The Impact of Student-AI Collaborative Feedback Generation on Learning Outcomes

Anjali Singh SINGHANJ@UMICH.EDU
Christopher Brooks BROOKSCH@UMICH.EDU
Xu Wang XWANGHCI@UMICH.EDU
University of Michigan

Abstract

This study explores the effectiveness of student-AI collaborative feedback generation for enhancing student feedback authors' learning outcomes. Situated in an online graduate-level data science course, this research compares different learning designs engaging students in writing hints for incorrect programming assignment solutions. We conducted an experiment comparing three designs: *Baseline* (students independently writing hints), *AI-assistance* (students writing hints with on-demand access to hints for the incorrect solution generated by GPT-4), and *AI-revision* (students first writing hints independently, then revising them after viewing GPT-4 generated hints). Our findings suggest that the *AI-revision* approach, which encourages students to first attempt the task independently and then engage with AI-generated content, can lead to better learning outcomes compared to the *AI-assistance* approach, which offers students constant access to AI-generated hints. The *AI-revision* approach prompts students to critically evaluate AI-generated responses and refine their own hints, leading to deeper understanding. This work contributes to research exploring generative AI integration in educational settings, emphasizing the need for designs that promote active student engagement with AI tools.

Keywords: Peer Feedback, Hint Writing, Student-AI Collaboration

1. Introduction

Peer feedback, where students of similar competence evaluate and provide constructive feedback on each other's work, serves as a critical tool for active learning (Liu and Carless, 2006). It benefits students' learning by enabling them to reflect on their own work through the lens of their peers' assessments and developing evaluating expertise (Harris and Brown, 2013). Peer feedback systems can also help in providing formative feedback at scale in resource constrained educational settings. However, in complex domains, writing good quality feedback can be challenging for learners (Singh et al., 2022). Consider the task of writing a hint as a form of formative feedback to a peer's incorrect assignment submission: first, the learner must identify the set of mistakes in the incorrect solution and map those mistakes to misconceptions. Then, they must determine appropriate domain-based remediation strategies. Finally, they should provide specific feedback to address the mistakes without giving away the full solution while ensuring that they use a positive and encouraging tone.

The advent of conversational Large Language Models (LLMs) such as GPT (Brown et al., 2020) and LLama (Touvron et al., 2023) has fueled new interest in using AI directly for providing feedback to learners. However, recent work has shown that providing LLM-generated feedback to learners can make them over-reliant on such support (Pankiewicz

and Baker, 2023). Moreover, limitations of these models to answer domain-specific prompts truthfully (Balse et al., 2023) have necessitated expert oversight for incorporation into educational settings.

To overcome the limitations of these two approaches (peer feedback and using AI directly for feedback), we explore the dynamics of student-AI collaborative hint writing to generate high-quality hints while providing student hint writers with a meaningful learning experience. Recently, we conducted a study with adult learners, comparing two designs: students writing hints independently versus students verifying and revising hints generated by GPT-4 Singh et al. (2023). We found that providing LLM-generated hints to student hint-writers can be a useful way to scaffold the peer feedback task of hint writing, especially when the LLM-generated hints are highly accurate. We also found that students who revised GPT-4 hints wrote better phrased and more specific hints and that this design helped the majority of students think critically about LLM responses. However, on probing students' preferences for writing hints with or without AI support, we did not find a clear preference. In particular, students' lack of trust in LLMs and willingness to engage in this activity independently without getting support or getting biased emerged as prominent reasons for preferring to write hints without AI support. This work emphasized the need for careful design in integrating AI in educational tasks and suggested providing AI-generated hints to students *after* they have attempted the task independently for a 'second opinion'.

This work extends this line of research through a randomized controlled experiment with adult learners enrolled in an online graduate-level introductory data science course. In our experiment, students were prompted to write a hint for an incorrect solution to a programming assignment they had recently worked on by comparing it to a correct solution. This experiment had three conditions: *Baseline*, where students wrote a hint for the incorrect submission by themselves, (2) *AI-assistance*, where students wrote a hint with GPT-4's assistance, i.e., they could click on a button to view the hint generated by the LLM GPT-4 for the same incorrect solution, and *AI-revision* where students first wrote a hint on their own, then saw a hint generated by GPT-4 for the same incorrect solution, and then re-wrote the hint. Such student-AI collaborative endeavors can also serve as opportunities to engage students in meaningful interactions with LLMs to help foster their critical thinking skills in the context of both the underlying curricula and validating the correctness of LLM-generated outputs.

With the goal of understanding the impact of students' interactions with LLM-generated hints on their learning outcomes, this work aims to answer the following research question:

RQ: What is the difference in the learning outcomes of the students when they write hints independently (*Baseline*), versus when they write a hint with the option to view the GPT-4 hint at all times (*AI-assistance*), versus when they first write a hint on their own, then view the GPT-4 hint and then rewrite the hint (*AI-revision*)?

This work contributes to understanding how students' interactions with LLMs impact their learning outcomes. We recognize the importance of comparing the quality of hints written by students in each experimental condition. However, we leave this in-depth analysis for future work. To instigate future research in this area, we release the collected dataset of student hints, LLM-generated hints, and student-LLM collaboratively generated hints

along with the incorrect programs and the accompanying correct programs for which these hints were written[1].

2. Related Work

2.1. Leveraging Generative AI for Feedback Generation

Given the recent advances in generative AI, researchers exploring the use of LLMs for generating feedback have found that LLMs like GPT-3.5 are able to generate helpful personalized hints for students solving programming assignments. However, they also cautioned that students may over-rely on such feedback (Pankiewicz and Baker, 2023). Others have reported issues such as high variability in the accuracy of identified mistakes and suggested fixes in the generated feedback (Balse et al., 2023). One of the proposed solutions is to leverage human-AI collaboration by involving humans-in-the-loop to both provide inputs to and evaluate and improve the LLM-generated artifacts (Denny et al., 2022). We explore this opportunity by exploring two different designs of involving learners in evaluating and revising LLM-generated hints for incorrect programming assignment solutions. Both designs (one where students can request for the LLM-generated hint to be displayed at any time during the activity, and another where students first work on their hint independently, then view the LLM-generated hint and finally revise their original hint) are inspired from the work of Singh et al. (2023). The latter design also aligns with suggestions made by (Choi et al., 2022) and has been found useful for improving the quality of student-generated artifacts.

2.2. Learning from Reflecting on Mistakes

Prior learning sciences research has highlighted the learning benefits from contrasting incorrect methods with correct ones, such as focusing students' attention on the distinguishing features of the correct examples and the underlying concepts (Durkin and Rittle-Johnson, 2012), and improving procedural flexibility (Rittle-Johnson and Star, 2011). In addition to identifying mistakes, the task of hint-writing involves *explaining* the mistakes and how to fix them without giving away the full solution. Elaborating on mistakes in domains prioritizing problem-solving has been found to augment students' learning outcomes substantially (Loibl and Leuders, 2019). Data science is one such domain where students concurrently learn effective ways of problem-solving and communicating data-driven insights. Actively engaging learners in writing constructive hints for incorrect solutions to problems they recently solved can encourage them to think deeply about data science code and compare different approaches to solving the same problem.

3. Method

3.1. Course Context

The study was conducted in a Data Manipulation course offered to graduate students as part of the online Masters of Applied Data Science program at the University of Michigan. This program consists of adult learners who are required to have introductory programming and

1. Link to dataset

statistics knowledge. The JupyterLab computational notebook environment[2] and Python are used for the programming component of this course. The course had four weekly programming assignments that were automatically graded using the Nbgrader (Blank et al., 2019) tool. Students were allowed unlimited submissions until the deadline for the programming assignments, which resulted in the majority of them eventually obtaining full points.

3.2. Experiment Design

Students were randomly assigned to one of the three experimental conditions: *AI-assistance*, *AI-revision*, and *Baseline*, and this assignment remained the same for the entire duration of the experiment. After each of the second and third programming assignments, students were given a hint-writing assignment in which they were prompted to compare a correct solution C to a question from the latest programming assignment to an incorrect solution I that was given to them. Following the comparison between the correct and incorrect solutions, students were asked to write a hint so that the person who wrote the incorrect solution could use it to understand and fix their mistakes. Each hint-writing assignment was worth 5% of the total course grade. Students received full points for completing the assignment and no points otherwise.

At the beginning of the hint-writing assignment, students were informed of the potential learning benefits of this exercise, i.e., encouraging them to think critically, learn from mistakes, and improve their problem-solving skills. Next, we provided a simple example of the task to the students. This example was based on an incorrect solution to another question (different from the one on which the hint-writing task was based) from the latest programming assignment. After going through the worked example, students proceeded to the main exercise. The programming assignment instructions were reproduced, and they were shown the code for a correct and an incorrect solution side-by-side. Students in the *AI-assistance* and *AI-revision* conditions were also informed that the GPT-4 hint they would be shown could be incorrect, incomplete, or both, and it would be their job to validate if it is correct and appropriately phrased.

Next, the instructions for the hint-writing activity were displayed, which depended on the condition to which a student was assigned. All students were asked to write a hint for the incorrect solution. Students in the *Baseline* condition were not given any additional support in the hint-writing task. Students in the *AI-assistance* condition could click on a button with the text "Show ChatGPT Hint" below the text box in which they wrote their hint at any point during the activity. This design mimics AI assistants that we typically encounter, such as Grammarly's AI Assistant[3]. Clicking on this button led to the GPT-4 for the incorrect solution being displayed. While we engaged in prompt engineering with GPT-4 for this task, we are not aiming to answer whether LLM-generated hints can be used directly per se, as the value of student-AI collaborative hint writing goes beyond feedback generation and provides metacognitive benefits to hint writers. hint for the incorrect solution being displayed. For students in the *AI-revision* condition, the GPT-4 hint was displayed after they wrote a hint on their own, after which they were prompted to compare their hint with

2. https://jupyter.org/
3. https://www.grammarly.com/ai

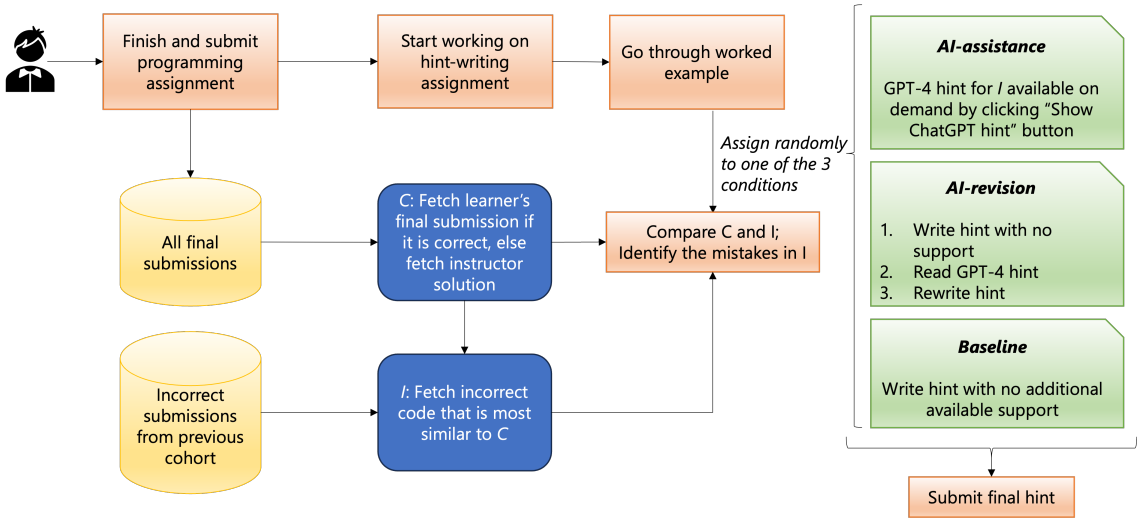

Figure 1: The study pipeline, starting from students submitting a programming assignment solution, to working on the hint writing assignment by comparing their correct solution (or instructor solution if their solution was incorrect) to an incorrect solution, to finally submitting their hint.

the GPT-4 hint and then rewrite the hint. While this design was the most complex due to reasons such as requiring students to both verify the correctness of the GPT-4 hint and write the hint twice, we expected this to benefit students' learning the most.

3.3. Selection of Correct and Incorrect Solutions

If a student's final submission for the latest programming assignment question was correct, it was used as the correct solution C; otherwise, the correct solution supplied by the instructor served as C. The incorrect solution I was selected from a repository R of incorrect solutions submitted by learners from previous offerings of the course. For a given correct solution C', the most similar incorrect solution I' was selected from R using a keyword-based similarity metric. For a given program, we consider all the imported libraries and built-in functions to be the keywords in that program. The keyword-based similarity metric computes the Jaccard coefficient (the ratio of the number of items in the intersection by the union of two sets) (Niwattanakul et al., 2013) between the keywords obtained from two programs. The rationale behind the selection criteria of C and I was to provide students an opportunity to reflect on their own correct solution and compare it to a similar incorrect solution. Comparison to an incorrect solution that uses a strategy that is very different from the correct solution can increase students' cognitive load in an already complex task, which could lead to students writing poor-quality hints.

Figure 1 provides an overview of the entire pipeline, from students submitting a programming assignment, to working on the hint writing assignment by comparing a correct solution to an incorrect solution and finally submitting their hint.

3.4. Prompting GPT-4

The prompt used to generate the GPT-4 hints is provided here. While we engaged in prompt engineering with GPT-4 for this task, we are not aiming to answer whether LLM-generated hints can be used directly per se, as the value of student-AI collaborative hint writing goes beyond feedback generation and provides metacognitive benefits to hint writers. Therefore, after some experimentation with prompting (for instance, asking GPT-4 to write concise hints when it produced verbose hints and not referring to the correct solution provided in the prompt in the hint text) we selected the first hint that GPT-4 generated to our final prompt.

3.5. Measuring Students' Learning Outcomes:

To measure the impact of the different hint-writing designs on students' learning outcomes, students were given a pre-test at the beginning and a post-test at the end of the course. The pre-test consisted of 10 Multiple Choice Questions (MCQs), with each MCQ having a single correct answer. The pre-test was a "practice test" that students were encouraged to take but did not contribute to their overall course grade. This test assessed students' introductory Python knowledge. The post-test consisted of 2 MCQs with a single correct answer (worth 1 point each) and 4 MCQs with more than one correct answer (worth 2 points each). In total, the post-test was worth 10 points and contributed to 5% of students' overall course grade. This test assessed students' knowledge in data manipulation, especially debugging skills, which we expected the students to acquire from engaging in the hint-writing assignments. Additionally, students had the option to provide feedback to the instructional team on their experience with these assignments.

4. Results and Discussion

We now report the results of our study based on the impact on students' learning outcomes. We use $p < 0.05$ as the criterion for assessing statistical significance. In total, 97 students gave the pre-test, out of which 62 students gave the post-test. While the post-test instructions mentioned that it was worth 5% of the total points in the course due to a printing mistake in the course syllabus, some students inferred that the post-test was optional. Consequently, we were unable to collect the post-test data for 35 students.

Out of the 62 students who gave both the pre- and post-tests, the number of students assigned to the conditions *AI-assistance*, *AI-revision* and *Baseline* were 27, 15, and 20, respectively. An ANOVA revealed that the difference in prior knowledge of these students between conditions as measured by their pre-test scores approached statistical significance ($p = 0.09$). Therefore, we used propensity score matching to select a sample of 20 students from the *AI-assistance* group whose pre-test scores were similar to those of the mean scores of the other two groups. Following this, we had 20, 15, and 20 students in the groups *AI-assistance*, *AI-revision* and *Baseline*. For these students, an ANOVA revealed that there was no significant difference in prior knowledge of students between conditions as measured by their pre-test scores ($p = 0.82$). Table 1 shows the aggregates of students' pre- and post-test scores for each experimental condition.

Condition	# students	Pre-test score (mean, std)	Post-test score (mean, std)
AI-assistance	20	(7.55, 1.05)	(4.75, 1.65)
AI-revision	15	(7.60, 1.55)	(5.67, 1.84)
Baseline	20	(7.80, 1.40)	(5.70, 1.81)

Table 1: Number of students and aggregates of pre- and post-test scores for each experimental condition. For each test, the maximum score was 10.

An ANOVA revealed that the difference in post-test scores of students between conditions was not statistically significant ($p = 0.18$). Despite the lack of statistical significance, the low mean score of students in the *AI-assistance* condition suggests that providing AI-based support to students on demand could be detrimental to their learning. This group of students was not given any incentive to work on the hint-writing assignment on their own, as they had the option to use the GPT-4 hint as is. Further, as highlighted by Singh et al. (2023), this design can even bias learners into aligning their thinking with the AI-generated response. The higher scores for the *AI-revision* group, where students first wrote a hint on their own before viewing the GPT-4 hint, suggest that students can learn more when prompted to first think of the solution on their own before seeking AI-based assistance. This design also allows students to compare their own hint to the GPT-4 hint, which can be helpful for both improving their original hint (Choi et al., 2022) as well as thinking about the accuracy and appropriateness of LLM-generated responses.

The response from students for the hint-writing assignments was overall quite positive. Students found these assignments helpful for practicing debugging and critical thinking skills and mentioned that providing feedback on incorrect solutions helped them "think like a data scientist". Several students expressed interest in providing such feedback to their peers in real time.

5. Limitations

All the students in the course learned the same content and engaged in the same activities, except for the hint writing activity, which varied with the experimental conditions. Therefore, the differences in the observed impact on students' learning outcomes are likely to be due to the varying design of the hint writing activities. However, since our sample size was small, there is a possibility that other factors that were not within our control impacted students' learning outcomes. Another limitation of our study is that we were unable to obtain the post-test scores for all students, which further reduced the study's statistical power. Future studies with a larger number of students can be helpful for obtaining statistically significant findings. Further, having multiple repetitions of such student-AI collaborative learning activities over a long period of time, with learning outcomes being evaluated both immediately on course completion and sometime after the course, can be helpful to measure the impact on students' learning in the long and short terms. A third limitation is that we focused specifically on a single course at a single institution, although having two hint-writing interventions based on two different programming assignments helped address this limitation to some extent. Finally, to understand the trade-off between generating

high-quality hints and improving students' learning outcomes, there is a need to conduct an in-depth comparison of the quality of student-generated hints using the three designs from this study.

6. Conclusion and Future Work

This work contributes to research on understanding the impacts of LLMs in education and provides design implications for optimizing student learning as students interact with LLMs. Overall, our findings highlight the importance of exploring more 'active' interaction designs between students and LLMs, where students are encouraged to actively engage with LLM-generated responses rather than passively consuming them. In the future, we plan to use the student-generated hints of high quality to improve the accuracy and pedagogical appropriateness of hints generated by LLMs using chain-of-thought prompting and fine-tuning techniques.

References

Rishabh Balse, Bharath Valaboju, Shreya Singhal, Jayakrishnan Madathil Warriem, and Prajish Prasad. Investigating the potential of gpt-3 in providing feedback for programming assessments. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*, pages 292–298, 2023.

Douglas S Blank, David Bourgin, Alexander Brown, Matthias Bussonnier, Jonathan Frederic, Brian Granger, Thomas L Griffiths, Jessica Hamrick, Kyle Kelley, M Pacer, et al. nbgrader: A tool for creating and grading assignments in the jupyter notebook. *The Journal of Open Source Education*, 2(11), 2019.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Kabdo Choi, Hyungyu Shin, Meng Xia, and Juho Kim. Algosolve: Supporting subgoal learning in algorithmic problem-solving with learnersourced microtasks. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–16, 2022.

Paul Denny, Sami Sarsa, Arto Hellas, and Juho Leinonen. Robosourcing educational resources–leveraging large language models for learnersourcing. *arXiv preprint arXiv:2211.04715*, 2022.

Kelley Durkin and Bethany Rittle-Johnson. The effectiveness of using incorrect examples to support learning about decimal magnitude. *Learning and Instruction*, 22(3):206–214, 2012.

Lois R Harris and Gavin TL Brown. Opportunities and obstacles to consider when using peer-and self-assessment to improve student learning: Case studies into teachers' implementation. *Teaching and Teacher Education*, 36:101–111, 2013.

Ngar-Fun Liu and David Carless. Peer feedback: the learning element of peer assessment. *Teaching in Higher education*, 11(3):279–290, 2006.

Katharina Loibl and Timo Leuders. How to make failure productive: Fostering learning from errors through elaboration prompts. *Learning and Instruction*, 62:1–10, 2019.

Suphakit Niwattanakul, Jatsada Singthongchai, Ekkachai Naenudorn, and Supachanun Wanapu. Using of jaccard coefficient for keywords similarity. In *Proceedings of the international multiconference of engineers and computer scientists*, volume 1, pages 380–384, 2013.

Maciej Pankiewicz and Ryan S Baker. Large language models (gpt) for automating feedback on programming assignments. *arXiv preprint arXiv:2307.00150*, 2023.

Bethany Rittle-Johnson and Jon R Star. The power of comparison in learning and instruction: Learning outcomes supported by different types of comparisons. In *Psychology of learning and motivation*, volume 55, pages 199–225. Elsevier, 2011.

Anjali Singh, Christopher Brooks, and Shayan Doroudi. Learnersourcing in theory and practice: synthesizing the literature and charting the future. In *Proceedings of the Ninth ACM Conference On Learning@ Scale*, pages 234–245, 2022.

Anjali Singh, Christopher Brooks, Xu Wang, Warren Li, Juho Kim, and Deepti Pandey. Bridging learnersourcing and ai: Exploring the dynamics of student-ai collaborative feedback generation. *arXiv preprint arXiv:2311.12148*, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.