. If a participant's answer is not correct or incomplete, the tutor initiates the three-level feedback mechanism.

3.1 Materials

Materials for this experiment included a self-confidence survey and a pre- and post-test. The self-confidence survey contained six questions with a 1–7 Likert scale where each question related to one programming concept. The pre- and post-test have similar levels of difficulty and contained 6 JAVA programs where each question assessed the student's understanding of a particular programming concept. For each question in the pre- and post-test, the participants were asked to predict the output of the code example.

3.2 Procedure

The experiment was conducted in a computer lab under supervision. First, participants were debriefed about the purpose of the experiment and were given a consent form. Those who consented took a self-confidence survey and the pre-test. Once they had finished the pre-test, an approximately 60-min tutoring session started. Finally, participants took the post-test and a post self-confidence survey.

3.3 Assessment

The pre and post-test questions were scored 1 when the student answer was correct and 0 otherwise. The learning gain score (LG) was calculated for each participant as follows [11].

$$LG = \begin{cases} \frac{post\text{-}test - pre\text{-}test}{6 - pre\text{-}test} & post\text{-}test > pre\text{-}test \\ \frac{post\text{-}test - pre\text{-}test}{pre\text{-}test} & post\text{-}test < pre\text{-}test \\ drop & pre\text{-}test = post\text{-}test = 6 \ or \ 0 \\ 0 & post\text{-}test = pretest \end{cases} \quad (1)$$

4 Results

4.1 Quantitative Analysis

Out of 70 participants, we dropped three participants from the treatment group and two from the control group because they had a perfect score in both tests and four students from the control group were dropped for not completing the experiment.

Table 1. Mean and Stander Deviation of Pre-test, post-test and Learning gain for control and treatment group

Section	n	Pre-test		Post-test		Learning gain (1)	
		Mean	SD	Mean	SD	Mean	SD
Control group	29	3.46	2.1	3.68	2	12%	9.1
Treatment group	32	3.47	1.8	4.8	1.3	57%	41

To understand how much do students learn when using Socratic Tutor, we analyzed the results from both groups in terms of average for pre-test, post-test, and learning gains as shown in Table 1. The results indicate that the learning gain of the treatment group was 45% higher and the results from a two-tailed t-test showed that there is a statistically significant difference between the two groups in learning gain scores ($t = 3.6$, $df = 51$, $p < 0.05$).

We analyzed the pre-post test improvement for each programming concept to understand how much do students learn on each programming concept when using Socratic Tutor. The results show between 10% and 33% higher improvement in treatment group, and there are a statistically significant differences in nested if-else ($t = -2.04$, $df = 56$, $p < 0.5$) and for loops ($t = -1.97$, $df = 54$, $p < 0.5$) concepts.

To understand how much does the Socratic Tutor affect students' self-confidence, we evaluated the pre-confidence and post-confidence scores. The results show that the treatment group participants improved their confidence level on average by 13% compared with −1.6% negative improvement in the control group. The result from an independent-sample t-test shows that the difference is statistically significant ($t = -3.1$, $df = 58$, $p < 0.05$).

To understand the relationship between the feedback and learning gains, we analyzed the relationship between the number of feedback each student received and his/her learning gains. The result shows that students received on average 15.4 feedback per tutoring session with a standard deviation of $SD = 7.1$. The relationship was investigated using the Pearson product-moment correlation coefficient. We found a strong, positive correlation between the number of feedback and learning gains ($r = 0.68$, $n = 32$, $p < 0.05$).

5 Conclusion

To understand the effectiveness of the Socratic Tutor ITS, we conducted a comparative study on seventy students who enrolled in Introductory to Computer Science course.

The seventy students were divided into two groups (1) control group where students have to read code and predict the output without any feedback from the system, and (2) treatment group where students interact with the Socratic Tutor.

The result shows that students who used Socratic-ITS improved their knowledge by 45% in term of learning gain, developed better understanding on concepts such as nested if-else and for loop, and improved their confidence level by 13%. Furthermore, the result of the Pearson product-moment correlation coefficient shows a positive correlation ($r = 0.68$) between feedback and learning gain.

Acknowledgments. This work as partially funded by the National Science Foundation under the award #1822816 (Collaborative Research: CSEdPad: Investigating and Scaffolding Students' Mental Models during Computer Programming Tasks to Improve Learning, Engagement, and Retention) to Dr. Vasile Rus. All opinions stated or implied are solely the authors' and do not reflect the opinions of the funding agency.

References

1. Anderson, J.R., Skwarecki, E.: The automated tutoring of introductory computer programming. Commun. ACM **29**(9), 842–849 (1986)
2. Banjade, R., et al.: Nerosim: a system for measuring and interpreting semantic textual similarity. In: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), pp. 164–171 (2015)
3. Beaubouef, T., Mason, J.: Why the high attrition rate for computer science students: some thoughts and observations. ACM SIGCSE Bull. **37**(2), 103–106 (2005)
4. Chang, K.E., Sung, Y.T., Wang, K.Y., Dai, C.Y.: Web/spl I.bar/soc: a socratic-dialectic-based collaborative tutoring system on the world wide web. IEEE Trans. Educ. **46**(1), 69–78 (2003)
5. Chi, M.T., De Leeuw, N., Chiu, M.H., LaVancher, C.: Eliciting self-explanations improves understanding. Cogn. Sci. **18**(3), 439–477 (1994)
6. Dadic, T., Stankov, S., Rosic, M.: Prototype model of tutoring system for programming. In: 28th International Conference on Information Technology Interfaces, pp. 41–46. IEEE (2006)
7. Danielson, R.L., Nievergelt, J.: An automatic tutor for introductory programming students (1974)
8. Johnson, W.L.: Understanding and debugging novice programs. Artif. Intell. **42**(1), 51–97 (1990)
9. Johnson, W.L., Soloway, E.: Proust: knowledge-based program understanding. IEEE Trans. Softw. Eng. **3**, 267–275 (1985)
10. Lane, H.C., VanLehn, K.: A dialogue-based tutoring system for beginning programming. In: FLAIRS Conference, pp. 449–454 (2004)
11. Marx, J.D., Cummings, K.: Normalized change. Am. J. Phys. **75**(1), 87–91 (2007)
12. Mcgettrick, A., Boyle, R., Ibbett, R., Lloyd, J., Lovegrove, G., Mander, K.: Grand challenges in computing: education-a summary. Comput. J. **48**(1), 42–48 (2005)
13. Mory, E.H.: Feedback research revisited. Handbook Res. Educ. Commun. Technol. **2**, 745–783 (2004)
14. Pillay, N.: Developing intelligent programming tutors for novice programmers. ACM SIGCSE Bull. **35**(2), 78–82 (2003)
15. Proulx, V.K.: Programming patterns and design patterns in the introductory computer science course. ACM SIGCSE Bull. **32**(1), 80–84 (2000)
16. Robins, A., Rountree, J., Rountree, N.: Learning and teaching programming: a review and discussion. Comput. Sci. Educ. **13**(2), 137–172 (2003)
17. Soloway, E.M., Woolf, B., Rubin, E., Barth, P.: Meno-II: An intelligent tutoring system for novice programmers. In: Proceedings of the 7th International Joint Conference on Artificial Intelligence, vol. 2, pp. 975–977. Morgan Kaufmann Publishers Inc. (1981)
18. Woods, P.J., Warren, J.R.: Rapid prototyping of an intelligent tutorial system. In: Proceedings of 12th Australian Society for Computers in Learning in Tertiary Education, pp. 557–563 (1995)