Improving AI in CS50

Leveraging Human Feedback for Better Learning

Rongxin Liu
Harvard University
Cambridge, MA, United States
rongxinliu@cs50.harvard.edu

Julianna Zhao
Harvard University
Cambridge, MA, United States
juliannazhao@college.harvard.edu

Benjamin Xu
Yale University
New Haven, CT, United States
ben.xu@yale.edu

Christopher Perez
Harvard University
Cambridge, MA, United States
christopherperez@college.harvard.edu

Yuliia Zhukovets
Harvard University
Cambridge, MA, United States
yuliia@cs50.harvard.edu

David J. Malan
Harvard University
Cambridge, MA, USA
malan@harvard.edu

Abstract

In 2023, we developed and deployed AI-based tools in CS50 at Harvard University to provide students with 24/7 interactive assistance, approximating a 1:1 teacher-to-student ratio. These tools offer code explanations, style suggestions, and responses to course-related inquiries, emulating human educators to foster critical thinking. However, maintaining alignment with instructional goals is challenging, especially with frequent updates to the underlying large language models (LLMs). We thus propose a continuous improvement process for LLM-based systems using a collaborative human-in-the-loop approach. We introduce a systematic evaluation framework for assessing and refining the performance of AI-based tutors, combining human-graded and model-graded evaluations. Using few-shot prompting and fine-tuning, we aim to ensure our AI tools adopt pedagogically sound teaching styles. Fine-tuning with a small, high-quality dataset has shown significant improvements in aligning with teaching goals, as confirmed through multi-turn conversation evaluations. Additionally, our framework includes a model-evaluation backend that teaching assistants periodically review, ensuring the AI system remains effective and aligned with instructional objectives. This paper offers insights into our methods and the impact of these AI tools on CS50 and contributes to the discourse on AI in education, showcasing scalable, personalized learning enhancements.

CCS Concepts

• **Social and professional topics → CS1**; • **Applied computing → Computer-assisted instruction**; • **General and reference → Evaluation**.

Keywords

AI, artificial intelligence, generative AI, large language models, LLMs

ACM Reference Format:
Rongxin Liu, Julianna Zhao, Benjamin Xu, Christopher Perez, Yuliia Zhukovets, and David J. Malan. 2025. Improving AI in CS50: Leveraging Human Feedback for Better Learning. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE TS 2025), February 26-March 1, 2025, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3641554.3701945

1 Introduction

In 2023, we developed and integrated a suite of AI-based tools using large language models (LLMs) into CS50, Harvard University's introductory course in computer science, to enhance the learning experience for students both on campus and online [26]. LLMs such as those based on Generative Pre-trained Transformers (GPT) are advanced AI systems capable of processing and generating human-like text, providing significant benefits in educational settings by offering immediate, interactive assistance [17, 19, 36]. Our approach approximated a 1:1 teacher-to-student ratio with tools like the CS50 Duck, an AI-powered chatbot that offers 24/7 support. We also employed the duck on Ed [6] to manage discussion forum questions and utilized two Visual Studio Code (VS Code) extensions for explaining code snippets and suggesting style improvements.

After testing with 70 students in Summer 2023, we expanded our AI tools to several hundred on-campus students in Fall 2023 and thousands online. Feedback was positive, with 75% of students using the tools frequently and 94% finding them helpful and effective [26]. By mid-November 2024, by way of a massive open online course, approximately 211,000 students had used the duck, which has processed 10 million queries at an average cost of $1.50 per student per year.

However, our ongoing use of AI tools has revealed shortcomings. A phenomenon that we might call "instruction dilution" has led to AI responses misaligned with teaching objectives when the system prompt (the set of instructions provided to the AI model that outlines how it should behave and respond [28]) is too large (e.g., our system prompts close to 1,000 tokens, containing overly detailed instructions). For example, despite instructions to avoid providing direct solutions to homework problems, the tools have often generated examples that were overly helpful and, with minor adjustments, could become a complete solution. Additionally, the lack of a robust system for AI system performance evaluation has made it difficult to ensure the LLM responses aligned with student and staff expectations.

To address these challenges and maintain alignment with teaching goals, we propose the following solutions:

- **Human-in-the-loop approach:** Continuously integrating student and instructor feedback into the AI system evaluation and improvement process [18].
- **Few-shot prompting:** Providing a few example interactions within the system prompt to guide the AI's responses.
- **Fine-tuning model:** Adjusting a pre-trained model to better align its responses with a curated dataset of example interactions that reflect desired teaching behaviors.

This paper addresses issues with our first integration of AI into CS50 and proposes a multifaceted approach for improvement that is largely applicable to other pedagogical AI tools. By sharing our methods and insights, we aim to contribute to the broader discourse on AI in education and demonstrate how scalable, personalized learning enhancements can be achieved through thoughtful AI evaluation and integration.

2 Motivation

Integration of AI-based tools in education offers significant benefits, as demonstrated by systems like CodeAid at the University of Toronto, which provides programming support while avoiding direct code solutions by using pseudo-code generation and line-by-line explanations to foster deeper conceptual engagement [23]. However, such tools face challenges in aligning with subjective pedagogical goals [38]. For instance, while CodeAid achieves 79% technical correctness, it struggles with complex debugging tasks and maintaining consistent pedagogical approaches across queries. These challenges highlight the difficulty of ensuring AI systems meet pedagogical standards, motivating our efforts to enhance the CS50 Duck's alignment with teaching objectives.

2.1 Our Pedagogical Goals

Cognitive science research has demonstrated that the student's act of discovery and inquiry are critical for information retention and future problem-solving [2]. Computer science education should aim to teach not only coding mechanisms but also the explanations behind them [35], fostering student autonomy and motivation in the learning process. However, creating an AI tutor using existing LLMs with such pedagogical goals in mind proves to be difficult. Current tools like ChatGPT [29], trained to be helpful in conversations, focus on providing immediate assistance, hindering deeper, long-term student understanding [10]. Our goal has been to develop an LLM-based system that, like a human teaching fellow (TF), promotes problem-solving and learning through interactive engagement.

Our system architecture for the CS50 Duck can be summarized as follows:

1. **Centralized AI Backend:** The CS50 Duck is powered by a web server, CS50.ai, that relays student messages to GPT-4o hosted on Azure [16, 27, 32]. It serves as the central backend for all our AI-based tools, incorporating pedagogical guardrails to ensure responses provide constructive feedback and guidance rather than outright solutions.
2. **Instructional System Prompt:** We employ a comprehensive system prompt that directs LLM behavior and corrects outdated responses[28].
3. **Hallucination Mitigation:** To improve AI response accuracy, we use Retrieval-Augmented Generation (RAG) [15], whereby we process lecture captions and store them in a searchable vector database [4], allowing the AI to reference relevant information [30], reducing the chance of incorrect or fabricated responses [12].

While our CS50 Duck functions satisfactorily overall [26], we have identified key areas where it requires improvement.

2.1.1 *Instruction Dilution.*

A major issue we identified is what we might call "instruction dilution," whereby the AI model fails to adhere to guidelines in a complex system prompt. Similar issues have been observed in other studies where LLMs struggle to follow complex prompts because of the inherent limitations in their instruction-following capabilities [25].

For example, despite explicit instructions to the CS50 Duck not to generate direct code solutions for students, instruction dilution sometimes leads to non-compliance, with the duck providing extensive code blocks. A code block, in this context, refers to Markdown code blocks detected in the generated output using patterns such as `` ```c `` to indicate C code and `` ```python `` to indicate Python code. Our analysis of 10 million chat messages between our students and the duck revealed that approximately 2.1 million responses—22% of all interactions—contained code blocks. At the conversation level, defined as a sequence of user inputs and duck responses within a single session, this percentage becomes 48%: around 635 thousand out of 1.3 million conversations involved code generation.

Shifting from using OpenAI's GPT-4 to GPT-4o [32] as our underlying model has increased the frequency of code block generation unexpectedly, further highlighting the instruction dilution problem. Table 1 shows a comparison of the frequency of code block generation before and after the model version update.

Table 1: Comparison of code block generation frequency across models

Model	Messages	Code Blocks Generated	Message Level %	Conversation Level %
gpt-4	6,487,201	1,326,273	20%	44%
gpt-4o	3,203,702	817,739	25%	56%

The high frequency of code block generation raises concerns that students might rely too much on the CS50 Duck for answers, without writing as much code of their own, which could potentially prevent them from fully engaging with the problem-solving process [22].

2.1.2 *Ineffective Teaching Styles.*

Effective teaching involves not only delivering content but also engaging students in a manner that fosters understanding and critical thinking [20]. We argue, then, that AI tutors should use language and tone that are supportive and constructive without being overly directive.

Our analysis of the CS50 Duck's message history data indicates that AI models often provide direct answers, even when examples or hints are requested. This directness can be seen in frequently used phrases such as "here's an <example>" (used 1.9 million times) and "here's how you can <verb>" (used 210 thousand times out of 10 million messages). These phrases often lead to responses that are overly prescriptive, providing complete solutions instead of just guidance.

For instance, when the AI says "here's a basic example," we have found that it often supplies a full answer to the student's query. When it claims to provide just a hint to a problem, it often lists in-depth descriptions of next steps for the student to take. In similar cases, a human TF may instead ask leading questions for the student to work their way to an answer or strategy themselves [9].

The CS50 Duck's tendency to provide extensive code examples or action items risks hindering students' learning [8]. While this amount of detail may be helpful in the short term, it undermines the learning process of independent work [37]. At the current stage of our project, we aim for our AI tools to imitate the behavior of our best human TFs in guiding students to find solutions themselves and prioritizing long-term learning.

2.1.3 Continuous AI Evaluation Challenges. Finally, we have lacked a systematic method of evaluating the AI tutor's qualitative performance, which limits the visibility of the tool's functionality and hinders the ability to determine suboptimal performance. Updates to the underlying AI model, made by OpenAI, can change behavior in unforeseen ways; the shift from GPT-4 to GPT-4o increasing code generation is an example. Furthermore, not only should we ensure that new models do not cause pedagogical regressions, we also want to guarantee that updates made to our own system prompt work in helping to align the AI tutor with our pedagogical goals. Because we have not had a structured evaluation framework, these performance metrics have been difficult to measure, and having TFs manually reviewing AI responses is time-consuming and not scalable in large educational settings.

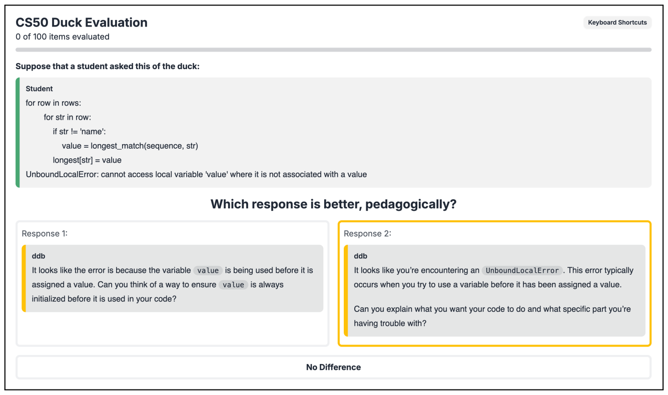
3 Solutions

To address these challenges, we propose a multifaceted approach to improve the CS50 Duck: an evaluation platform to continuously assess and refine performance and using techniques like"few-shot prompting" and "fine-tuning" to emulate human-like teacher responses and improve the quality of interactions.

3.1 AI System Evaluation Platform

To systematically assess and improve the quality of our system prompts, we developed an evaluation platform designed for prompt refinement through structured feedback. This platform facilitates pairwise comparisons of responses generated by two models to the same student query, a widely recognized approach for evaluating LLMs [3]. The platform supports two types of evaluation: one focused on single-turn responses, where users select the preferred response from two models (Figure 1), and another involving multi-turn interactions, where users engage in conversations with both models before determining their preference. The single-turn evaluation requires no additional generation during the interaction, while the multi-turn evaluation captures the dynamic nature of conversational contexts.

The evaluation process begins with the creation of a dataset containing student queries. For single-turn evaluation, each query is paired with AI-generated responses from two models (like in a "battle"). For multi-turn evaluation, a set of models to be evaluated is provided, and the two models used to generate the compared conversations are randomly sampled from this set per comparison (like in an "arena"). These datasets are then uploaded to our evaluation platform, where TFs evaluate each student query alongside its corresponding AI replies, selecting the response they believe is more pedagogically sound or indicating if there is no significant difference. Additionally, TFs can provide written feedback for each comparison, offering valuable insights into the AI's performance.



CS50 Duck Evaluation
0 of 100 items evaluated

Suppose that a student asked this of the duck:

Student
```
for row in rows:
    for str in row:
        if str != 'name':
            value = longest_match(sequence, str)
        longest[str] = value
UnboundLocalError: cannot access local variable 'value' where it is not associated with a value
```

Which response is better, pedagogically?

Response 1:

ddb
It looks like the error is because the variable `value` is being used before it is assigned a value. Can you think of a way to ensure `value` is always initialized before it is used in your code?

Response 2:

ddb
It looks like you're encountering an `UnboundLocalError`. This error typically occurs when you try to use a variable before it has been assigned a value.

Can you explain what you want your code to do and what specific part you're having trouble with?

No Difference

Figure 1: Interface for comparing single-turn AI responses, enabling TFs to review and provide feedback on the CS50 Duck's performance.

For single-turn evaluation, automated assessments are also generated using another AI model [24]. These model-graded evaluations of the same dataset are compared with the human-graded choices. Once validated, the model-graded evaluations can be applied to larger datasets of queries, significantly increasing the evaluation scale while reducing the need for human labor.

3.2 Emulating Human Teacher Responses

To ensure the CS50 Duck generates responses akin to human educators, we focused on aligning its behavior with effective teaching strategies using few-shot prompting and fine-tuning on curated data.

3.2.1 Few-shot prompting. As a first step towards this approach, we adopted a few-shot prompting technique. Few-shot prompting, a method in natural language processing (NLP), involves providing the language model with a few example interactions through the system prompt or through an appended history of interactions before generating a response, which helps the model align its replies with the desired context and style [11][13].

3.2.2 Fine-tuning Model. However, there are some limitations to the few-shot prompting approach. For one, since the examples have to be provided to the LLM for every interaction, the size of the input increases proportionally with the number of examples provided, which means adding more than a few can be costly and exacerbate instruction dilution. To address this challenge, we propose fine-tuning a model specific to the purpose of a teaching assistant. Fine-tuning involves adjusting the parameters of a pre-trained model to imitate the behavior of task-specific training data, which in our case consists of student-tutor conversations.

4 Implementation Details

We focus on improving the CS50 Duck's effectiveness and alignment with pedagogical goals through three primary strategies: incorporating few-shot prompting, fine-tuning the model, and utilizing a model evaluation framework.

4.1 Fine-Tuning Dataset

To improve the model's ability to provide contextually relevant, human-like responses, we constructed a dataset consisting of 50 carefully curated examples of student-tutor conversations for the AI to imitate. Of these, seven examples were multi-turn conversations, each containing up to five student-tutor interactions.

The dataset was designed to encompass a diverse set of student needs and employed teaching strategies supported by feedback from previous TFs, ranging from requests for help with code generation (for which the CS50 Duck was expected to tactfully decline) to debugging questions (for which the goal was for the duck to guide students toward identifying and resolving issues on their own). Additionally, the dataset included cases focused on enhancing general understanding, for which we expected the duck to provide informative answers to support learning. Capturing the subtle variations in these interactions proved challenging with few-shot prompting alone, as it is difficult to encode such nuanced responses within the confines of a single system prompt. However, the slightly larger, fine-tuned dataset allowed us to better reflect the varied instructional strategies and adaptive tones necessary for effective tutoring.

4.2 Human and Model-Graded Evaluation

We employed a dual evaluation framework combining automated evaluations with human-in-the-loop assessments.

(1) **Human-in-the-Loop Evaluation:** Human-graded evaluation involve iterations of TFs reviewing AI-generated responses, selecting their preferred responses, and providing detailed feedback for the AI system's maintainers. On our evaluation platform, administrators set up evaluation tasks and assign them to TFs for evaluation, who will compare how different models respond to the same student message and choose which they prefer.
(2) **OpenAI's Eval Framework:** We utilized OpenAI's Eval framework to automate this evaluation, which uses an LLM to answer open-ended questions about AI-generated responses. The framework allows for customized evaluation templates tailored to our educational context [31, 34]. By using a customized version of OpenAI's "Battle" template, which compares two responses against some subjective standard, we use GPT-4 to compare responses to determine which is more pedagogically sound. Our template starts with the question: "Which response is from the better tutor?"

5 Results

5.1 Diagnosing Problems with Evaluation

In Summer 2024, we created a dataset comprising 50 student queries submitted to CS50 Duck over the past year to test our evaluation framework and potential improvements to our system prompt. The dataset was broken down as follows: 15 queries on code generation, 15 on debugging, 10 on error messages, 5 on introductory coding concepts, and 5 on conceptual CS questions. This distribution reflected the typical questions posed to CS50 Duck while emphasizing areas where we sought improvement.

We evaluated the initial AI-generated response to each query using two distinct versions of our system prompt:

- **V0:** The current system prompt used by CS50 Duck.
- **V1:** An altered prompt aimed at guiding students more interactively by posing leading or clarifying questions.

For instance, V1 includes additional instructions such as: *"If the student asks you a question about their code, do not diagnose errors or provide future steps, but rather provide some encouragement and ask them leading questions or hints that could help them diagnose the error or find the next steps by themselves."* V1 does not include any few-shot prompting or fine-tuning.

5.1.1 Human Evaluation. We asked TFs from previous iterations of the course to complete the evaluation. We acquired data from 29 TFs, 24 of whom completed all 50 comparisons. Among them, 18 had at least two semesters of experience on staff, meaning they taught both before and after the introduction of the CS50 Duck.

In total, 1,309 comparisons were assessed, with 801 by TFs with at least two semesters of experience. We present the results in Table 2.

Table 2: The distribution of choices made in the TF-graded evaluation reveals a split between preference for V0 and V1, suggesting areas for improvement in both. TFs with at least two semesters of experience showed more preference for V1.

	All Surveyed TFs			Surveyed TFs (2+ Semesters)		
	V0	V1	No Diff.	V0	V1	No Diff.
Generate Code	41%	54%	5%	38%	58%	4%
Debug Code	53%	46%	2%	46%	52%	3%
Error Messages	46%	47%	7%	40%	53%	7%
Intro Coding	33%	46%	21%	35%	45%	19%
Conceptual	26%	27%	46%	29%	30%	41%

We observe roughly equal instances of TFs preferring V0 over V1 and vice versa, with few instances of "No Difference." TFs generally preferred V1 for code generation and introductory coding questions. Among TFs with at least two semesters of experience, the preference for V1 was more pronounced, suggesting that teaching experience prior to the introduction of CS50 Duck negatively correlated with favoring a more helpful bot.

These evaluation results quantitatively confirmed issues that TFs have anecdotally reported with the current V0 system prompt over the past year. TFs who preferred V1 in certain comparisons remarked:

- *"I feel like if a code snippet is given [like in V0], students will not read or think critically at all. In general I am anti-giving these snippets, and in this case [V1] forces the student to think and understand the code."*

- *"The habit of understanding and translating error messages is important! [V0] digests the meaning of the error and applies it for the student. [V1] is better; it merely provides context, and prods for more."*
- *The "CS50 Duck does the work of articulating what the code does for the student in [V0]."*

Furthermore, the variation in TF preferences between V0 and V1, even within the same category or for the same student query, underscores an overall dissatisfaction with both versions. TFs who chose V0 over V1 critiqued that:

- *"[V0] is basically a rephrase, but it's more considerate to the first ask. This can help during points of stress."*
- *"[V0] gives too much help in one go! As an evaluator, I cannot choose [V1] since I don't know the follow-up help it gives after the second input. This isn't a 'help start the problem!' sort of question so the CS50 Duck should at least give a direction or little diagnosis, I believe."*

These comments highlight the need for a model that combines the strengths of both versions. While V1 is a step in the right direction, it requires adjustments to adopt a more considerate tone and provide support tailored to each situation. Our results demonstrate that systematic evaluations can reveal issues that tool developers might initially overlook and enable the creation of more tailored, pedagogically-aligned AI tutors. We can now incorporate these insights from the evaluation into future versions of the CS50 Duck.

The shortcomings of V1 also underscore the limitations of relying solely on the system prompt to shape LLM behavior, suggesting the necessity of incorporating few-shot prompting or fine-tuning to achieve further improvements.

5.1.2 Model-Graded Evaluation. We subsequently conducted our customized evaluation using OpenAI's Eval framework, running it 10 times on the same dataset. For each comparison, we selected the most frequent response across runs as the model's final choice, splitting the choice uniformly in case of ties. The model chose V1 in 25.5 cases, V0 in 15 cases, and "No Difference" in 9.5 cases out of the 50 student queries. The model demonstrated high internal consistency, making the same choice in 8 or more runs for 44 queries and producing identical choices across all 10 runs for 34 queries.

We assessed consistency with human decisions by determining whether the model's choice for each query aligned with the development team's choice, the top choice of all TFs, or the top choice of TFs with at least two semesters of experience. The model was consistent with human preferences 82% of the time, indicating strong alignment with human evaluators. Upon examining the reasoning provided in both human and model evaluations, we found that in all consistent results, the model's justifications generally mirrored human rationale. These evaluations highlighted that V0 tended to offer assistance too readily, while V1 often failed to provide sufficient support.

5.2 Confirming Improvement with Evaluation

We introduced two new versions of the CS50 Duck, incorporating additional modifications to the system prompt used in V1:

- **V2:** Few-shot prompting with 4 example interactions included in the system prompt.

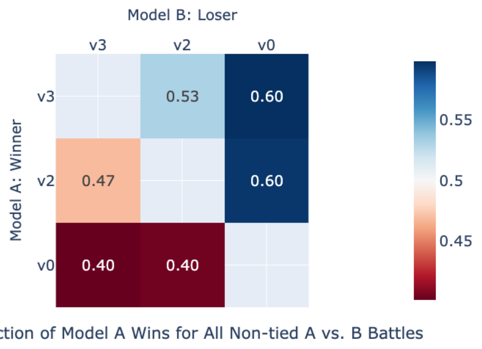


Figure 2: The win rates of the multi-turn evaluation show that TFs preferred conversations generated by the models with few-shot prompting and fine-tuning over the original version (V0) 60% of the time.

- **V3:** A fine-tuned model using GPT-4o-mini [33], trained on 50 example conversations over 5 epochs, or the number of times the training algorithm iterates through the dataset.

At the time of the experiment, fine-tuning was available only for GPT-4o-mini. To ensure the evaluation focused on differences in our strategies rather than disparities between underlying OpenAI models, we used GPT-4o-mini as the base model for all versions in the comparison.

In Fall 2024, we then created multi-turn evaluation tasks using V0, V2, and V3. These evaluations were conducted with participation from 72 current TFs at Harvard University and Yale University, 40 of whom were returning staff members. Across all participants, a total of 480 comparisons were made, and the results are shown in Figure 2. The win rates from pairwise model comparisons—measured as the fraction of times model A is chosen when compared with model B—indicate that both V2 and V3 are more likely to be chosen over V0, with V3 showing a slight edge over V2.

To further analyze the results, we employed the open-source code of ChatBot Arena [3] to create an Elo rating system [7] for our models. The Elo rating system, originally designed for ranking players in dynamic games such as chess, has been increasingly used for LLM evaluation [1]. Despite having only 480 data points, the Elo rates clearly ranked V0 lower than V2 and V3. While the estimated Elo score for V3 was slightly higher than that of V2, the current data was insufficient to narrow the confidence intervals enough to eliminate overlap between them. Moving forward, since V0 consistently ranked lower, we plan to prioritize future evaluations between V2 and V3 to refine them further.

Remarks left by TFs during the evaluations provided insights into the reasoning behind their preferences:

- *(Comparing with V0) "[V2] actually makes me talk to it and figure it out myself."*
- *(Comparing with V0) "Both a bit overexplanatory, but [V2] more teacherly."*
- *"[V3] attempts to walk the students through the code. I think it is slightly better than [V0], because [V0] reveals too much information, all at once."*

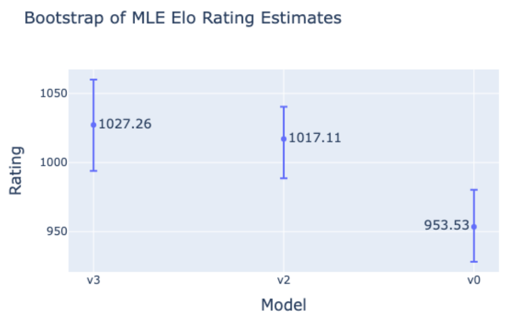


Figure 3: The estimated Elo score of V0 was the lowest, with its 95% confidence interval showing no overlap with the confidence intervals of V2 and V3, ranking it last.

- *"I think [V3] is better for the students to actively learn, but [V0] seems to have better feedback."*

While this qualitative feedback is informative, it is important to remember that these individual comments apply to specific instances of generated conversations. For example, in some evaluations, V3 appeared overly verbose compared to V0, leading evaluators to note that V3 was occasionally too eager to provide complete answers. However, this behavior was not consistently observed across all conversations generated by V3. Thus, a single piece of qualitative feedback should not be generalized to the entire model but instead used to illustrate human preferences in particular cases. That said, since the evaluated dataset that determined the initial turn of the student-AI conversations was constructed to be representative of past student queries, the overall qualitative feedback should still be broadly generalizable to future student interactions.

5.3 Discussion

While most efforts in training and evaluating LLMs rely heavily on extensive data, our findings align with studies highlighting that a limited amount of high-quality data can achieve satisfactory results and provide conclusive evaluations [21].

The two types of evaluations on our platform offer distinct advantages and drawbacks:

- **Single-turn evaluation** simplifies the process by not requiring the generation of student-like inputs during the evaluation, making it significantly easier and faster for human evaluators. Moreover, it enables model-graded evaluation when results align with human preferences, facilitating efficient assessments on larger datasets.
- **Multi-turn evaluation**, by contrast, more accurately reflects real-world student-AI interactions, providing a deeper understanding of how models function as AI tutors.

While single-turn evaluation is beneficial for AI tutors focused on providing immediate answers, we chose to prioritize multi-turn evaluations. Multi-turn evaluations align more closely with our objective of developing a 24/7 assistant and better reflect how CS50 students interact with the existing CS50 Duck. Although prior research has explored model-graded evaluations in multi-turn contexts [5], our efforts revealed challenges in generating sufficiently realistic student-like queries to sustain meaningful conversations with the AI tutor. Consequently, we concluded that human evaluators—especially those who were recently CS50 students themselves—are best suited for this task.

6 Future Work

For future improvements in CS50's AI tools, we propose exploring ways of reducing costs and expanding the CS50 Duck's capabilities.

To reduce costs associated with using a fine-tuned model, we plan to explore strategies such as prompt routing, which dynamically selects a shorter, most suitable system prompt based on a student's query. Given that the model is already trained to imitate provided conversation patterns, it can behave according to expectations without extensive system prompts, thereby lowering token usage and related expenses. We believe that carefully implementing this approach with a fine-tuned model could significantly decrease costs while maintaining high quality interactions.

Finally, we plan to tailor the CS50 Duck's responses based on individual student proficiency and learning styles. By incorporating student profiles and learning analytics, the AI can provide personalized, contextually appropriate guidance, ensuring students receive the right level of challenge and support [14].

7 Conclusion

Over the past two iterations of CS50, the integration of AI tools in the course has significantly enhanced student learning by providing personalized and immediate assistance. However, through the evaluation of the CS50 Duck's responses, we identified key areas needing improvement, including instruction dilution, ineffective teaching styles, and a lack of a systematic way to evaluate our AI system. Following the implementation of newer versions with this feedback in mind through few-shot prompting and fine-tuning, we have quantitatively demonstrated evidence of improvement in model behavior and qualitatively shown alignment with our teaching team's pedagogical values of TF-like support that prioritizes long-term learning.

Our evaluation framework has demonstrated its effectiveness in improving the relevance and quality of AI responses. Model-graded evaluations allow for scalable assessment of large datasets, while human-in-the-loop evaluations confirm the alignment of AI outputs with pedagogical objectives and allow for evaluating multi-turn conversations. These advancements highlight the potential of AI tools to offer scalable and personalized learning support.

The success of these tools underscore their potential for broader application in education. As we continue to refine these technologies, our goal remains to enhance student learning experiences and ensure that AI tools effectively support educational goals.

Acknowledgments

Many thanks to Ed, GitHub, Microsoft, and OpenAI for their support of this work. And many thanks as well to Brenda Anderson, Sophie Anderson, Doug Lloyd, and CS50's whole team for their assistance with this work.

References

[1] Meriem Boubdir, Edward Kim, Beyza Ermis, Sara Hooker, and Marzieh Fadaee. 2023. Elo Uncovered: Robustness and Best Practices in Language Model Evaluation. arXiv:2311.17295 [cs.CL] https://arxiv.org/abs/2311.17295

[2] Jérôme Seymour Bruner. 1961. The act of discovery. *Harvard Educational Review* 31 (1961), 21–32. https://api.semanticscholar.org/CorpusID:142938071

[3] Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024. Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference. arXiv:2403.04132 [cs.AI] https://arxiv.org/abs/2403.04132

[4] Chroma. 2024. Chroma DB. https://docs.trychroma.com/ Accessed: 2024-07-21.

[5] Haodong Duan, Jueqi Wei, Chonghua Wang, Hongwei Liu, Yixiao Fang, Songyang Zhang, Dahua Lin, and Kai Chen. 2023. BotChat: Evaluating LLMs' Capabilities of Having Multi-Turn Dialogues. arXiv:2310.13650 [cs.CL] https://arxiv.org/abs/2310.13650

[6] Ed. 2024. https://edstem.org/ Accessed: 2024-07-21.

[7] A.E. Elo. 1966. *The USCF Rating System: Its Development, Theory, and Applications*. United States Chess Federation. https://books.google.com/books?id=onUazQEACAAJ

[8] Harry Barton Essel, Dimitrios Vlachopoulos, Albert Benjamin Essuman, and John Opuni Amankwa. 2024. ChatGPT effects on cognitive skills of undergraduate students: Receiving instant responses from AI-based conversational large language models (LLMs). *Computers and Education: Artificial Intelligence* 6 (2024), 100198. https://doi.org/10.1016/j.caeai.2023.100198

[9] Denny et al. 2024. Desirable Characteristics for AI Teaching Assistants in Programming Education. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2024)*. ACM. https://doi.org/10.1145/3649217.3653574

[10] Finnie-Ansley et al. 2022. The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. In *Proceedings of the 24th Australasian Computing Education Conference* (Virtual Event, Australia) *(ACE '22)*. Association for Computing Machinery, New York, NY, USA, 10–19. https://doi.org/10.1145/3511861.3511863

[11] Garcia et al. 2023. The Unreasonable Effectiveness of Few-shot Learning for Machine Translation. In *Proceedings of the 40th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 10867–10878. https://proceedings.mlr.press/v202/garcia23a.html

[12] Ji et al. 2023. Survey of Hallucination in Natural Language Generation. *Comput. Surveys* 55, 12 (March 2023), 1–38. https://doi.org/10.1145/3571730

[13] Jared Kaplan et al. 2020. Scaling Laws for Neural Language Models. arXiv:2001.08361 [cs.LG] https://arxiv.org/abs/2001.08361

[14] Khor et al. 2024. A Systematic Review of the Role of Learning Analytics in Supporting Personalized Learning. *Education Sciences* 14, 1 (2024). https://doi.org/10.3390/educsci14010051

[15] Lewis et al. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 9459–9474. https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf

[16] OpenAI et al. 2024. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL] https://arxiv.org/abs/2303.08774

[17] Tom B. Brown et al. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165 [cs.CL] https://arxiv.org/abs/2005.14165

[18] Xingjiao Wu et al. 2022. A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems* 135 (2022), 364–381. https://doi.org/10.1016/j.future.2022.05.014

[19] Wensheng Gan, Zhenlian Qi, Jiayang Wu, and Jerry Chun-Wei Lin. 2023. Large Language Models in Education: Vision and Opportunities. arXiv:2311.13160 [cs.AI] https://arxiv.org/abs/2311.13160

[20] Jennifer M Gore. 2021. The quest for better teaching. *Oxford Review of Education* 47, 1 (2021), 45–60.

[21] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. 2023. Textbooks Are All You Need. arXiv:2306.11644 [cs.CL] https://arxiv.org/abs/2306.11644

[22] A.M. Hasanein and A.E.E. Sobaih. 2023. Drivers and Consequences of ChatGPT Use in Higher Education: Key Stakeholder Perspectives. *European Journal of Investigation in Health, Psychology and Education* 13, 11 (Nov 2023), 2599–2614. https://doi.org/10.3390/ejihpe13110181

[23] Majeed Kazemitabaar, Runlong Ye, Xiaoning Wang, Austin Zachary Henley, Paul Denny, Michelle Craig, and Tovi Grossman. 2024. CodeAid: Evaluating a Classroom Deployment of an LLM-based Programming Assistant that Balances Student and Educator Needs. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24)*. ACM, 1–20. https://doi.org/10.1145/3613904.3642773

[24] Simon Lermen and Ondřej Kvapil. 2023. Exploring the Robustness of Model-Graded Evaluations and Automated Interpretability. arXiv:2312.03721 [cs.CL] https://arxiv.org/abs/2312.03721

[25] Jiaqi Li, Yixuan Tang, and Yi Yang. 2024. Know the Unknown: An Uncertainty-Sensitive Method for LLM Instruction Tuning. arXiv:2406.10099 [cs.CL] https://arxiv.org/abs/2406.10099

[26] Rongxin Liu, Carter Zenke, Charlie Liu, Andrew Holmes, Patrick Thornton, and David J. Malan. 2024. Teaching CS50 with AI: Leveraging Generative Artificial Intelligence in Computer Science Education. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (Portland, OR, USA) *(SIGCSE 2024)*. Association for Computing Machinery, New York, NY, USA, 750–756. https://doi.org/10.1145/3626252.3630938

[27] Microsoft. 2023. Azure OpenAI. https://azure.microsoft.com/en-us/products/ai-services/openai-service/ Accessed: 2024-07-21.

[28] Microsoft. 2024. Introduction to prompt engineering. https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/prompt-engineering Accessed: 2024-07-21.

[29] OpenAI. 2024. ChatGPT. https://chatgpt.com Accessed: 2024-11-17.

[30] OpenAI. 2024. Embeddings. https://platform.openai.com/docs/guides/embeddings Accessed: 2024-07-21.

[31] OpenAI. 2024. Getting Started with OpenAI Evals. https://cookbook.openai.com/examples/evaluation/getting_started_with_openai_evals Access: 2024-07-22.

[32] OpenAI. 2024. GPT-4o. https://openai.com/index/hello-gpt-4o Access: 2024-07-21.

[33] OpenAI. 2024. GPT-4o Mini. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence Access: 2024-07-21.

[34] OpenAI. 2024. OpenAI Evals. https://github.com/openai/evals Access: 2024-07-21.

[35] E. Soloway. 1986. Learning to program=learning to construct mechanisms and explanations. *Commun. ACM* 29, 9 (sep 1986), 850–858. https://doi.org/10.1145/6592.6594

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention Is All You Need. arXiv:1706.03762 [cs.CL] https://arxiv.org/abs/1706.03762

[37] Hiroko Kawaguchi Warshauer. 2011. The Role of Productive Struggle in Teaching and Learning Middle School Mathematics.

[38] Boštjan Šumak, Diego López-De-Ipiña, Olga Dziabenko, Sĕrgio Duarte Correia, Luĭsa M. Serrano De Carvalho, Secundino Lopes, Irfan Şimşek, Tuncer Can, Darja Ivanuša Kline, and Maja Pušnik. 2024. AI-Based Education Tools for Enabling Inclusive Education: Challenges and Benefits. In *2024 47th ICT and Electronics Convention, MIPRO 2024 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., 472–477. https://doi.org/10.1109/MIPRO60963.2024.10569714