Online Programming Semantic Error Feedback using Dynamic Template Matching

Razali M.K.A, S. Suhailan, Mohamed M.A, M.D. M. Sufian
Faculty of Informatics and Computing
Universiti Sultan Zainal Abidin (UniSZA)
Terengganu, Malaysia

***Abstract*—Many of automated computer programming feedback is generated based on static template matching that need to be provided by the experts. This research is focusing on developing an automated online programming semantic error feedback by using dynamic template matching models based on students' correct answers submission. Currently, there is a lack of research using dynamic template matching model due to their complexity and varies in terms of programming structure. To solve the formulation of the dynamic templates, a new automated feedback model using front and rear n-gram sequence as the matching technique was developed to provide feedback to students based on the missing structure of the best-matched template. We have tested 60 student's Java programming answers on 3 different types of programming questions using all the dynamic templates randomly chosen for each student. An expert was assigned to manually match the student's answer with the 3 randomly chosen templates. The result shows that 80% of the best-matched templates for each student using the technique were similarly chosen by the expert. Based on the matched template, the student will be given feedback notifying the possible next programming instruction that can be included in the answer to get it correct as was achieved by the template. This model can contribute to automatically assist students in answering computational programming exercises.**

***Keywords*—Dynamic; feedback; online programming; semantic error; template matching**

I. Introduction

Computer science is a discipline that involves the understanding and design of computers and computational processes, including their theory, analysis, software and hardware design, efficiency, implementation, and application and effect on society [1]. In other words, computer science is an emergent, scientific and practical method, which deals with the theoretical basis of information and computation, and combines its realization and application technology [2]. Computer programming is one of the core subjects that every computer science student must be competent to become a good programmer. Therefore, to obtain the programming skill, lots of programming exercises need to be completed [3].

Students need to develop programming logic and thinking skills to understand and solve the tasks especially on code writing. Students also need to solve any encountered programming errors in their coding regarding the syntax, semantic, and also question requirements. From there, students will get the knowledge and experiences on how to encounter any common programming errors or mistakes. Learning through practice is the best way to learn computer programming and attract novice students [4].

Unfortunately, most computer science students face difficulties in learning computer programming especially in writing the programming scripts [5]. Despite the importance of computer science, there is a high percentage of failures and dropout rates in introductory programming courses recorded by most educational institutions around the world [2]. Lecturers must also be responsible for assisting and providing some feedback to their students to resolve students' misunderstandings or mistakes. Helping a large number of students in providing personalized feedback during programming exercises will be a difficult role for teachers [6].

Furthermore, there are a lot of automated programming assessment tools with automated feedback that have been continuously developed to help students practice programming and build up logic skills and also programming syntax [7]. By using any automated programming tools, a student can submit a computer program on a problem-solving exercise while the tool will promptly produce automated feedback to highlight any encountered errors or mistakes during the compilation or implementation of the program [8]. The error is produced by the compiler known as Syntax Error. The compiler will highlight which lines that contain errors. However, for a beginner student, the syntax error j does not explain on how to fix the code in solving the question problem. This research is focusing on developing an online programming semantic error feedback by using a dynamic template matching model.

II. Related Work

The teacher-student ratio can reach thousands to one by implementing the advancement of Massive Open Online Courses (MOOCs) [14]. This makes the feedback design more specific and personalized. Unfortunately, providing manual teacher feedback for programming assignments is determined as a traditional method and it is no longer suitable for MOOCs. Current automatic feedback methods have some weaknesses, such as the inability to extend to larger programs, manual teacher involvement, and lack of accuracy in determining errors.

There are two techniques to design the programming feedback; static and dynamic approaches. Static approaches identify and study the source code without running the computer program [9]. It is used to evaluate the syntax and semantic error and programming style. The dynamic approach

is based on the execution of the computer program [10]. It is used to evaluate run-time errors, programming design, and software metrics such as timing and resources utilization.

For beginners, static feedback is crucially needed in helping them to visualize the logic of the computer program in solving a question. As to master the programming skills, lots of exercises need to be completed by a student. With the advance of an e-learning platform, many platforms offer programming exercises to be done online. There are a lot of programming tools where users can learn and train their programming skills by solving the given problem with some programming code to find the best solution for that problem [11]. Most of these tools are developed as web applications. Some of these tools are CodingBat [12], betterprogrammer, Practice-It, and CodeWorkout [13]. By using these systems, users can get feedback about their submitted answers because these systems already provide a set of practical programming problems to be solved in the web browser and the results are evaluated by checking them against unit tests or test cases. Unfortunately, this dynamic feedback is difficult to be understood by the beginner who wants to start learning the logic or flow of the programming. The feedback is general in highlighting how the output should be generated. Writing hints and preparing the feedback in this way needs meta cognition and involves critical thinking which is not yet developed among the beginners.

The novice programmer tries to imitate the steps prepared by the teacher, and some errors that the novice programmer could not solve appeared during compilation [15]. One of the challenges in writing coding for novice programmers is insufficient feedback error messages. The only feedback that is available is the compiler-based error on the syntax errors [16]. Therefore, the best compiler errors are those that can deliver important messages that are desperately needed by programmers in response to all the errors they make. Decaf is a Java editor that serves as a medium for improving javac compiler error messages. An error message will be produced by the compiler if there are some errors contains in the student's source code. Then, the error codes and error messages are analyzed to produce enhanced error messages that provide more valuable information to students, in the hope that the error can be corrected more effectively as compared to the ordinary error messages alone.

With the existence of the standard error and enhanced errors, students can avoid making the same error in the future by referring to the both types of error Decaf is an enhanced compiler error message as shown in Fig. 1 that elaborate the common syntax error produced by Java.

However Decaf only provide feedback in clarifying the error related to the programming syntax. A logic error which is part of the semantic error is not presented in most of the compilers as it depends on the individual question requirements.

A semantic error feedback is meant to provide feedback based on specific question requirements using a solution template [17]. A template consists of a correct program instructions sequence (keywords, symbols, numbers). This computer programs need to be converted to certain features numbers before it can be processed as a matching template. [8] was using the instruction ratio (IGR) and the instruction count ratio (ICR) as the features to represent the computer program. IGR is the ratio of sequential instructions or symbol sequences in a program to instructions or symbol sequences of templates with some skippable instructions. Meanwhile, ICR is the average ratio of the amount of all unique instructions within the program that matches the amount of all unique instructions laid out in the template. Based on these features, the K-Means algorithm was used to assign similar computer programs to certain clusters. Based on each cluster, programs that have a similar Euclidean distance to the centroid in the cluster are grouped. These groups represent unique rules that will be associated with semantic feedback. Under this rule, an expert will add an assisted feedback to add comments of what further actions need to be done in solving the question.

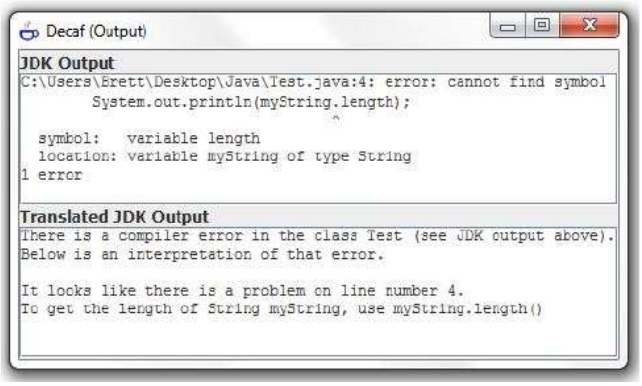

Fig. 1. Example of an enhanced Compiler Error Message Produced by Decaf.

However, the technique needs to design the enhanced feedback manually to make sure their student understands the way on how to fix the error in their code. It needs to be done on pre-defined templates. This method requires more resources from the teacher not only to prepare the template but also need to manually assign students' program clusters with feedback from time to time. This research further enhanced this technique by making the matching template more accurate by comparing forward and reverse sequences of the n-gram algorithm. It also provides automated feedback based on the missing instruction sequence based on the selected dynamic templates mining from the correct submission answers from other students.

III. Methodology

The N-gram model is improved by calculating the sequence N-gram in two different ways. The first approach is using front N-grams where the value is gained by calculating the matched sequences based on the forward parsing of the codes. The second approach is using rear N-grams where the value is gained by calculating matched sequences based on the reverse parsing of the codes.

The combination of front and rear N-gram values which are referred to as the FR-Grams model are then used to enhance the similarity finding technique between two different programs. Fig. 2 shows the framework of generating the semantic error feedback using FR-Grams.

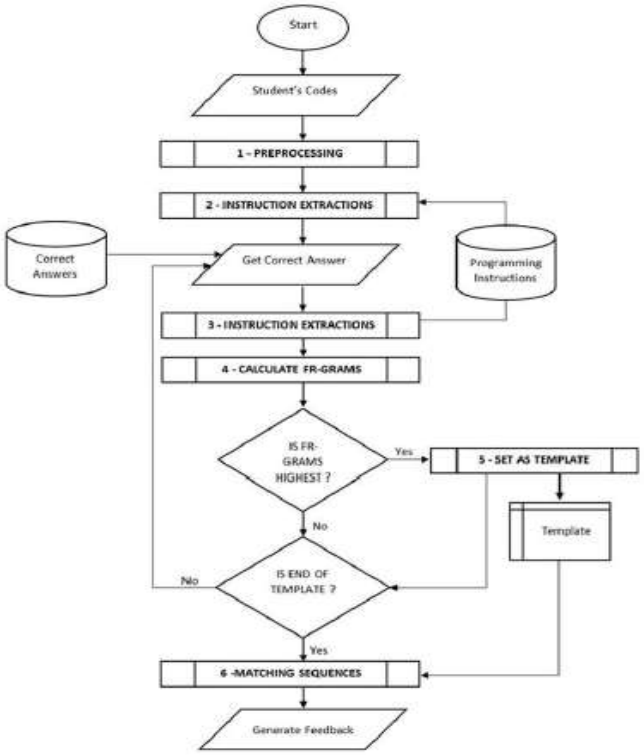

Fig. 2. Semantic Error Feedback Framework using FR-Grams Technique.

FR-Grams Model is then used to match a student's program with the dynamic program templates. The templates were auto-generated based on the list of correct answers of the submitted student's program. The answers were graded automatically by executing the answers and matched with the expected test cases. Based on the best-matched program templates between the student's program, the student will get feedback notifying any missing program instruction sequence that needs to be added to the program in order to get the correct answers according to the selected template.

A. Step 1: Pre-processing

The library of computer instructions keywords needs to be prepared first before the computer program instruction's extraction can be generated. For example, a list of Java instructions or keywords are "int", "if", "string", "nextInt()", "public" and others. These keywords are inserted into a file to be used as a key point to filter the submitted answers. In the pre-processing process, any essential code before the program body needs to be removed. In java, these codes are something like "import java.util.Scanner", "public static void main". This framework only considers the main program body.

B. Step 2: Instruction Extractions from Student's Answer

Student programs need to be converted as a sequence of instructions. The sequence of Java instructions from a student example answer shown in Fig. 3 will be produced as the following: -

"Scanner", "System", "String", "next()", "int"

```
1  import java.util.Scanner;
2  public class Q4 {
3    public static void main(String[] args) {
4      Scanner k = new Scanner(System.in);
5      String name = k.next();
6      int age = k.next();
7      int balance = 50-age;
8      System.out.println("Hi, " + name +"! Your age is "+ i +".");
9    }
10 }
```

Fig. 3. Sample of a Student's Program.

Then, the total instruction (N) is calculated based on the sequence of the program instructions.

C. Step 3: Instruction Extractions from Template

Templates are the collection of successful and accepted computer program submissions made by the previous student attempts to a question. These templates need to be converted into a sequence of instructions similar to the Step 1 to 2.

D. Step 4: Calculate FR-Grams

The number of FR-grams from student answers and correct answers is compared to calculate the sequence of programming. The algorithm to calculate N-grams is given in Fig. 4.

E. Step 5: Find the Best Template

All the correct students' answer stored in the database will be selected as the dynamic template matching. A student's attempt answer will be matched with these templates. The highest total FR-grams among them will be the considered as the most accurate template for further feedback generation.

F. Step 6: Feedback Generation

After the comparison, the total FR-grams value is produced based on the template selected in step 4. After finishing comparing the answers, the FR-Grams are calculated to get the total FR-grams for each template. The highest total will be processed as the feedback template.

```
1:   N = number of unique instructions (I') in the template
2:   NGRAM = 0
3:   for i=1 to N do
4:       SA = Student answer
5:       CA = Correct answer
6:       if SA == CA then
7:           NGRAM++
8:       else
9:           N++
10:      end if
11:      N++
12:  end for
13:  return NGRAM
```

Fig. 4. Algorithm to Calculate N-grams.

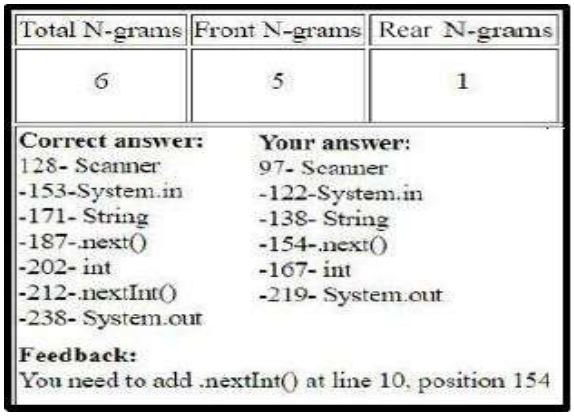

Fig. 5. Automated Feedback based on Template Matching.

Fig. 5 shows that the student's attempt consists of instructions sequence of "Scanner", "System.in", "String", ".next()", "int" and "System.out". While comparing to the template, a feedback will be generated by the system notifying that "You need to add .nextInt() at line 10, position 154". This is the missing instruction sequence that the student needs to add for the program to be tailored to the template. This feedback can provide some clues for the student on how to proceed and make the correction to the program.

IV. Result and Analysis

There were 60 student's Java programming answers were tested using all the dynamic templates. The answers were based on 3 different set of programming questions. The templates were randomly chosen for each student's attempt by the system. Table I is the sample feedback that was generated by the system along with the template chosen by the system.

An expert was assigned to manually match the student's answer with the three randomly chosen templates. The experts are chosen based on their experience in validating the source code and marking the student's programming answer. For each student's answer, the expert will be presented with the three template that have highest total FR-grams to be compared with the student's answer. The expert was provided with a rubric in order to choose which template should the student's attempt be referred most.

1) Check for similar variables: In a student's answer, many variables contain in the source code such as "string", "int", "char", "for" and others. If the student's answer contains a "string" variable, the experts will search this variable in the correct answers to get the most accurate template.

2) Check for quantity and type of variable: If the student's answer contains a "string" variable but in the template use "string[]" which is a string array type, the template will not be chosen.

3) Check for the simpler with the student's answer: If each template has passed the first and second rules which means that there is almost similarity between the templates, the experts will choose the simplest template according to the student's answer.

TABLE I. Sample Feedback by System

Student ID	Answer	Best Feedback	Result

01	import java.util.Scanner;
	public class Q2 {
	public static void main(String[] args) {
	Scanner k = new Scanner(System.in);
	String a="Apology";
	char [] b=new char[7];
	int i=a.length();
	System.out.print("*");
	System.out.print(a.charAt(1)+"*****");
	}
	}
		You need to
		add next()
		at line 5,
		position 43
			True

The result shows that there were 48 out of 60 or 80% similar decision made by the model and the expert. 9 from the 12 answers that were not matched with the expert's decision was due to the small difference of the total FR-Grams (only one missing sequence different) among the templates. On the other hand, these cases will also contribute difficulty for the expert to decide which template should be considered as the most matched. Based on the matched template, the student will be given feedback notifying the possible next programming instruction that can be included in the answer to get it correct as was achieved by the template. This will be like personal coaching to help students recover from any cluelessness on the programming command sequences to answer computational programming exercises.

However, there was a weakness for this sequence-based model as it does not recognize the template based on the data type usage. For example, answer that was using array data type should have only seek template that using the same data type. This will be the future research works need to be conducted in identifying template context in order to get the best template matching for a more accurate feedback.

V. Conclusion

In conclusion, the best semantic error feedback for the student should meet these criteria:

1) Can guide the student on what is missing.

2) Can highlight to the student what student needs to include in the source code to fix the error.

With this semantic error feedback, students can get a valuable idea or hint to solve the error in their source code. The critical thinking skills of students will increase based on computational logic skills practice tools. The system continuously collects feedback as a repository which eventually fully automated interactive assisted learning system can be achieved. Lastly, the student will keep interested and motivated to self-practice programming exercises towards problem-solving skill development.

Acknowledgment

Special thanks to the Ministry of Higher Education Malaysia and Universiti Sultan Zainal Abidin (UniSZA) for providing equipment and supporting this research project under the grant number of UNISZA/2018/GOT/03.

REFERENCES

[1] Tucker, A. (Ed.). (2006). A model curriculum for K-12 computer science: Final report of the ACM K-12 task force curriculum committee (2nd ed.). New York: Association for Computing Machinery (ACM).

[2] Queiros, Ricardo. (2014). Innovative teaching strategies and new learning paradigms in computer programming. 10.4018/978-1-4666-7304-5.

[3] Kwiatkowska, M., 2016. Measuring the Difficulty of Test Items in Computing Science Education. In: Proceedings of the 21st Western Canadian Conference on Computing Education, BC, Canada, 6 - 7 May 2016. ACM Press.

[4] Gross, P., & Powers, K. (2015). Evaluating assessments of novice programming environments. In Proceedings of the First International Workshop on Computing Education Research (pp. 99-110). New York: ACM. doi:10.1145/1089786.1089796.

[5] Anthony Robins, Janet Rountree and Nathan Rountree (2003). Learning and Teaching Programming: A Review and Discussion. Computer Science Education, Vol. 13, No. 2, pp. 137–172.

[6] S. Suhailan, M.K. Yusof, A.F.A. Abidin, S.A. Fadzli, M.S. Mat Deris and S. Abdul Samad (2018). Automated Ranking Assessment based on Completeness and Correctness of a Computer Program Solution. International Journal of Engineering & Technology, 7 (3.28) (2018) 278-283.

[7] S. Suhailan, S. Abdul Samad, M.A. Berhannuddin (2015). A perspective of Automated programming error feedback Approaches in problem solving exercises. Journal of Theoretical and Applied Information Technology. 70(1) 121-129.

[8] S. Suhailan, M.S. Mat Deris, S. Abdul Samad, M.A. Burhanuddin (2019). A Recommended Feedback Model of a Programming Exercise Using Clustering-Based Group Assistance. International Journal of Recent Technology and Engineering (IJRTE), ISSN: 2277-3878, Volume-7, Issue-5S4.

[9] Ala-Mutka, K. M. (2005). A survey of automated assessment approaches for programming assignments. Computer science education, 15(2), 83-102.

[10] Adidah Lajis, Shahidatul Arfah Baharudin, Diyana Ab Kadir, Nadilah Mohd Ralim, Haidawati Mohd Nasir and Normaziah Abdul Aziz (2018). A Review of Techniques in Automatic Programming Assessment for Practical Skill Test.

[11] Ashlesha Patil (2010). Automatic Grading of Programming Assignments.

[12] Priyanka Mohan (2015). Student Perceptions of Various Hint Features while Solving Coding Exercises.

[13] Kevin Buffardi and Stephen H. Edwards (2014). Adaptive and Social Mechanisms for Automated Improvement of eLearning Materials.

[14] Ke Wang, Benjamin Lin, Bjorn Rettig, Paul Pardi, and Rishabh Singh (2017). Data-Driven Feedback Generator for Online Programing Courses.

[15] Aniket Bhawkar, Rohit Belsare, Fenil Gandhi and Pratiksha Somani (2013). Analysis of Errors - A Support System for Teachers to Analyze the Error Occurring to a Novice Programmer.

[16] Brett A. Becker (2016). An Effective Approach to Enhancing Compiler Error Messages.

[17] S. Suhailan, S. Abdul Samad, M.A. Berhannuddin. Nazirah (2017). Program Statement Parser for Computational Programming Feedback. Journal of Engineering and Applied Sciences, 12(5S) 7057-7062.