

Students' Perceptions and Preferences of Generative Artificial Intelligence Feedback for Programming

Zhengdong Zhang^{*2}, Zihan Dong^{*1}, Yang Shi¹, Thomas Price¹, Noboru Matsuda¹, Dongkuan Xu¹

¹ North Carolina State University

² Georgia Institute of Technology

zzhang3135@gatech.edu, {zdong7, yshi26, twprice, nmatsud, dxu27}@ncsu.edu

Abstract

The rapid evolution of artificial intelligence (AI), specifically large language models (LLMs), has opened opportunities for various educational applications. This paper explored the feasibility of utilizing ChatGPT, one of the most popular LLMs, for automating feedback for Java programming assignments in an introductory computer science (CS1) class. Specifically, this study focused on three questions: 1) To what extent do students view LLM-generated feedback as formative? 2) How do students see the comparative affordances of feedback prompts that include their code, vs. those that exclude it? 3) What enhancements do students suggest for improving LLM-generated feedback? To address these questions, we generated automated feedback using the ChatGPT API for four lab assignments in a CS1 class. The survey results revealed that students perceived the feedback as aligning well with formative feedback guidelines established by Shute. Additionally, students showed a clear preference for feedback generated by including the students' code as part of the LLM prompt, and our thematic study indicated that the preference was mainly attributed to the specificity, clarity, and corrective nature of the feedback. Moreover, this study found that students generally expected specific and corrective feedback with sufficient code examples, but had diverged opinions on the tone of the feedback. This study demonstrated that ChatGPT could generate Java programming assignment feedback that students perceived as formative. It also offered insights into the specific improvements that would make the ChatGPT-generated feedback useful for students.

Introduction

The rapid recent evolution of artificial intelligence (AI) technology makes the integration of AI into the education sector promising (Chen, Chen, and Lin 2020). Simple AI tools are already widely utilized in education to automate relatively mundane and repetitive tasks (Kathe Pelletier 2023). As AI technology continues to advance in complexity and capability, there is a growing potential to utilize it in more intricate educational tasks. Particularly, Large Language Models (LLMs), such as OpenAI's ChatGPT, have strong performance on various natural language processing

tasks (Chang et al. 2023). They present a compelling opportunity for automating a series of natural language-based educational tasks. Specifically, this paper evaluated LLM-generated feedback for programming assignments in an undergraduate Introduction to Java course (CS1) by delving into students' perceptions, using formative feedback guidelines (Shute 2008) and questions about their preferences of the LLM-generated feedback.

In educational contexts, feedback can be categorized into summative and formative feedback. While summative feedback mainly functions as an evaluative tool at the end of an instructional period, formative feedback offers ongoing guidance aimed at enhancing learning and performance (Dixson and Worrell 2016). Feedback is a crucial component in students' learning process. Numerous studies across diverse academic disciplines consistently indicate that formative feedback is effective in facilitating student learning (Hao et al. 2019). While the importance of feedback is well-established, generating feedback for programming assignments is challenging since it requires substantial effort by teaching personnel (Gulwani, Radiček, and Zuleger 2018). As interest in computer science education continues to surge, the need for automated feedback systems tailored for programming assignments becomes increasingly critical.

Extensive research has been conducted on automated feedback for programming assignments. The state of practice in most classrooms includes evaluating the correctness (e.g. (Morris 2003; Cheang et al. 2003)), maintainability (Cordova et al. 2021), and readability (Liu and Petersen 2019) of students' code through dynamic analysis and static analysis (Messer et al. 2023). AI-based methods, mainly machine learning algorithms trained on various datasets, have also emerged as a complement to traditional methods for programming assignment feedback generation (Piech et al. 2015; Corbett and Anderson 1994; Shi et al. 2022, 2021).

While these feedback generation systems have been effective in many aspects, there is room for further enrichment in the area of natural language-based automated feedback. Prior work, such as JavaTutor (Wiggins et al. 2015), mostly relied on pre-defined templates for natural language-based feedback generation. The rise of LLMs opens up a new opportunity for generating more customized natural language-based feedback without pre-defined templates, as LLMs exhibit strong reasoning and natural language gen-

^{*}These authors contributed equally.

Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

eration abilities (Chang et al. 2023). Recent studies also demonstrated the ability of LLMs to solve introductory programming problems (Finnie-Ansley et al. 2022) and create new programming problems (Sarsa et al. 2022), suggesting LLMs’ potential for generating feedback specific to programming assignments. Given that LLM-generated feedback is an emerging area, there are many research questions that deserve exploration in order to understand how LLMs can be integrated into real-world educational settings, such as how LLM-generated feedback affects learning outcomes and how students perceive LLM-generated feedback.

In this paper, our research focus is on students’ perception of LLM-generated feedback. We aim to understand how students perceive the programming assignment feedback generated by ChatGPT, one of the most popular LLMs, in terms of formative qualities. We also seek to understand how students see the comparative affordances of feedback prompts that include their code, vs. those that exclude it since ChatGPT has the capability to offer in-depth suggestions when provided with code. However, there is also a risk of generating inaccurate advice if the code is misinterpreted. Lastly, we gathered students’ suggestions for how to further improve the ChatGPT-generated feedback. We conducted our experiments in a CS1 introductory Java course, with 58 students who mostly major in computer science, at a U.S. public university during the Summer of 2023. We designed a series of surveys based on the formative feedback guidelines proposed by Shute (Shute 2008). Then we generated automated feedback by ChatGPT API for four selected lab assignments in the Java course. After the grades of each lab assignment had been released, we distributed the feedback to students and collected students’ responses through the survey. We also performed a thematic analysis of the open-ended questions in the survey to understand the reasons for feedback preference and the areas for improvement suggested by students. In general, experiments in this paper answered the following three research questions (RQs):

RQ1. *To what extent do students view ChatGPT-generated feedback as formative?*

RQ2. *How do students see the comparative affordances of feedback prompts that include their code, vs. those that exclude it?*

RQ3. *What enhancements do students suggest for improving ChatGPT-generated feedback?*

Our study makes several contributions to the field of computer science education. Firstly, by evaluating the ChatGPT-generated feedback against formative feedback guidelines through surveys, we provide empirical evidence that ChatGPT-generated feedback aligns well with formative feedback guidelines established by Shute. Secondly, we find a clear preference among students for feedback generated from prompts that contain their code. Our thematic analysis suggests that this preference is largely driven by the perceived specificity, clarity, and corrective nature of the feedback. Thirdly, this study captures students’ suggestions for improving the ChatGPT-generated feedback. According to our thematic analysis, students commonly desire more spe-

cific and corrective feedback with sufficient code examples. Additionally, we uncover divergent preferences concerning the tone of the feedback; while some students appreciate an encouraging tone, others opt for more critical evaluations. This divergence underlines the need for personalization in automated feedback systems, offering a direction for future advancements in this area.

Related Work

Formative Feedback and Feedback Quality Evaluation.

Formative feedback is generally regarded as effective in educational contexts (Hao et al. 2019). According to Shute, it is defined as “information communicated to the learner that is intended to modify the learner’s thinking or behavior to improve learning” (Shute 2008). Prior research often evaluated the quality of formative feedback by measuring improvement in students’ learning outcomes (Hao et al. 2022; Irons and Elkington 2021). This approach, while objective, may not fully capture students’ own perceptions of the feedback or the multi-faceted nature of formative feedback as articulated by Shute’s formative feedback guidelines. To address this nuanced aspect, we implemented a survey-based methodology, involving students to evaluate the degree to which ChatGPT-generated feedback adheres to Shute’s formative feedback guidelines. This methodology enables us to understand students’ subjective evaluations of the feedback, thereby enriching the multi-dimensional assessment of formative feedback quality.

Automated Generation of Feedback for Programming Assignment.

Providing manual feedback for programming assignments is challenging as it requires substantial efforts by teaching personnel (Gulwani, Radiček, and Zuleger 2018), and the instructor-to-student ratio is often low for computer science courses (Camp et al. 2017). To mitigate this issue, initial efforts focused on automatic grading systems that offered binary feedback on the correctness of code submissions (Von Matt 1994). The advent of more advanced technologies has enabled the provision of richer and more nuanced feedback. For instance, Singh et al designed a rule-based system to provide feedback on how incorrect a given solution was (Singh, Gulwani, and Solar-Lezama 2013). Besides, some studies underline the importance of programming styles, developing rule-based tools to assess code style in students’ programming assignments (Ala-Mutka, Uimonen, and Jarvinen 2004). More recently, AI techniques, such as machine learning and deep learning, have been integrated into the feedback generation process. These technologies are used in intelligent tutoring systems for student modeling (VanLehn 1988) and undergo modifications to better suit the realm of programming education. For example, Bayesian knowledge tracing (Corbett and Anderson 1994), commonly used for tracking student performance in multiple-choice scenarios, was adapted by Shi et al. to be applicable for open-ended programming assignments (Shi et al. 2022). Despite these advancements, the potential of ChatGPT in programming assignment feedback generation remains largely unexplored. This paper aims to fill this gap by examining student perspectives on ChatGPT-generated feedback for

programming assignments.

Large Language Models for Education. The integration of large language models (LLMs), such as ChatGPT, into educational settings, has become an increasing research interest. Many studies were conducted to explore how LLMs can assist students’ learning process, such as generating quizzes (Dijkstra et al. 2022), algebra hints (Pardos and Bhandari 2023), and code explanations (MacNeil et al. 2022). Those studies demonstrate that LLMs have the potential to generate educational materials for students.

In a recent study, Maciej et al. explored the potential of using OpenAI’s GPT-3.5 model for automated hint generation in programming assignments (Pankiewicz and Baker 2023). They designed a controlled experiment to quantify the impact of AI-generated feedback on students’ performance and conducted a quick affect survey to gauge students’ emotional states during task completion. The study indicated that compared to the control group, the availability of the AI-generated hints improved students’ performance but didn’t significantly impact students’ affective state. Different from their experiment that involves students’ performance, we focus on understanding students’ subjective evaluation and preference of the ChatGPT-generated feedback and soliciting their suggestions for future improvements.

Concurrent with the advancements in the educational applications of LLMs, there is a growing concern about the ethical implications of their integration. For instance, Qureshi et al. discuss the possibility of cheating and a shallow understanding of the course material when students are provided with direct access to ChatGPT (Qureshi 2023). These concerns are not isolated. Educational institutions in New York State have banned the use of LLMs due to fears over academic integrity and plagiarism (Elsen-Rooney 2023). While the debate over the ethical use of LLMs in education continues, the academic community has started to form guidelines and policies for the use of these tools. Researchers and educational institutions begin to regard LLMs not just as potential sources of academic misconduct but also as useful teaching aids (Abramson 2023; Rose 2023).

Experimental Design

Our IRB-approved experiments were conducted in the Summer of 2023 in an introductory Java programming course (CS1) at a U.S. public university. A total of 58 students consented to participate in the study and completed one or more of our four surveys. The number of respondents for each survey ranged from 23 to 28 students. Our study focused on four specific lab assignments—lab 6, lab 9, lab 12, and lab 14—that were designed to evaluate fundamental Java skills, such as string manipulation, arithmetic calculations, loops, and file processing (see Figure 1 for an overview of our experimental process). These labs were strategically selected to offer a balanced representation of both the course timeline and the varying complexities of the assignments.

All students in the class would receive two pieces of ChatGPT-generated feedback within 1 to 2 days after the professor published the final grades for each of the four selected lab assignments. These pieces of feedback aimed

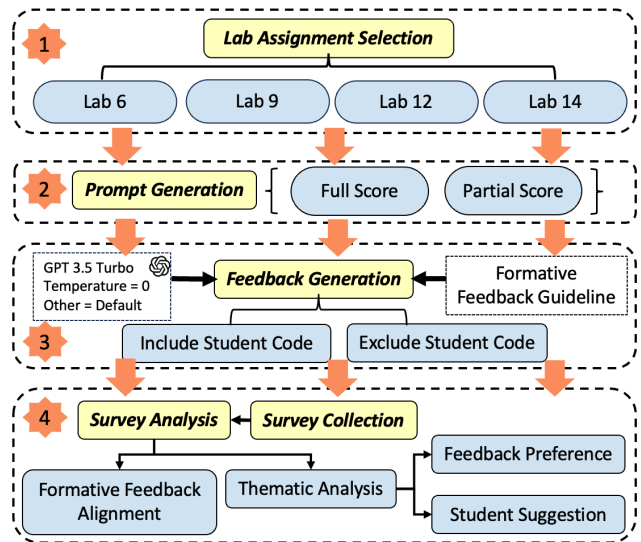


Figure 1: Experimental Design: The Experiment is divided into 4 sequential phases, ①-④. In ①, we selected 4 lab assignments covering critical topics in the Java course. In ②, we categorized students into groups based on their grades to tailor the prompts accordingly. In ③, Prompts were processed through ChatGPT to generate feedback. And in ④, we collected and analyzed survey results.

to assist students in improving their grades by evaluating their failed test cases, code, and the GitHub feedback, which was automatically generated by the grading system with each push to GitHub. The first piece of feedback includes students’ code in the prompt while the second did not. This design allowed us to explore the impact of including students’ code in the prompt on students’ preferences towards the ChatGPT-generated feedback. After receiving the ChatGPT-generated feedback, students were requested to complete a survey within one week. These surveys were designed to capture students’ experiences, satisfaction levels, and suggestions for improvements. This assignment-feedback-survey procedure was repeated for all four lab assignments to study the extent of the formative nature, students’ preference for prompts, and potential improvement of ChatGPT in Java assignment feedback generation. The procedures for feedback generation and survey generation are elaborated in the sections below.

Feedback Generation

A pivotal part of our experimental design involves the generation of feedback for students’ assignments using ChatGPT API. This process includes the following key considerations.

Message Roles: The ChatGPT API allows users to send messages to ChatGPT by two roles: “system” and “user” (J. 2023). Messages tagged with the “system” role serve as predefined instructions that guide ChatGPT’s behavior. They are referred to as system prompts in this paper. Messages tagged with the “user” role represent the input from users interacting with the ChatGPT and are termed as user prompts in this paper. Student-specific information such as code and test cases was entered into ChatGPT through the “user” role.

Role	Your name is AI Assistant. You are a teaching assistant to provide formative feedback to students' Java homework (<i>Lab Name</i>).	
Purpose	For students with full score: The student achieved a full score in the homework. Your goal is to encourage the student to keep up the great work and excel in future homework.	For students with partial score: The student didn't achieve a full score in the homework. Your goal is to help the student correct mistakes and excel in future homework.
Requirements	Requirements about the feedback you provide: 1. The feedback must be polite, encouraging and second person writing. 2. The feedback must be no more than 150 words. 3. The feedback must have clear structure for readability. 4. The feedback must greet the student. 5. The feedback must not contain false information. 6. Do not provide feedback if you don't have valuable feedback 7. Avoid using words like but and however in your feedback.	
Lab Results	[Failed Test Case Start] Test Case for <code>StringArrays.equals2D()</code> : Input: "Cat", "Dog", "Puppy", "A", "B", "C" Expected Output: "True" Student Output: "false" [Failed Test End] [Student's Code Start] Java Code written by the student [Student's Code End] [Github Feedback Start] Git Commit: x1234 Sun Jul 2 23:12:15 EDT 2023 .gitignore NOT included [Github Feedback End]	

Figure 2: Prompts: In the Role section, ChatGPT was designated as a teaching assistant to provide feedback. The Purpose section categorized students by full or partial scores and tailored prompts for customized feedback. The Requirement section outlined basic criteria for generating feedback, and the Lab Results section included specific student conditions as prompts for feedback.

Temperature Setting: Temperature is a parameter that controls the randomness of ChatGPT's outputs. Lower temperature values make the output more deterministic, while higher values introduce more randomness (OpenAI 2023a). To avoid unnecessary randomness in responses, the temperature was set to 0 for feedback generation in this paper.

Token Restrictions: OpenAI imposes token limits on ChatGPT, which can be challenging when generating feedback for assignments with extensive code. Although our experiments were unaffected by this constraint, we acknowledged it as a potential bottleneck for real-world applications. To assess the impact of including students' code in user prompts on their preferences, we generated two feedback versions for students. One included their code in the prompt, while the other didn't. This approach helped us understand how code inclusion affected perceived feedback usefulness (OpenAI 2023b).

Prompts: In our experiments, prompts were divided into system prompts and user prompts as defined in the Message Roles section. System prompts varied based on students' grading outcomes. For students receiving full scores, the system prompt required ChatGPT to generate words of encouragement and a summary of the student's performance. For students who did not receive a full score, the system prompt required ChatGPT to give tailored feedback highlighting mistakes in the code and how the student could improve. The user prompt was composed of the following com-

ponents, separated by square brackets indicating the beginning and end of each component (See Figure 2):

- **Failed Test Case:** Details of test cases that students failed, comprising the name of the function these test cases are designed for, input and expected output of these test cases, and the actual output from students.
- **Students' Code** (only included in the version of prompt that contained code): The code submitted by students for an assignment.
- **GitHub Feedback:** the automated feedback generated by the grading system covering build time, Checkstyle errors, commit info, code/test compilation status, presence of `.gitignore` or `*.class` files, and project structure validity.

Survey Design

To gain insight into students' perceptions and preferences of the ChatGPT-generated feedback, we designed surveys for the selected four lab assignments. Besides the survey agreements, each survey consisted of 10 questions. The first question verified students' performance on the assignment. The next two explored students' preferences regarding the two types of feedback provided. Continuing with six questions, we evaluated students' perceptions of how well the selected feedback aligns with the formative feedback guidelines outlined by Shute (Shute 2008) for the feedback they received. The last question solicited open-ended comments for future improvement. Shute divided the formative feedback guidelines into four categories: things to do, things to avoid, timing considerations, and learner characteristics. In our study, we concentrated on the first category, as things to avoid, timing considerations, and learner characteristics are not directly relevant to ChatGPT's inherent ability to generate feedback (e.g., one could present ChatGPT-generated feedback immediately, or after a delay). For guidelines within the "things to do" category, we designed survey questions specifically for those requiring students' subjective input. For instance, for the guideline "*Present elaborated feedback in manageable units*", the survey question is "*Was the feedback presented in small and manageable pieces to avoid overwhelming you?*", which gauges students' perspectives of this guideline. The survey questions and associated guidelines can be found in Appendix A of the arXiv version.

Data Processing

We gathered a range of data to facilitate feedback generation for student assignments. This included students' names which were anonymous by index, code, grades, GitHub feedback, and details of failed test cases. Students' code and GitHub feedback were collected from GitHub/GradeScope using their respective APIs. Information regarding students' grades and repositories was retrieved from the course grading system. Test cases for each assignment were provided by the course lecturer. Utilizing these pieces of data, we ran a specialized Autograder to detect failed test cases for each student. To avoid feeding too many failed test cases to ChatGPT, we categorized failed test cases based on the programming functionality for which the test cases were designed. For students with more than 4 failed test cases, we took only

Question #	Response	Lab Assignment				Grand Total
		lab 6	lab 9	lab 12	lab 14	
Be specific and clear with feedback message.	1 (Too ambiguous)			1 (4%)		1 (1%)
	2	4 (14%)	1 (4%)	1 (4%)	1 (4%)	7 (7%)
	3	7 (25%)	8 (30%)	4 (17%)	2 (9%)	21 (21%)
	4	7 (25%)	12 (44%)	9 (38%)	4 (17%)	32 (31%)
	5 (Clear enough)	10 (36%)	6 (22%)	9 (38%)	16 (70%)	41 (40%)
Give unbiased, objective feedback, written or via computer.	No				2 (9%)	2 (2%)
	Yes	28 (100%)	27 (100%)	24 (100%)	21 (91%)	100 (98%)
Keep feedback as simple as possible but no simpler.	Simple enough	23 (82%)	24 (89%)	20 (83%)	19 (83%)	86 (84%)
	Somewhat too complex		1 (4%)		1 (4%)	2 (2%)
	Somewhat too simple	4 (14%)	1 (4%)	1 (4%)	2 (9%)	8 (8%)
	Too complex	1 (4%)		1 (4%)		2 (2%)
	Too simple		1 (4%)	2 (8%)	1 (4%)	4 (4%)
Present elaborated feedback in manageable units.	FALSE	1 (4%)	2 (7%)	1 (4%)	1 (4%)	5 (5%)
	TRUE	27 (96%)	25 (93%)	23 (96%)	22 (96%)	97 (95%)
Promote a "learning" goal orientation via feedback.	1 (Not at all)		1 (4%)			1 (1%)
	2	3 (11%)	2 (7%)	2 (8%)	2 (9%)	9 (9%)
	3	9 (32%)	10 (37%)	6 (25%)	3 (13%)	28 (27%)
	4	14 (50%)	11 (41%)	13 (54%)	14 (61%)	52 (51%)
	5 (Extremely)	2 (7%)	3 (11%)	3 (13%)	4 (17%)	12 (12%)
Reduce uncertainty between performance and goals.	No	5 (18%)	1 (4%)	1 (4%)		7 (7%)
	Yes	23 (82%)	26 (96%)	23 (96%)	23 (100%)	95 (93%)

Figure 3: Survey Questions for Formative Feedback Guidelines

one failed test case from each test case category (if available) and put it into the user prompt. After that, we generated students' outputs for each of these selected failed test cases. The inputs and expected outputs for these failed test cases were extracted from the test case scripts provided by the lecturer. Based on the processed data, we created both system and user prompts for each student. Finally, we leveraged the ChatGPT API to produce tailored feedback.

Qualitative Analysis

We conducted an in-depth analysis of students' responses to the two open-ended questions in the survey. The analysis involved two key components: categorizing survey submissions based on pre-defined properties and processing open-ended responses through thematic analysis. In the first step, survey submissions were categorized based on conditions including respondents' grades (either a full or partial score) and their preference for the two pieces of feedback. For the thematic analysis (Skripchuk, Shi, and Price 2022) in the second step, three authors independently generated initial code to identify key elements in the students' responses, focusing on the reasons behind their feedback preference and suggestions for future improvements. Following this, the researchers collaboratively reviewed and reconciled these initial code to identify overarching themes. These themes were then meticulously refined and explicitly defined. See Appendix B of the arXiv version for detailed definitions.

Results

In this section, we thoroughly analyzed the survey responses regarding ChatGPT-generated feedback on Java programming assignments. Most notably, we observed a promising alignment between ChatGPT-generated feedback and formative feedback guidelines, with the majority of students

offering positive responses. Furthermore, the survey results indicate a significant preference for feedback generated from the prompt that contains the student's code, as it provides more specific, clear, and corrective feedback. Lastly, we captured insights from students on how to improve ChatGPT-generated feedback, offering a direction for future work.

Formative Feedback Guidelines Alignment

We first analyzed the alignment of the ChatGPT-generated feedback with the formative feedback guidelines articulated by Shute. Figure 3 presents the survey outcomes. The data reveals that the ChatGPT-generated feedback largely aligns with formative feedback guidelines, with over 70% of students giving favorable evaluations of this alignment.

Among all the guidelines, "Give unbiased, objective feedback, written or via computer." received the highest level of agreement. Overall, 98% (100/102) of the responses mentioned that the feedback meet this criterion, indicating that most of the students perceive ChatGPT-generated feedback as unbiased and objective. In addition, 95% (97/102) of the responses affirmed the guideline "Present elaborated feedback in manageable units." This high rate of agreement can be attributed to our approach of limiting feedback to fewer than 150 words and selectively presenting only a handful of failed test cases, ensuring that the feedback remains succinct and avoiding redundancy for similar test case failures.

Meanwhile, for the guideline "Be specific and clear with feedback message", it is notable that although 71% (73/102) of the responses viewed the feedback as either somewhat clear or clear enough, only 40% (41/102) of the responses perceived the feedback as clear enough. This discrepancy underscores the potential for further refinements to enhance the specificity and clarity of ChatGPT-generated feedback.

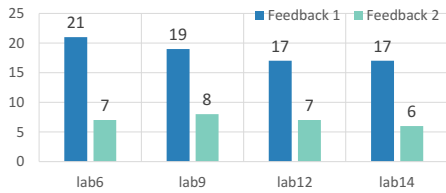


Figure 4: Students' Preference of Feedback

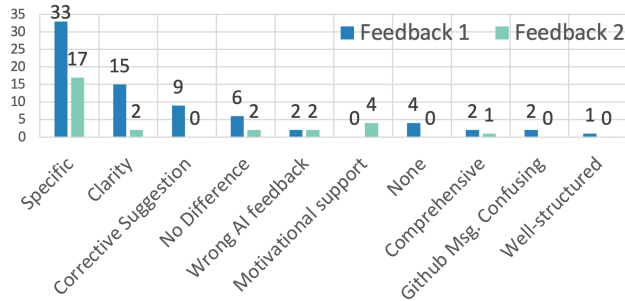


Figure 5: Students' Reasons for Their Preferred Feedback

Impacts of Code Inclusion on Students' Preference

The survey results indicate that in total 72.5% (74/102) of the responses preferred the feedback generated with the prompt that contains students' code (feedback 1) over the feedback without students' code (feedback 2). This preference is also consistent in each lab (ranging between 70.37% and 75.00%), underscoring the significance of including students' code in the prompt (Figure 4). Intuitively, ChatGPT, when provided with students' code, can offer more specific suggestions on students' code issues and thus benefit students more. Through a thematic analysis of the 102 survey responses to the open-ended question asking students why they preferred either feedback 1 or feedback 2, we found that the top three reasons that students preferred feedback 1 were its specificity (33/74), clarity (15/74) and corrective suggestions (9/74), which accounts for 74% of the survey responses for feedback 1 (Figure 5). This observation indicates that including code in the prompt enhances ChatGPT's ability to identify errors and provide actionable solutions. It also contributes to the readability and clarity of the feedback, possibly because the code snippets make the feedback more concrete than the ones without including the code.

To illustrate this, we analyzed the entirety of feedback given to students and presented a representative example in Figure 6, showing the two pieces of feedback provided to a student for the lab 12 assignment. Both pieces of feedback are well-structured, beginning with praising the student for completing the homework and ending by encouraging the student to keep up the good work.

Nonetheless, feedback 1 excels in its specificity, addressing multiple issues within the student's code, rather than proffering vague critiques. For example, feedback 1 specifically mentions the misuse of the "==" operator for string comparison, whereas feedback 2 mentions a more general issue of the palindrome logic being incorrect without pin-

Feedback Generated from Prompt Including Students' Code

Great job on completing your Java homework! I have reviewed your code and would like to provide you with some feedback to help you improve. In your `isPalindrome` method, you correctly check if the input array is null and throw an `IllegalArgumentException`. However, you also need to check if any element in the array is null and throw the same exception. This will ensure that the method handles all possible invalid inputs. In addition, I noticed that you are **comparing strings using the `==` operator instead of the `equals` method**. Remember that the `==` operator compares object references, not the actual content of the strings. To compare strings for equality, you should use the `equals` method. **In your `equals2D` method**, you correctly check if the input arrays are null and throw an `IllegalArgumentException`. However, you also need to check if the dimensions of the arrays are equal before comparing the elements. This will prevent an `ArrayIndexOutOfBoundsException` from occurring. Overall, your code is well-structured and easy to read. Keep up the good work! If you have any questions or need further clarification, feel free to ask.

Feedback Generated from Prompt Excluding Students' Code

Great job on completing Lab 12! I have reviewed your code and test cases, and I have some feedback to help you improve. In your `isPalindrome` method, it seems that you are not correctly identifying whether the array is a palindrome or not. The expected output for the given test case is "False", but your code is returning "true". I recommend reviewing your logic for checking palindromes and making sure you are correctly identifying them. **In your `equals2D` method**, it appears that you are encountering an `ArrayIndexOutOfBoundsException`. This error occurs when you are trying to access an index that is outside the bounds of the array. I suggest checking your array indices and making sure they are within the correct range. Additionally, ensure that you are comparing the elements of the arrays correctly to determine if they are equal. Remember to carefully review your code and test cases to catch any errors before submitting your work. Keep up the good work and don't hesitate to reach out if you have any questions or need further assistance.

Figure 6: Example of ChatGPT-generated Feedback

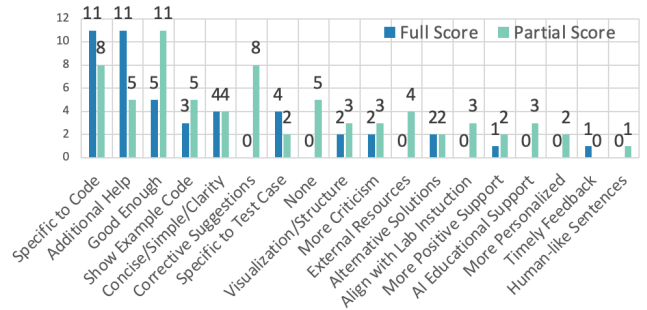


Figure 7: Students' Ideas for Improving ChatGPT-generated Feedback

pointing the core problem. Furthermore, feedback 1 also provides more explanations. This includes explaining the difference between "==" and "equals" in comparing strings, and why checking array dimensions can prevent the error of "ArrayIndexOutOfBoundsException". In contrast, feedback 2 primarily identifies errors without elucidating their underlying causes or suggesting future improvement opportunities. Consequently, feedback 1, with its more detailed and instructive nature, is favored by students.

While feedback 1 was the clear favorite among the majority, a significant 20% of participants preferred feedback 2. The primary reasons they cited for this preference were the feedback's specificity (17/28) and its motivational support (4/28). This indicates that while code snippets might make error identification easier, they may also skew the feedback too much toward fault-finding, thus reducing the motivational support provided. Despite the advantages that feedback 2 offers in terms of motivational support, it's still recommended to include code snippets in the prompt for ChatGPT-generated feedback since over 70% of the survey responses prefer feedback 1 to feedback 2.

Improvements Suggested by Students

In the survey, we included an open-ended question, in which one student could express more than one opinion, to collect students' perspectives on potential improvements to the ChatGPT-generated feedback. A thematic analysis was performed on these responses to identify patterns in students' responses. One major finding from the thematic analysis is that students appreciate the specificity, clarity, and corrective nature of the existing ChatGPT feedback generated from prompts that include students' code, as indicated in Figure 5. However, there is a clear further demand in these areas. Students expressed a particular interest in receiving clearer feedback (8/117), more corrective (8/117), and more specific to their code (19/117) and associated failed test cases (6/117) than the existing ChatGPT-generated feedback. They sought detailed explanations on which test cases their code failed, reasons for the failure, insights into the root causes of issues in their code, and how to fix those issues.

Furthermore, students advocated for the incorporation of example code for correct solutions or similar bugs (8/117). As a student suggested, "The AI-generated feedback could include other examples in other programs of bugs similar to mine and why they don't function properly". Students believed this would enhance their understanding of code issues and serve as a valuable guide to arrive at the correct solution. In addition, there is a call for receiving external resources (4/117) and additional help beyond the assignment scope (16/117) to supplement students' studies. Interestingly, the desire for such resources varied based on academic performance, as suggested by Figure 7. Students with full scores were more interested in feedback that extends beyond the course syllabus, while those with partial scores were more inclined to seek external learning materials and websites.

The analysis also reveals diverging preferences concerning the tone of the feedback. Some students were more receptive to a positive and encouraging tone (3/117), arguing that it enhances motivation. Conversely, others expressed a preference for a more critical tone (5/117) that explicitly outlines mistakes and areas requiring improvement. This finding indicates that a one-size-fits-all approach to feedback tone may not be effective, suggesting the need for personalization to better meet individual preferences.

Discussion

Teaching Implications. In this study, one critical takeaway for educators in computer science is the positive perception of ChatGPT-generated feedback by students, noting its alignment with formative feedback guidelines. This suggests that ChatGPT has considerable potential for generating practical and meaningful assignment feedback. Such automated systems could offer multiple benefits, including improving instructors' efficiency and reducing the financial overhead associated with hiring teaching assistants. Additionally, It's worth mentioning that student preferences for feedback tone varied. Some students favored a positive tone, while others sought more critical assessments. Given these differences, educators could explore personalization options to better meet students' diverse needs in real-world edu-

cational contexts. Techniques such as prompt engineering could be utilized to tailor the feedback according to individual preferences.

Research Implications. To fully integrate LLMs into the education sector, one challenge is to develop interfaces that are readily accessible to educators. A study with European teachers shows that while teachers have a favorable view of AI in education, they seem to have low AI-related skills and a basic level of digital skills (Polak, Schiavo, and Zancanaro 2022). Thus, one direction of future research is to develop an intuitive software interface that facilitates the end-to-end process from data collection to feedback distribution, enabling educators to efficiently generate, evaluate, and optimize LLM-generated feedback. In addition, in our study, students suggest that the LLM-generated feedback could be improved to include code examples and be more specific and clearer. Based on their suggestions, several opportunities exist for enhancing the quality of LLM-generated feedback. Firstly, future research could explore the feasibility of incorporating correct code answers into prompts. This might allow LLMs to identify errors in students' code efficiently by comparing students' code with correct code. Secondly, future research can explore the feasibility of fine-tuning LLMs on human-generated feedback and common student errors. By doing so, LLMs might be able to provide more specific and corrective feedback to students. Thirdly, in our study, the LLM model lacks context regarding students' prior actions and their specific needs. Addressing these limitations by improving the model's understanding of students' thought processes might enhance the assistance it can provide.

Limitations. Our study is limited by a small sample of four labs and 58 students, potentially impacting the findings' generalizability. Future research could expand the participant pool for broader validation. Furthermore, our study primarily examines student perspectives on ChatGPT-generated feedback, but future studies could assess LLM-generated feedback from various angles, including its effects on learning outcomes. Our study is also constrained by the delayed release of grades due to course policy, limiting timely access to feedback. Future research could examine settings where feedback is more immediately available, as this may improve learning outcomes and present different perspectives. Lastly, to evaluate the extent to which students view ChatGPT-generated feedback as formative, we solely look at the absolute percentage of students who provided favorable responses, using a subjective threshold of 70%. The lack of a baseline for comparison might limit the conclusiveness of our assessment. However, our focus on the student perspective adds an often overlooked but crucial dimension to the evaluation of automated educational feedback systems, offering a meaningful starting point for more comprehensive studies.

Acknowledgments

We would like to express our gratitude to Dr. Sterling McLeod from North Carolina State University for the laboratory support and the reviewers for their comments. This project was supported under NSF grant #2013502.

References

- Abramson, A. 2023. ChatGPT: A Learning Tool. <https://www.apa.org/monitor/2023/06/chatgpt-learning-tool>. Accessed: 2023-06-01.
- Ala-Mutka, K.; Uimonen, T.; and Jarvinen, H.-M. 2004. Supporting students in C++ programming courses with automatic program style assessment. *Journal of Information Technology Education: Research*, 3(1): 245–262.
- Camp, T.; Adrion, W. R.; Bizot, B.; Davidson, S.; Hall, M.; Hambrusch, S.; Walker, E.; and Zweben, S. 2017. Generation CS: the growth of computer science. *ACM Inroads*, 8(2): 44–50.
- Chang, Y.; Wang, X.; Wang, J.; Wu, Y.; Zhu, K.; Chen, H.; Yang, L.; Yi, X.; Wang, C.; Wang, Y.; et al. 2023. A survey on evaluation of large language models. *arXiv preprint arXiv:2307.03109*.
- Cheang, B.; Kurnia, A.; Lim, A.; and Oon, W.-C. 2003. On automated grading of programming assignments in an academic institution. *Computers & Education*, 41(2): 121–131.
- Chen, L.; Chen, P.; and Lin, Z. 2020. Artificial intelligence in education: A review. *Ieee Access*, 8: 75264–75278.
- Corbett, A. T.; and Anderson, J. R. 1994. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4: 253–278.
- Cordova, L.; Carver, J.; Gershmel, N.; and Walia, G. 2021. A comparison of inquiry-based conceptual feedback vs. traditional detailed feedback mechanisms in software testing education: an empirical investigation. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 87–93.
- Dijkstra, R.; Genç, Z.; Kayal, S.; Kamps, J.; et al. 2022. Reading Comprehension Quiz Generation using Generative Pre-trained Transformers.
- Dixson, D. D.; and Worrell, F. C. 2016. Formative and summative assessment in the classroom. *Theory into practice*, 55(2): 153–159.
- Elsen-Rooney, M. 2023. NYC schools ban ChatGPT writing artificial intelligence. <https://ny.chalkbeat.org/2023/1/3/23537987/nyc-schools-ban-chatgpt-writing-artificial-intelligence>. Accessed: 2023-01-03.
- Finnie-Ansley, J.; Denny, P.; Becker, B. A.; Luxton-Reilly, A.; and Prather, J. 2022. The robots are coming: Exploring the implications of openai codex on introductory programming. In *Proceedings of the 24th Australasian Computing Education Conference*, 10–19.
- Gulwani, S.; Radiček, I.; and Zuleger, F. 2018. Automated clustering and program repair for introductory programming assignments. *ACM SIGPLAN Notices*, 53(4): 465–480.
- Hao, Q.; Smith IV, D. H.; Ding, L.; Ko, A.; Ottaway, C.; Wilson, J.; Arakawa, K. H.; Turcan, A.; Poehlman, T.; and Greer, T. 2022. Towards understanding the effective design of automated formative feedback for programming assignments. *Computer Science Education*, 32(1): 105–127.
- Hao, Q.; Wilson, J. P.; Ottaway, C.; Iriumi, N.; Arakawa, K.; and Smith, D. H. 2019. Investigating the essential of meaningful automated formative feedback for programming assignments. In *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 151–155. IEEE.
- Irons, A.; and Elkington, S. 2021. *Enhancing Learning through Formative Assessment and Feedback*. Key Guides for Effective Teaching in Higher Education. Taylor & Francis. ISBN 9781000421125.
- J., J. 2023. ChatGPT API Transition Guide. <https://help.openai.com/en/articles/7042661-chatgpt-api-transition-guide/>. Accessed: 2023-06-10.
- Kathe Pelletier, N. M. M. J. R. N. A. S. G. T. B. D. L. J. M. A. M. A. P. R. R. J. Z., Jenay Robert. 2023. 2023 EDUCAUSE Horizon Report: Teaching and Learning Edition. <https://library.educause.edu/resources/2023/5/2023-educause-horizon-report-teaching-and-learning-edition>. Accessed: 2023-05-08.
- Liu, D.; and Petersen, A. 2019. Static analyses in python programming courses. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 666–671.
- MacNeil, S.; Tran, A.; Mogil, D.; Bernstein, S.; Ross, E.; and Huang, Z. 2022. Generating diverse code explanations using the gpt-3 large language model. In *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 2*, 37–39.
- Messer, M.; Brown, N. C.; Kölling, M.; and Shi, M. 2023. Automated Grading and Feedback Tools for Programming Education: A Systematic Review. *arXiv preprint arXiv:2306.11722*.
- Morris, D. S. 2003. Automatic grading of student’s programming assignments: an interactive process and suite of programs. In *33rd Annual Frontiers in Education, 2003. FIE 2003.*, volume 3, S3F–1. IEEE.
- OpenAI. 2023a. Temperature parameter in ChatGPT model. <https://beta.openai.com/docs/api-reference/completions/create/?lang=python#completions/create-temperature>. Accessed: 2023-06-10.
- OpenAI. 2023b. Token limits on ChatGPT. <https://platform.openai.com/docs/guides/chat/managing-tokens>. Accessed: 2023-06-10.
- Pankiewicz, M.; and Baker, R. S. 2023. Large Language Models (GPT) for automating feedback on programming assignments. *arXiv:2307.00150*.
- Pardos, Z. A.; and Bhandari, S. 2023. Learning gain differences between ChatGPT and human tutor generated algebra hints. *arXiv preprint arXiv:2302.06871*.
- Piech, C.; Huang, J.; Nguyen, A.; Phulsuksombati, M.; Sahami, M.; and Guibas, L. 2015. Learning program embeddings to propagate feedback on student code. In *International conference on machine Learning*, 1093–1102. PMLR.
- Polak, S.; Schiavo, G.; and Zancanaro, M. 2022. Teachers’ perspective on artificial intelligence education: An initial investigation. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, 1–7.
- Qureshi, B. 2023. Exploring the use of chatgpt as a tool for learning and assessment in undergraduate computer science curriculum: Opportunities and challenges. *arXiv preprint arXiv:2304.11214*.

- Rose, J. 2023. ChatGPT as a teaching tool, not a cheating tool. *Times Higher Education*.
- Sarsa, S.; Denny, P.; Hellas, A.; and Leinonen, J. 2022. Automatic generation of programming exercises and code explanations using large language models. In *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 1*, 27–43.
- Shi, Y.; Chi, M.; Barnes, T.; and Price, T. 2022. Code-DKT: A code-based knowledge tracing model for programming tasks. *arXiv preprint arXiv:2206.03545*.
- Shi, Y.; Shah, K.; Wang, W.; Marwan, S.; Penmetsa, P.; and Price, T. 2021. Toward semi-automatic misconception discovery using code embeddings. In *LAK21: 11th International Learning Analytics and Knowledge Conference*, 606–612.
- Shute, V. J. 2008. Focus on formative feedback. *Review of educational research*, 78(1): 153–189.
- Singh, R.; Gulwani, S.; and Solar-Lezama, A. 2013. Automated feedback generation for introductory programming assignments. In *Proceedings of the 34th ACM SIGPLAN conference on Programming language design and implementation*, 15–26.
- Skripchuk, J.; Shi, Y.; and Price, T. 2022. Identifying Common Errors in Open-Ended Machine Learning Projects. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education-Volume 1*, 216–222.
- VanLehn, K. 1988. Student modeling. *Foundations of intelligent tutoring systems*, 55: 78.
- Von Matt, U. 1994. Kassandra: the automatic grading system. *ACM SIGCUE Outlook*, 22(1): 26–40.
- Wiggins, J. B.; Boyer, K. E.; Baikadi, A.; Ezen-Can, A.; Grafsgaard, J. F.; Ha, E. Y.; Lester, J. C.; Mitchell, C. M.; and Wiebe, E. N. 2015. JavaTutor: an intelligent tutoring system that adapts to cognitive and affective states during computer programming. In *Proceedings of the 46th acm technical symposium on computer science education*, 599–599.