Int J Artif Intell Educ
https://doi.org/10.1007/s40593-025-00505-6

ARTICLE

Generating In-Context, Personalized Feedback for Intelligent Tutors with Large Language Models

Jennifer M Reddig[1] · **Arav Arora**[1] · **Christopher J. MacLellan**[1]

Accepted: 10 July 2025

Abstract

This study explores how large language models (LLMs), specifically GPT-4, could be used to generate personalized feedback within an Intelligent Tutoring System (ITS). The research focuses on evaluating the model's ability to (1) diagnose student errors, (2) generate personalized corrective feedback, and (3) assess the accuracy of diagnoses and helpfulness of the feedback. We analyze student errors from the Apprentice Tutor College Algebra ITS and prompt GPT-4 to give targeted feedback on those errors. The findings suggest that while this model can effectively diagnose a range of student errors, its feedback varies in effectiveness based on the complexity of the problem and the type of error. While GPT-4 generates relevant, specific feedback a majority of the time, 35% of the hints were too general, incorrect, or give away the correct answer. The study also explores methods for using an LLM to automatically evaluate the validity of generated feedback, and finds that only 35% of feedback passes automated helpfulness evaluations.

Keywords Immediate feedback · Generative AI · Large language model · Educational technology

Introduction

Personalized feedback is an important component of effective learning, particularly in educational technology like Intelligent Tutoring Systems (ITS). Students benefit from tailored guidance that addresses their unique challenges (Shute, 2008). However, providing such feedback at scale has been challenging due to the high cost and time required to craft error-specific feedback.

✉ Jennifer M Reddig
jreddig3@gatech.edu

Arav Arora
aarora@gatech.edu

Christopher J. MacLellan
cmaclell@gatech.edu

[1] Georgia Institute of Technology, North Avenue, Atlanta, GA 30332, USA

Published online: 25 July 2025

Current ITS frameworks typically depend on pre-defined, rule-based responses. This can limit the system's flexibility to adapt beyond the defined rules to respond to the full range of student errors. Creating these rule-based responses is time-intensive and limits the scalability of these systems. While a human tutor could offer specific, response-dependent feedback, these tutoring systems can lack robust adaptation to student errors.

The creation of large language models (LLMs) offers a promising solution to these scalability issues. LLMs, with their ability to generate natural language responses based on a wide range of inputs, present an opportunity to provide scalable, personalized feedback in real-time. Recent studies demonstrate promise for using LLMs to tutor students directly (Butgereit et al., 2023; Butgereit, 2024; Shahri et al., 2024) with Khan Academy's Khanmigo (Yamkovenko, 2023) as the poster child for AI tutoring. However, the reliability of Generative AI for tutoring remains uncertain, as these models often produce misleading and incorrect information. Reliable quality assurance measures would be required to separate accurate and beneficial feedback from misleading instructions. Perhaps by embedding LLMs into ITS, we can achieve the best of both worlds – proven tutoring practices and 100% accurate answers with responsive, personalized, natural language wrong-answer feedback.

This paper presents a series of studies that explore the potential of LLMs to craft real-time feedback within intelligent tutors for adult math learners. By analyzing student transaction data from a College Algebra class, we identify distinct types of student errors, investigate GPT-4's ability to recognize and respond to student mistakes, and compare human and AI evaluation of GPT-generated feedback. This research also examines whether LLMs can serve as a quality control mechanism, autonomously evaluating the helpfulness and informativeness of the feedback they generate. Specifically, this research centers around three research questions:

RQ1: Can an LLM diagnose student errors? If so, what types of errors are LLMs especially adept at identifying?
RQ2: Can an LLM generate helpful and informative corrective feedback?
RQ3: Can an LLM be used to evaluate the helpfulness and informativeness of feedback?

For a well-structured subject like College Algebra, that is likely well-represented in the training data, we find that GPT-4 can diagnose student errors with approximately 80% accuracy. However, this accuracy decreases when student responses contain more than one error. While the LLM generates mostly relevant, specific feedback, 35% of the generated hints are too general, incorrect, or give away the correct answer. Ensuring that misleading hints are never presented to students requires robust quality control mechanisms. We also examined if LLMs could be used to identify quality feedback. Although LLMs have shown success in simulating student interactions (Lu & Wang, 2024), when applied to real student data, GPT-generated feedback only pass simulated student helpfulness evaluations approximately 35% of the time.

Achieving reliable error diagnosis, feedback generation, and autonomous evaluation with large language models could significantly reduce the workload of ITS authors and developers by minimizing manual input. Existing systems would be able to respond in a student-specific way to even the most unusual of student actions. This

could result in a more scalable tutoring system that adapts in real-time to individual student errors, vastly expanding the capabilities of personalized instruction.

Background

Learning from Feedback

Feedback is one of the main ways learning technologies guide students in changing their behavior to achieve better learning outcomes. It can close the gap between student performance and desired performance, reduce the cognitive load for novices, and support correction of misconceptions and errors (Shute, 2008). Feedback is more effective when it is contingent on the learner's response (Gilman, 1969; Mandernach, 2005). Verification feedback tells the learner if their response is correct or incorrect and is effective for error correction (Marsh et al., 2012). Elaborative feedback goes beyond verification feedback to address the topic, response, or error; this can guide students toward a correct answer or help them understand why an answer is correct. McKendree (1990) found that a specific type of elaborative feedback – goal-oriented feedback – can help students self-correct and is more effective for learning than explaining when a rule has been violated.

Elaborative feedback outperforms verification feedback at enhancing learning (Enders et al., 2021; Olms et al., 2016). It improves conceptual learning (Finn et al., 2018) more than verification feedback, and learners perform better on transfer tasks (Butler et al., 2013). However, creating elaborative feedback can be time-consuming and costly (VanLehn, 2006). For the feedback to be contingent on the learner's response, intelligent tutoring systems need to anticipate possible types of student response and prepare targeted feedback in advance. Prior ITS efforts like the Help Tutor aimed to improve self-regulated help-seeking behavior (Aleven et al., 2016), yet found that better hint usage did not always lead to improved learning, highlighting the need for more personalized hinting strategies.

Automated Feedback

Most instructional systems offer some form of automated feedback. According to Deeva et al. (2021), they deliver immediate, corrective feedback based on student errors. The most common strategy for generating feedback is to highlight differences between student and expert responses and to deliver pre-written, rule-based feedback. Considerable time must be spent brainstorming common mistakes and crafting feedback to address each error. Model tracing (Anderson & Pelletier, 1991), intention-based analysis (Johnson, 1990), text mining, and other response analysis techniques can recognize many kinds of misconceptions and fill in the blanks of pre-defined feedback messages to create relevant and adaptive hints. While this feedback is often robust, informative, and responsive to learner actions, it is rarely generated from scratch in real time.

The Hint Factory (Stamper et al., 2008) is an example of a technique that can generate new hints on-demand. It uses prior student data to create a state space of

partial solutions, that is continually updated as students interact with the system. By mapping a student's current partial solution into the state space, the Hint Factory can guide them towards an appropriate next action. For student solutions that cannot be mapped to a state, some Hint Factory extensions create edit-based hints describing the changes to be made to turn the solution into a more complete reference solution (Rivers & Koedinger, 2017). The Continuous Hint Factory (Paaßen et al., 2018) applies edits to the state space to account for large, sparse spaces. These methods, which do describe *how* to proceed, do not often include *why* or provide elaborative feedback.

Now that LLMs are widely available and can deliver human-like responses to a wide range of inputs, many researchers are exploring their use for generating educational feedback. LLMs have been used to explain correct answers (Kumar et al., 2023), generate static hints (Pardos & Bhandari, 2024), guide the problem solving process (Pal Chowdhury et al., 2024), and create distractor answers (McNichols et al., 2023) for math content. Within programming tutors, LLMs have been used to create learner-contingent next-step hints to help learners correct mistakes and progress toward a complete answer (Xiao et al., 2024; Roest et al., 2024). Recent work reframes LLMs not just as sources of static hints, but as pedagogical collaborators (Mollick & Mollick, 2024). Most evaluations find that LLM hints are appropriate, specific, and personalized, and in some cases, students prefer LLM-generated feedback to human-written feedback (Wan & Chen, 2024). However, to our knowledge, there is not yet any evidence that LLMs can provide *personalized mathematical corrective feedback*.

While LLMs look quite promising for generating natural language feedback, studies have shown they are limited in their ability to reason (Berglund et al., 2023; Gendron et al., 2023), answer mathematics questions (Frieder et al., 2024; Dziri et al., 2024), or create a plan (Valmeekam et al., 2022, 2023), often generating incorrect or made-up facts (i.e., hallucinating) (Lee et al., 2018). LLM-generated text needs to be verified for accuracy, especially before passing it on to learners. Intelligent Tutoring Systems can provide a grounding mechanism to help LLMs overcome these limitations by including structured information in the prompt. When ITS data, such as the expected answer and the solution steps, is used within the LLM's prompt, the model does not need to predict mathematical answers, which reduces the risk of factual inaccuracy. Additionally, ITSs have a structured layout and step sequence that guide the problem-solving process. The LLM can fit within this framework and offer partial guidance between steps which ensures that the generated feedback aligns with the tutoring system's pedagogical framework. By combining the robust generative power of LLMs with the structured, expert-driven framework of ITSs, we aim to create feedback that is accurate, reliable, pedagogically sound, and aligned with the individual learner's progress, resulting in a more effective and trustworthy educational tool.

Tutoring Systems

Intelligent Tutors

These educational technologies are designed to provide personalized instruction and feedback, adapting to students' learning needs in real-time. Notable systems for mathematics include the Cognitive Tutor (Ritter et al., 2007) (now known as MathIA),

ASSISTments (Heffernan & Heffernan, 2014), and OATutor (Pardos et al., 2023). These systems use data from student interactions to deliver tailored feedback and guidance. By providing hints, explanations, and step-by-step solutions, ITSs aim to enhance learning outcomes by mirroring one-on-one human tutoring. In addition to math, ITSs have been successfully applied in other domains, such as language learning (Tafazoli et al., 2019), science (Azevedo et al., 2009; Rau et al., 2015), and computer programming (Nesbit et al., 2014).

Apprentice Tutors is a web-based tutoring platform covering several mathematics topics and problem types within those topics. This platform provides the context and data for our current work. Tutors for each problem type were built by examining student thinking and learning through cognitive task analyses (Lovett, 1998). Every step in each tutor maps to a single skill, which is represented in the expert model by a production rule. The platform gives immediate step-level correctness feedback to students, with multiple levels of hints for each step available upon request. The first level hint is an orientation hint which directs students to the appropriate field to fill out next. The second level hint is an instrumental hint, a description of the math operation that is needed to satisfy the field. The third hint is a bottom-out hint that gives the exact correct answer to enter. The student can copy this answer to satisfy the field, essentially turning the problem into a worked example.

Tutor Authoring

Building tutors is challenging and authoring tools enable educators and developers to create custom ITSs without needing extensive programming expertise. Tools like the Tutor Development Kit (TDK) (Anderson & Pelletier, 1991), CTAT (Cognitive Tutor Authoring Tools) (Aleven et al., 2009), and the ASSISTment Builder (Razzaq et al., 2009) simplify the process of designing interactive, adaptive learning environments. These platforms allow instructors and developers to build tutoring content, define problem-solving rules, and create hint structures. However, crafting a cognitive model that supports contextualized feedback still requires significant time and expertise (VanLehn, 2006).

The Apprentice Tutor Builder (ATB) (Smith et al., 2024) is a platform that simplifies tutor creation. Using a drag-and-drop interface builder, instructors can build a tutor layout without any knowledge of web development or programming languages. ATB also uses a large language model to let instructors draft initial interface designs by providing a problem description (Calo & Maclellan, 2024). Following interface creation, ATB builds on prior work (MacLellan & Koedinger, 2022; Weitekamp et al., 2020) to let instructors build an expert model by interactively providing demonstrations and feedback directly in their tutor interface. First, the instructor solves a problem in the tutor. From these demonstrations, ATB induces an expert model that can replicate their actions. Next, the instructor gives ATB new problems and observes its attempt to solve each problem using its expert model. If ATB takes incorrect actions, then the instructor flags these and the system updates its expert model to reflect this feedback. Whenever ATB does not know what to do next, it asks the instructor for additional demonstrations and the process continues. Through interface creation and expert model training, instructors can build and deploy their own intelligent tutors.

At the moment, ATB has no method for learning what feedback is appropriate nor when it is relevant. Most production rule tutors have 'bug' rules, or common incorrect answers that receive specialized feedback. Bug rules need to be specially crafted to trigger upon receiving certain kinds of answers. In fact, every piece of feedback in a tutor needs to be human authored and the tutor needs to know when to display such feedback. This functionality can be incredibly time consuming to create. To address this challenge, we explore whether generative AI can serve as a tool for simplifying and accelerating feedback creation, making it easier to generate timely, targeted hints for learners.

Study 1: Diagnosing Student Errors

In order to decide what feedback to deliver, ITS must first identify the problem with a student's response. Traditional ITS rely heavily on pre-defined bug rules to identify common student errors and provide targeted feedback. While effective, these rules can be inflexible and often fail to capture the full range of errors that students may present. As bug rules become more robust and complex, crafting rules by hand and writing feedback messages requires significant human effort, which limits scalability and adaptivity. This study focuses on diagnosing student errors, with the intention of crafting feedback that addresses those specific errors.

To address these limitations, we explore whether an LLM can accurately diagnose student errors, especially when they fall outside the range of easy identification. By enabling ITS to use an LLM for error identification, we aim to augment the current rule-based approach. When an error does not trigger any of the existing bug rules, the ITS could pass feedback creation off to an LLM to identify what the student did wrong and generate an appropriate hint. If successful, this approach could enable adaptive error diagnoses within ITS, reducing the need for manual rule-coding and feedback authoring while supporting more personalized feedback for students.

We start by analyzing the types of errors students make within the ITS, Apprentice Tutors, which allows us to understand common challenges and patterns in responses. By examining specific error categories – such as logical mistakes, syntax errors, and incomplete responses – we can assess the capabilities of LLMs to accurately identify different kinds of errors. Understanding these distinctions will help us evaluate the potential for LLMs to complement existing bug rules.

Methods

Categorizing Errors from Transaction Data

The Technical College System of Georgia used the Apprentice Tutor platform in several College Algebra classes during the Spring 2024 semester. Students had access to tutors on factoring polynomials(example interface in Fig. 1), exponent rules, operations with radicals (example interface in Fig. 2), solving rational equations, properties of logarithms, characteristics of quadratic functions (example interface in Fig. 3), solving quadratic equations, solving exponential equations, and solving logarithmic

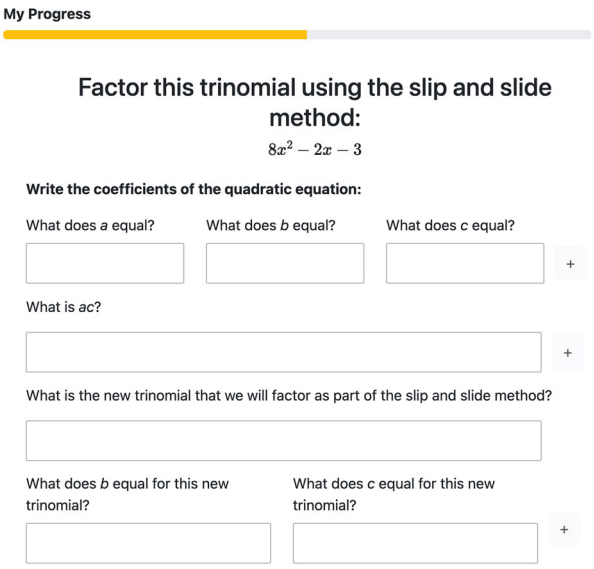

Fig. 1 The interface for Factoring using the Slip and Slide method. The textual instructions are also very descriptive

equations. Each tutored topic has several different problem types. Each problem type contains multiple steps that students must complete in sequence. Each step corresponds to a knowledge component (KC) and Bayesian Knowledge Tracing (BKT) (Corbett & Anderson, 1994) assess the student's mastery of each KC.

As students interacted with the tutors, the Apprentice Tutor platform automatically logged each action they performed, resulting in 6926 tutor transactions. Of these

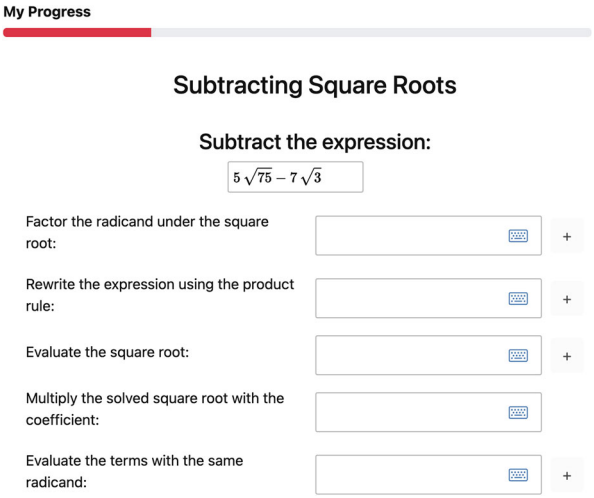

Fig. 2 The interface for Subtracting Square Roots tutor. Notice that the instructions for each field are quite descriptive

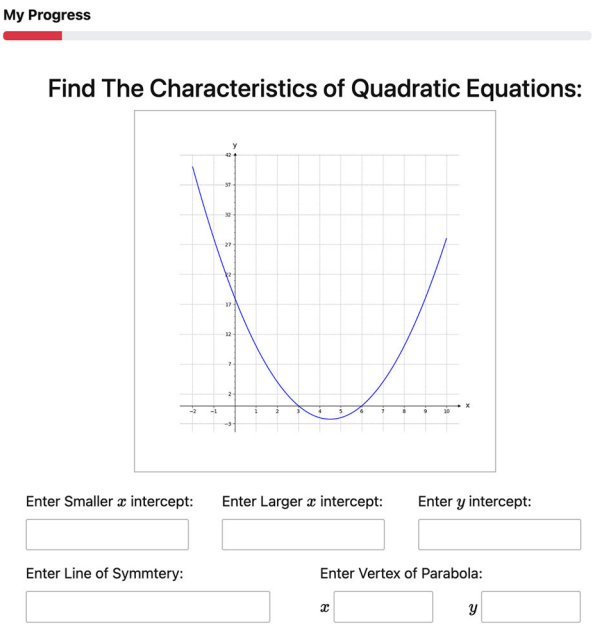

Fig. 3 The interface for identifying parts of a quadratic function. The students are given a picture of the graph and asked to extract information

transactions, 1307 were incorrect answers. The full distribution of transaction data is described in Table 1. We excluded the errors from the exponential equations tutor in our analysis because there were only 4 student errors.

Our team took an inductive approach (Thomas, 2003) to analyzing student errors. For each incorrect answer, two independent coders compared the student's incorrect response to the correct answer. They described the discrepancy in detail, then hypothesized possible explanations for the discrepancy. In a follow-up round, the coders resolved any disagreements in their diagnosis.

Once the team arrived at an accepted explanation for each student response, we looked for commonalities in the explanations to build our code book. We arrived at five types of errors. The two coders reviewed all errors and classified the attributes

Table 1 Summary of student interactions within AT, detailing the frequency of each action type logged during practice sessions

Action	Count
Start new problem	1563
Correct Answer	1555
Incorrect answer	1307
Incorrect "Done" click	1097
Hint Request	783
Correct "Done" click	374
View study material	218
Tutor walkthrough	1

of each student response according to the five types of mistakes. Through a second round, the two coders resolved any disagreements and reached strong agreement on the entire dataset (Cohen's Kappa 0.931). In cases where disagreements remained, the head coder's classifications were used. Most errors were classified using only one code. Less than 5% of errors were categorized using more than one code. For the full distribution of error types, see Table 2. We now describe each error category.

The first type was a *Logical Error*. These errors involved students performing the incorrect operation or applying the incorrect rule. For example, when simplifying the exponential expression $(47^5)^2$, the student should multiply 2 and 5 to obtain 47^{10} as the correct answer, but instead the student enters $(47)^{25}$, having incorrectly squared the exponent 5.

Another type of error was a *Syntax Error* or *Typo*. These are correct answers that have unexpected characters or non-standard character replacements. The student could enter `-975^{11+2o}`- when the expected answer is `-975^{11 + 20}`-, or `z=-4` when the expected answer is `x=-4`. The student logically followed the correct steps but the format of their answer triggered an incorrect answer.

A third type of error was an *Incomplete* answer. In this situation, the student started to enter an expression, then exited from the field before completing their response. It is difficult to tell from an incomplete response whether the complete response will be correct or not. Incomplete responses include single characters such as (—a single open parenthesis—when the correct answer is $(x + 4)(x - 2)$, or nearly complete answers like $\frac{5}{2}-$ instead of $\frac{5}{2} - \frac{\sqrt{17}}{2}$.

The forth and fifth types of errors are variations on a correct answer. First, we have an answer that would satisfy another field – a *Wrong Field* error. The student has correctly completed another part of the problem-solving process, but entered the answer in the wrong field. Another type is the *Correct Answer* entered into the correct field but because there is an error elsewhere in the problem, the problem is logged as incorrect.

Prior work on classifying student errors arrived at similar categories. Newman (1977) uses categories of Processing (applying the mathematical strategy, akin to our Logical Mistakes) and Encoding (an error in expression like our Syntax Errors, but would also encompass an Incomplete answer). The categories of Comprehension (understanding the written question) and Transformation (identifying an appropriate method for solving) have no equivalent error in our schema. The online tutors describe the appropriate procedure to use to solve the problem (i.e. factor the trinomial using the grouping method) and comprehension is difficult to assess in a short, mostly symbolic

Table 2 Distribution of error types identified in incorrect student responses, categorized by type and displayed as a percentage of total errors logged

Logged Incorrect Transaction Type	Percentage
Logical Mistake	30.1%
Syntax Error	21.2%
Incomplete	15.7%
Wrong Field	19.3%
Correct Answer	18.4%
More than one Error	4.7%

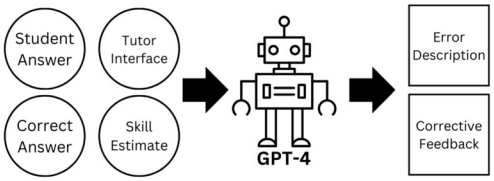

Fig. 4 We incorporated most of relevant information available in the Apprentice Tutors platform into the prompt (See Appendix-A.3), including the student's input, the correct answer, the tutor interface, the skill and skill mastery estimate. GPT-4 first described the student's error then crafted feedback targeted to that mistake

problem to solve. Knifong and Holtan (1976) identifies Clerical and Computation errors. A Computation error would be a Logical Mistake, by applying the wrong operation or a calculation error. Clerical errors would encompass a Typo, putting the right answer in the Wrong Field, and an Incomplete response. These three error types would require sufficiently different corrective feedback so we chose to keep them separate. These prior methods support but do not completely capture student errors in Apprentice Tutors, so we use the five identified categories in further analyses.

Error Diagnosis

With a good sense of the challenges students face in the ITS, we now explore how a Large Language Model could deliver responsive corrective hints. To craft targeted corrective feedback, the author must first identify what mistake the student made. So before asking the LLM to generate feedback, we prompt it to identify the student's error. We drew inspiration from Chain-of-Thought prompting (Wei et al., 2022), which includes intermediate reasoning steps. For more relevant feedback, we first prompted OpenAI's GPT-4[1] using the API to identify the student's error then generate targeted feedback. In this way, we make sure the elaborative feedback is targeted to the student's response, and not just the type of question. For every error, GPT-4 generated a diagnosis and a hint. Though we ask GPT-4 to describe the students error and write feedback in the same prompt, we will analyze the two parts of the response separately. We examine the error descriptions and describe the prompt engineering process in this study and will analyze the feedback in Section "Study 2: Generating Corrective Feedback". The full prompt is detailed in Appendix A.3 and outlined in Fig. 4.

Following diagnosis, we reviewed all the responses to see if they accurately described the difference between the student's answer and the correct answer. We did not ask GPT to use our classification scheme so that the LLM would generate a diagnosis and feedback that addressed the student's specific difference rather than the broader error type. We wanted to see some level of explanation of the student's difference, so responses of the form "*The student incorrectly entered X when they should have entered Y*" were not acceptable diagnoses. A response like "*The student incorrectly entered the numerical value X when they should have entered the equation Y*", would provide enough information to successfully diagnose an error, since feedback that describes the format of the correct answer could be useful to a student. We

[1] gpt-4-0613, accessed August 8, 2024

classified GPT-4's response as either a correct description of the student error or an incorrect description. To answer RQ1, we compiled the LLM's accuracy into the five categories of errors.

We found that without the full context of the problem, GPT-4 was less successful at describing student errors on tutors that had more fields, more sub-problems, and a longer problem-solving process. We performed three rounds of prompt engineering on the factoring tutors to evaluate the appropriate amount of context to include in the prompt since these tutors contained the most sub-steps. Once GPT-4's performance was greater than 80% across all error types, we applied this prompt to the rest of the tutoring content. Table 3 demonstrates how, with additional context to situate the student's answer within the appropriate sub-problem, GPT-4's accuracy increased on a complex tutoring environment like factoring. With only the problem, student answer, and correct answer (see Appendix A.1), GPT-4 questioned whether a correct answer was indeed correct 21% of the time. Even though the correct answer was included in the prompt, we often saw responses like "*There seems to be an error in the problem, as the correct answer provided doesn't match the problem statement. The correct answer provided couldn't be the answer to the given problem.*"

When we added the complete tutor interface (see Appendix A.2), GPT-4 identified a correct answer as correct 100% of the time. However, some responses from GPT-4 described out-of-order problem solving instead of identifying the logical difference from the correct answer, such as "*The student entered a value in the factor_1_ac_2 field without first identifying the values of a, b, and c, as well as ac.*" To reach this point in the problem-solving process, the student necessarily must have completed the prior steps correctly. We added one more piece of context to the prompt – the skill estimate that the Apprentice Tutor platform uses to track student progress (see Appendix A.3). With the addition of the skill estimate, GPT-4's performance drastically improved when identifying logical mistakes. We were satisfied with the results and used this prompt on the rest of the tutoring topics.

Results

Our results show in Table 4 that when given enough context to describe what step in the problem-solving process the student is working on, GPT-4 can describe student errors with 87.8% accuracy. The LLM is most adept at identifying a correct answer, recognizing a matching answer 99% of the time, and questioning the validity of the correct answer only 1% of the time. GPT-4 performed the poorest on *Wrong Field* errors, unable to recognize nearly 30% of the time when a student had skipped a step in the problem solving process. The remaining types of errors – *Logical*, *Syntax*, and *Incomplete* – were accurately described roughly 80% of the time. The full breakdown by tutor content is shown in Table 5. With respect to **RQ1**, GPT-4 performed best at identifying *Incomplete* answers and performed the poorest when identifying *Wrong Field* errors and answers with multiple errors.

When the student's mistake is a common misapplication of the mathematical concept, GPT-4 excels. For the problem $400^3 \cdot 400^2$ when the student entered $(400 \cdot 400)^{3+2}$, GPT-4 correctly identified that "*The student mistakenly squared the*

Table 3 Accuracy of mistake identification by GPT-4 across different error types, using progressively more context, with the Factoring dataset (N = 199)

Error Type	Basic Context (Student Answer, Correct Answer, Problem)	Basic Context + Interface Information	Basic Context + Interface and Skill Estimate
Logical Error	40% (40)	68% (69)	81% (82)
Syntax Error	59% (14)	88% (21)	92% (22)
Incomplete	88% (8)	88% (8)	100% (9)
Correct Answer	80% (43)	100% (54)	100% (54)
Wrong Field	0% (0)	88% (7)	100% (8)
More than one error	0% (0)	0% (0)	66.7% (2)

The first column uses the basic context (Appendix A.1), the second adds the interface context (Appendix A.2), and the third adds skill estimates (Appendix A.3)

base 400 before applying the product rule of exponents." However, GPT-4 was unable to correctly diagnose student errors when the answer contained multiple mistakes. For example, when solving the problem $183^{15} \cdot 183^{6}$, the expected response is 183^{15+6} but the student entered 9. A plausible explanation from our team was that the student subtracted the exponents and did not write the base, classifying it as a Logical and Syntax error. GPT-4 stated "*The student seems to have misunderstood the problem entirely. Instead of simplifying the given exponents by applying the product rule, they've provided a single-digit number, which is unrelated to the initial problem.*". Rather than calling out the missing base or incorrect operation, the LLM did not find a pattern in the student's error.

The LLM performed especially poorly when describing *Wrong Field* errors. Frequently, GPT-4 would misclassify the error as a common mathematical misconception. When solving $(320^{4})^{3}$, a student skipped straight to 320^{12} instead of demonstrating the Product Rule step $320^{4 \cdot 3}$. The output included an incorrect, hallucinated diagnosis of a common misconception – "*The student made a mistake in understanding the power rule of exponents. They added the exponents (4+3 = 7) instead of multiplying them (4*3 = 12).*" – even though that statement is false. This type of correction must

Table 4 Overall accuracy of mistake identification across all tutoring topics by GPT-4 on each error type

Original Prompt	Final Prompt
75.3% (275)	83.2% (304)
67.3% (162)	82.9% (199)
80.3% (141)	86.9% (152)
90.0% (216)	99.0% (238)
34.5% (77)	72.3% (162)
32.0% (19)	71.67% (42)
71.9% (937)	87.8% (1144)

The first column shows the accuracy with the original prompt with the count of passing errors in parenthesis. The second column shows the accuracy of the final prompt with context with the count of passing errors in parenthesis

Table 5 The accuracy of mistake identification by GPT-4, on each error type, broken down by tutoring topics

Error Type	Exponents (511 errors)	Radicals (350 errors)
Logical Error	84.9% (90)	84.5% (60)
Syntax Error	80.6% (83)	84.6% (44)
Incomplete	85.7% (60)	83.0% (44)
Correct Answer	100% (106)	100% (58)
Wrong Field	66.0% (66)	67.1% (59)
More than one error	80.8% (21)	64.3% (18)
Overall	83.6% (426)	80.9% (283)
	Quad Eq. (89 errors)	**Quad Func. (56 errors)**
Logical Error	97.0% (32)	16.7% (6)
Syntax Error	92.3% (12)	75.0% (6)
Incomplete	92.3% (12)	75.0% (3)
Correct Answer	100% (4)	87.5% (7)
Wrong Field	72.0% (18)	-
More than one error	100% (1)	-
Overall	88.8% (79)	39.3% (22)
	Rational Eq. (98 errors)	**Factoring (199 errors)**
Logical Error	61.1% (11)	81.2% (82)
Syntax Error	85.0% (34)	91.7% (22)
Incomplete	96.2% (25)	100% (9)
Correct Answer	90.0% (9)	100% (54)
Wrong Field	100% (3)	100% (8)
More than one error	0% (0)	66.7% (2)
Overall	83.7% (82)	88.9% (177)

The Quadratic Functions tutor had no student errors categorized as "Wrong Field" or errors with multiple categories. We used hyphens ('-') to designate that GPT-4 was not assessed on that particular topic

be present often in the training data for GPT-4 to default to when it does not identify a pattern in the student's response.

For some fields, multiple students made the same error. We treated these as separate mistakes, since the students may differ on skill estimate or answer formatting. Because of the nondeterministic nature of LLMs, we often received wildly different responses. For example, when solving quadratic equations, if the correct answer is $16x^2 = 3 + 8$ but the student entered $16x^2 = 11$, GPT-4 would describe the error correctly as "*The student made a mistake in prematurely simplifying the equation. They added the 3 and 8 together to get 11 rather than expressing the right hand side as 3 + 8.*" or incorrectly as "*The student made a computational mistake by incorrectly adding 3 and 8 to get 11 instead of the correct sum, 11.*" We saw this specific error on the exact same problem fifteen times, nine of which GPT-4 was able to correctly diagnose.

Discussion

These results appear consistent with other studies (Phung et al., 2023; Azaiz et al., 2024) that indicate GPT-4 is approximately 80% accurate at identifying problems

in student responses. GPT-4 is near perfect at recognizing correct answer formats, though analytical educational technology (like intelligent tutors) can recognize a correct answer with 100% accuracy. In addition, Phung et al. (2023) finds that human tutors still outperform LLMs with a near 100% accuracy. Is 80% accuracy enough for LLMs to scale feedback generation in intelligent tutors? This depends on the kind of feedback that comes after misdiagnosing an error. It is possible that GPT-4 could craft feedback that disregards the incorrect diagnosis, or non-specific feedback about the concept as a whole. We explore this further in Section 4.

The least correctly diagnosed errors were Wrong Field errors, though we see an immense improvement from the original prompt. By adding the interface to the prompt, the LLM was more frequently able to identify when students prematurely simplified. This type of diagnosis describes which part of the student's response was prematurely simplified or indicates that the student should have written an expression rather than a single numerical value. The model does not output a diagnosis that the student should have entered their response in a different field. We hypothesize that feedback that addresses the current field and answer will help students turn an incorrect answer into a correct answer, rather than direct the student's attention elsewhere in the problem.

When examining the breakdown by topic, GPT-4 performed quite well when describing errors with operations with radicals and factoring polynomials. When examining these tutors, they stand out in terms of problem granularity. Both tutors have detailed breakdown of minute steps, as well as very descriptive instructions for each step seen in Figs. 2 and 1. The detailed problem breakdowns provide a lot of context for GPT-4 and describe how the student's response fits within the greater problem-solving process.

In contrast, GPT-4 performed quite poorly on identifying parts of quadratic functions. One possible explanation is that while students are given an image of the graph (as seen in Fig. 3), we gave GPT-4 the LaTeX equation. Though GPT-4 can process images, we thought the equation would be more informative for generating feedback, as we also provided the correct numerical answer as equations. This tutor also has minimal intermediate steps, so most incorrect answers appear to the untrained eye as a random selection of numbers, and the fewest incorrect responses, resulting in a small sample size.

Syntax errors and Typos can be especially hard to anticipate when writing hints. Fortunately, GPT-4 performed well when identifying these errors, especially in Quadratic Equations, Factoring, and Radicals when students need to use a variety of mathematical symbols in their answers, but less so in Exponents, which has fewer characters in each answer. A potential explanation is that with more characters, students can better represent the logic of the answer. With shorter answers, each character is more important to understanding the answer as a whole, and GPT-4 could not parse logically correct but syntactically invalid answers.

It is also likely that GPT-4's performance reflects some of the relative frequency of specific problem types in its training data. For instance, problems like $(-x-9)(x+10)=0$ are canonical examples that appear frequently in textbooks, instructional websites, and online math forums. In contrast, expressions like 483^{3+10} are rare and idiosyncratic. Because LLMs 'learn' from statistical regularities across massive datasets, problems that follow common patterns are more likely to be accu-

rately diagnosed (both examples used here were correctly diagnosed). This suggests that part of GPT-4's success in this study may stem from the alignment between our dataset and common instructional examples. We attempted to accommodate for this fact by not asking GPT-4 to do math. ITS are well equipped to address logic errors. LLMs have the potential to supplement ITS not by doing math, but by identifying irregular syntactic and language differences between the student's response and the correct answer.

One important note is how critical context is to GPT-4's performance. Especially in longer, multi-step problems where students must solve simpler subproblems and combine answers to obtain the final solution, GPT-4 struggled to situate the subproblem in the broader problem-solving context. Problems in Factoring, Quadratic Equations, and Rational Equations were incredibly difficult for the LLM to appropriately assess. Only once GPT-4 was given the HTML interface could it accurately situate the student's response into the whole problem.

These methods could be further improved by including an image of the problem layout. The HTML does give some clues to the layout of the problem, but the visual structure of the problem is important to students. GPT-4 may also benefit from the visual layout of the problem, more than just the code layout. This may help in tutors like Quadratic Functions, where the graph of the function is key to answering all fields.

We might also improve the accuracy by reconstructing the current problem state. Errors like the student entering a response into the wrong field might be better identified if GPT-4 was given the student's responses in other fields. GPT-4 might also better detect when the student has an error elsewhere in the problem and can direct student attention to another field. By considering the whole problem state, we may see more varied feedback that addresses student misconceptions beyond the current field. We could also give the LLM all prior student actions, such as prior responses to the field or previous hints received. This could help GPT-4 detect maladaptive student behaviors, like guessing and checking, or hint abuse. However, these behaviors would be difficult for the Large Language Model to correct, and might better be handled by the student's instructor.

Even with the extended context given, GPT-4 still underperformed when identifying errors in complex, multi-step problems that require greater contextual awareness. Providing GPT-4 with more detailed problem layouts, visual representations, or access to the student's full problem-solving history could improve its accuracy. In the next study we examine the GPT-generated feedback to see whether the content can mitigate some of these diagnostic challenges.

Study 2: Generating Corrective Feedback

Intelligent Tutoring Systems typically rely on static, pre-planned feedback that addresses the most common student errors. This rule-based approach can limit their ability to respond to the full range of student input. As a result, ITS often miss the opportunity to guide students through their individual problem-solving processes like a human tutor would, responsive and informed by the student's answers. Using a

large language model for feedback authoring could enable ITS to deliver explanations personalized to the specific nature of each error.

The previous study found that GPT-4 can diagnose student errors with 87.8% accuracy. This study now investigates whether GPT-4 can craft hints targeted to each specific mistake. Even in cases where GPT-4 was unable to accurately describe the student's mistake, it still may be able to craft feedback general enough to be helpful. By first identifying the error in the prior study, we set the foundation for generating feedback that aligns closely with each student's error. This approach could make ITS more adaptive by supporting on-demand hint generation that adjusts to the context of each error. If LLMs can generate targeted feedback on the fly, ITS could more easily scale their content offerings without extensive manual input.

For feedback to be useful to students, it also has to be accessible. This study also examines the sentiment and readability of the hints provided. Affective elements in feedback – such as a supportive tone or clear language – play a role in student outcomes (Howard, 2021). Research shows that when feedback is overly complex, it can impact students' ability to correct errors and reduce overall learning efficiency (Shute, 2008). Moreover, feedback that conveys a negative or critical tone may cause students, particularly those with low self-efficacy, to disengage and focus on personal inadequacies rather than the learning task itself (Erickson et al., 2022; Hattie & Timperley, 2007). Prior research found that GPT-generated hints were targeted, comprehensible, appropriate, and delivered with a positive tone (Roest et al., 2024; Xiao et al., 2024). While these studies examined affective feedback through human coding, this study uses automated measures to analyze sentiment and readability. This method could allow scalable evaluation of an on-demand hint generation system that does not require extensive manual oversight, ensuring that hints are supportive and accessible to the target audience.

Methods

As described in the previous study, the prompt (see Appendix-A.3) instructed GPT-4 to first diagnose the student's error, then craft targeted feedback that addresses their specific mistake. Section "Study 1: Diagnosing Student Errors" analyzed GPT-4's error diagnoses. We examine the generated feedback in this section for its relevancy to the student's answer and the broader problem. We again took an inductive approach to reading the feedback, first just looking for whether the feedback was relevant to the student's mistake. After a first pass through the generated hints, we noted that even if a hint did not address the mistake, it could still be mathematically correct and potentially helpful. During the second pass, two independent coders categorized the hints into Targeted, General, Gives Answer, and Incorrect, achieving substantial agreement on the entire dataset (Cohen's Kappa 0.768). For our analysis, we used the head coder's categories when there were disagreements.

The main overarching pattern to LLM-generated feedback is that nearly every hint followed the same structural pattern, shown in Fig. 5. Most feedback began with the word "Remember", followed by a description of what the student needs to do to get a correct answer, finishing with an encouragement to try again. Even though GPT-4 was prompted to provide targeted feedback, the description of what to do was not always

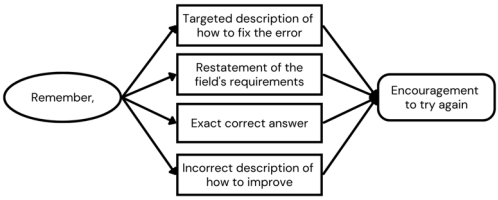

Fig. 5 Most GPT-generated feedback started with the word "Remember" and finished with an encouragement to try again

specific. Sometimes it was a general description of the math concept, sometimes it was the exact correct answer that the student should have entered, and sometimes it was an incorrect description of what would improve the student's answer. We coded each hint according to these four categories.

To examine the sentiment of the feedback, we used TextBlob (Loria et al., 2018), a common lexicon-based python library for textual analysis. We categorized the feedback as Positive if the polarity was greater than 0.001, Negative if the values are less than −0.001, and Neutral if the values fell in between. To assess the readability of tutor-generated hints, we employed both the Crowdsourced Algorithm of Reading Comprehension (CAREC) (Crossley et al., 2019) and the Coh-Metrix L2 Readability Index (Crossley & McNamara, 2008) using the Automatic Readability Tool for English (ARTE) (Choi & Crossley, 2021). These metrics were chosen because they move beyond traditional grade-level formulas, offering a more nuanced and empirically grounded understanding of text accessibility. CAREC incorporates linguistic features tied to actual human comprehension – such as lexical sophistication, text cohesion, and semantic content. Similarly, the Coh-Metrix L2 Readability Index captures text characteristics like syntactic simplicity, word familiarity, and cohesion. Many of the students using the Apprentice Tutors are still developing academic language and content knowledge. As such, readability metrics grounded in cognitive and psycholinguistic factors are especially relevant for ensuring that hints are accessible and supportive of comprehension in adult education contexts.

Results

Figure 6 describes the type of feedback given when the LLM did or did not identify a mistake correctly. Even when GPT-4 did not accurately diagnose the mistake, most of the time it was still able to craft feedback that described the discrepancy. However, most feedback with incorrect or misleading information comes after misdiagnosing the student's error.

The goal of our prompting strategy was to scaffold the generation of corrective feedback by first identifying the student's mistake. Using this approach, 66% of the generated feedback was *Targeted* to the student and specifically addressed the mistake. For example, when using the Quadratic Formula on $-x^2 + 5x - 2 = 0$, the student entered the *Incomplete* answer $\frac{5}{2}$ when they should have entered $\frac{5}{2} - \frac{\sqrt{17}}{2}$. The LLM identified the mistake correctly as "*The student forgot to subtract the square root of the discriminant over 2 from the value they calculated.*" and the associated feedback directly addresses the student's mistake: "*Remember that the quadratic formula is* -b ± sqrt(bˆ2 - 4ac) / 2a. *You correctly calculated* -b/2a, *but forgot to subtract the* sqrt(bˆ2

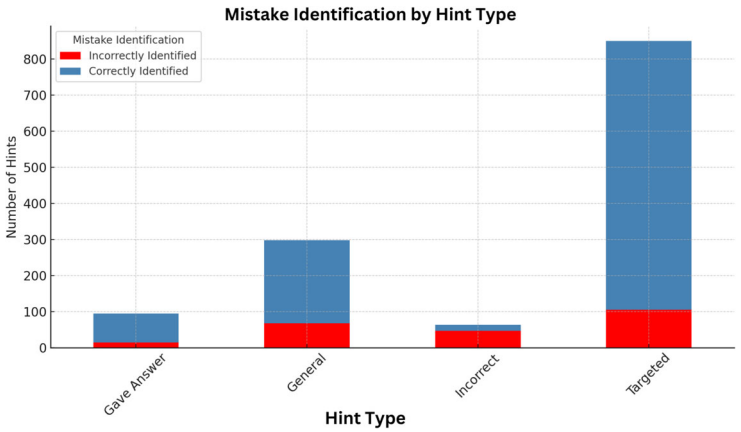

Fig. 6 When prompted to give targeted feedback, GPT-4 responded with different types of feedback. This shows the breakdown of feedback type when the LLM did or did not identify the mistake

- 4ac)/2 *part. Try the problem again, and make sure to include all parts of the quadratic formula in your solution.*". Even when the answer is *Correct*, the LLM can still give feedback targeted to the particular skill the student demonstrated success with: "*Good job identifying one of the values for which the function is undefined! Continue with the same process to find any other values.*".

Particularly interesting are the *Wrong Field* errors. Rather than directing the student to enter their answer elsewhere, the feedback states to slow down and verify what the question asks. When solving quadratic equations using the square root property, the student entered $x^2 = \frac{11}{16}$ into the "Simplify right hand side" field when the tutor expected to see $16x^2 = 11$. The feedback from GPT-4 stated, "*You may have moved too quickly to get* x^2 *by itself. Remember to only move the constant term, -8, to the other side of the equation first. This step is to isolate the quadratic term,* $16x^2$*, on one side*." Or, when the student needs to factor the first radicand but they have factored *and* multiplied the coefficients, the feedback says "*Remember at this stage you are simply factoring the radicand (the number under the square root). Factoring does not require any multiplication with the coefficient. Identify what numbers multiply to 72 and rewrite the expression at this point.*"

Even hints that did not directly address the student's mistake were often mathematically correct and could be potentially helpful. We classified 22.3% of hints as *General* – that is, a general description of how to satisfy the field. Rather than directly addressing the student's mistake, the LLM re-states the goal of the field. This type of feedback is especially effective when the LLM was unable to correctly identify the mistake. The generated feedback is still applicable to the field and does not offer any incorrect direction. Feedback like "*Remember the product rule for exponents: when multiplying two powers of the same base, add the exponents. Try solving the problem again with this rule in mind.*" reinforces the goal and what kind of expression will satisfy the field, however it is not personalized or tailored to student errors. The equivalent pre-written hint from the Apprentice Tutor platform is "*The product rule says that when you have a product of two exponential terms with the same base, you add*

the exponents.". These types of general hints are very similar to the pre-written hints already available in the ITS platform.

Even though 7.1% of the feedback includes the exact answer, it also includes explanatory feedback for how the student's answer differed from the correct answer. For example, when the tutor expects the equation $x = -4$ but the student entered only the numerical value -4, GPT-4 gives the feedback "*Remember to write your answer in the form of an equation, indicating which variable the answer corresponds to. In this case, you should write* '$x = -4$' *instead of just the numerical value.*" Or, when a student calculated the exact value of 537^4 as 83, 156, 680, 161, the feedback reads "*Remember the power rule of exponents: when you raise a power to another power, you multiply the exponents. So,* (537^2)^2 *should be simplified to* 537^4. *Avoid calculating the resulted value.*" These hints are very similar to the bottom-out hints in the Apprentice Tutors platform, except the bottom-out hints do not include explanatory feedback about the difference in the two responses.

Incorrect directions were given in 4.8% of the feedback. Typically, when the feedback was incorrect, it was because the LLM did not accurately identify the student's mistake. For example, in this Wrong Field error 234^{27} should have been 234^{17+10}. The LLM diagnosed this error incorrectly as "*The student added the bases instead of the exponents, as the product of powers rule instructs.*" and gave feedback targeted to its interpretation of the student's mistake: "*Remember, the product of powers rule states that to multiply powers with the same base, you keep the base and add the exponents. You added the bases instead. Try it again, but add the exponents instead of the bases.*" Even when the LLM was able to identify the mistake, in rare cases the feedback instructions were incorrect. As an example, the student entered $5\sqrt{6^2 \cdot 5} + 3\sqrt{5}$ instead of $5\sqrt{36 \cdot 5} + 3\sqrt{5}$ and the LLM said that "*The student correctly factored* 180 *as* 6^2 * 5*, however they mistakenly included the exponent* 2 *inside the square root.*". Even though it correctly pointed out the difference between 36 and 6^2, the associated feedback is practically unintelligible: "*Remember, when factoring under a square root, the whole factor remains outside the square root, not just the base number. Try factoring* 180 *again with this in mind.*".

The generated feedback was predominantly positive in sentiment. Most hints included an encouraging statement like "*Try again!*" or "*You're on the right track!*". Figure 7 shows that the polarity ranges from -0.75 to 1.00, with a high concentration centered just below 0.00. The histogram being skewed right and most data being greater than zero indicates that most feedback is neutral to positive in polarity. The bar graph in Fig. 8 agrees, with most feedback falling into the positive category, and only 292 hints categorized as negative. The negative hints are more directive in nature, such as "*You need to carefully analyze the initial problem and the characteristics of quadratic equations in order to correctly find the vertex of the parabola. In this case, you should try to calculate the y-coordinate of the vertex using the known x-intercepts and the y-intercept.*" and generally do not include any encouragement, just instructions to try again.

The Crowdsourced Algorithm of Reading Comprehension (CAREC) in Fig. 9 (left) reflect predicted text comprehension difficulty, with higher values indicating greater comprehension difficulty. The histogram shows that most GPT-generated feedback

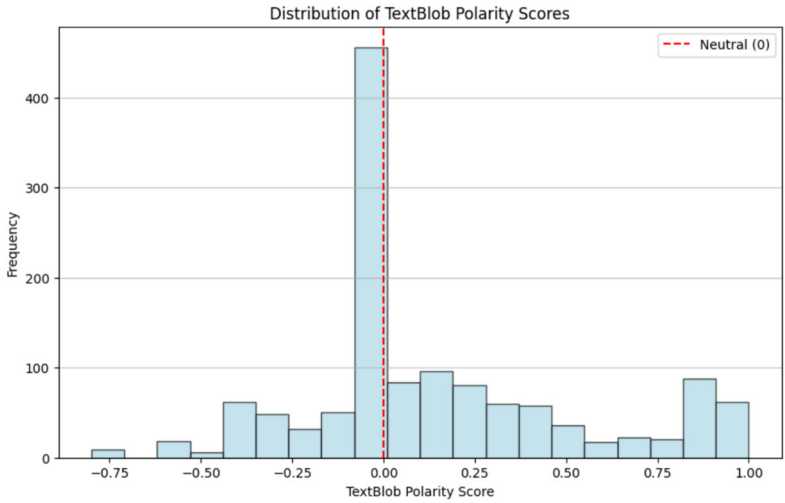

Fig. 7 Histogram of polarity values from TextBlob. Most hints scored just below 0.00 as neutral, with the rest skewed positive

falls between approximately -0.2 and 0.4, with the average score 0.128. This distribution suggests that the majority of hints are moderately accessible. The highest scoring hints are the most difficult, using advanced vocabulary complex sentence structures, and procedural instructions like "*Remember that the x-coordinate of the vertex is the line of symmetry, which is the average of the x-intercepts in a quadratic equation. Try to use that information to recalculate the x-coordinate of the vertex.*" (CAREC = 0.778). The lowest scoring hints contain very little academic content and are mostly

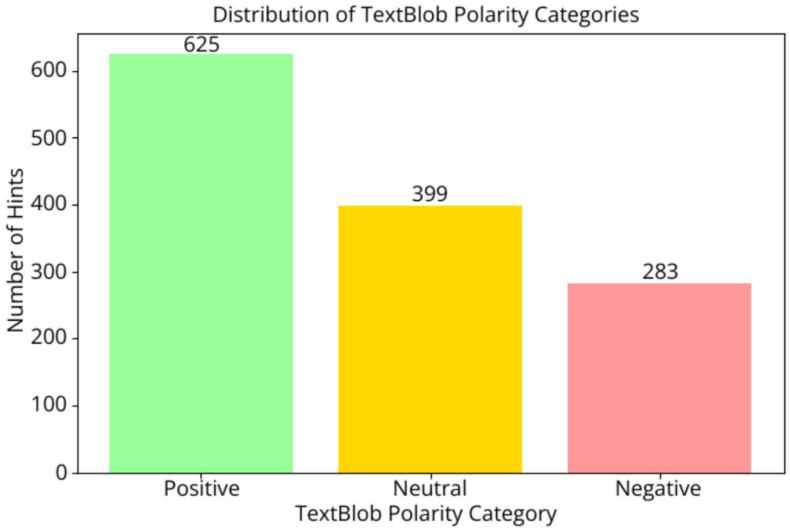

Fig. 8 Counts of feedback classified as Positive (polarity > 0.001), Negative, (polarity < -0.001) or Neutral from TextBlob. Most feedback was classified as positive or neutral

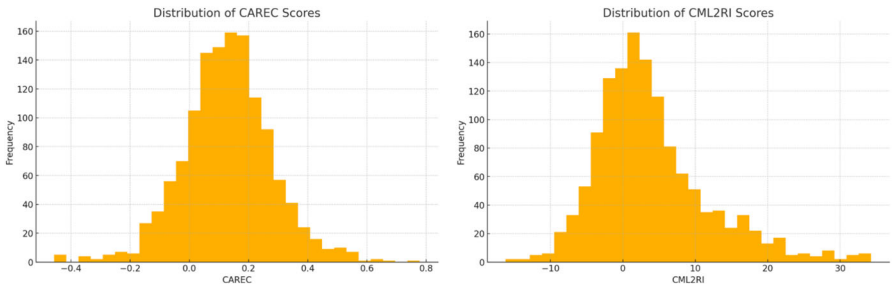

Fig. 9 Crowdsourced Algorithm of Reading Comprehension (CAREC) (left) clustered just below 0.2 with most hints falling between -0.2 and 0.4. Coh-Metrix L2 Readability Index (CML2RI) (right) similarly clusters just above 0 with a long tail to the right

short conversational sentences like "*Keep up the good work!*" (CAREC = -0.457). The CAREC scores help differentiate hints that require close attention compared to those that are brief and motivational.

On the other hand, Coh-Metrix L2 Readability Index (CML2RI) in Fig. 9 (right) reflect predicted reading fluency, particularly for second language readers, where higher values indicate that a text is easier to process. The histogram shows a slightly right-skewed distribution with an average at 3.95. The highest scoring hints have strong lexical cohesion or familiar vocabulary, like "*Remember that the coefficient includes the sign. In this case, c is -90, not 90. Be sure to include the sign when identifying the coefficient*" (CML2RI = 33.96176), which make them easier to process. The lowest scoring hints lack the lexical cohesion that supports fluent reading, like "*Check your input format. When using the quotient rule for exponents, we simply subtract the exponent of the denominator from the exponent of the numerator without bracketing the denominator exponent*" (CML2RI = -16.20385). The histogram shows that most hints are more cohesive than average.

Discussion

Prior work also finds that most GPT-generated hints are targeted, comprehensible, appropriate, and delivered with a positive tone (Roest et al., 2024; Xiao et al., 2024). For example, Wan and Chen (2024) found that students rated GPT-generated feedback as equally or more useful than human feedback, often appreciating its detailed explanations and clarity. Our results are consistent with previous findings, in that most hints were specific to the learner's mistake, had a neutral to positive sentiment, and are generally comprehensible and cohesive. This suggests that GPT-generated feedback is not only relevant and specific, but also accessible to a wide audience of learners, and that these attributes could be captured through automated analysis.

Using the Crowdsourced Algorithm of Reading Comprehension (CAREC) and the Coh-Metrix L2 Readability Index (CML2RI) in the automated feedback pipeline captures both the lexical sophistication and the syntactic simplicity of generative feedback. CAREC measures how human comprehensible a text is through features like semantic content and lexical sophistication. CML2RI measures word difficulty and text cohesion. We observed that feedback with high CAREC scores included academic

vocabulary like '*symmetry*', '*quadratic*' and '*numerator*' and procedural instructions on how to derive the correct answer. Feedback with low CAREC scores were readable, supportive statements like "*Good job!*" and did not include technical vocabulary. Hints with high CML2RI are easier to process due to their strong lexical cohesion and familiar vocabulary, while hints with low CML2RI are dense and difficult to parse.

The interaction between CAREC and CML2RI offers a more nuanced perspective on the readability and instructional quality of feedback. We sampled hints from the score space and qualitatively examined their instructional and linguistic characteristics. Hints that score high on both CAREC and CML2RI tend to have an accessible structure, yet still present a meaningful instructional challenge. For example, "*The line of symmetry in a quadratic equation is the vertical line that passes through the vertex. Use this to check your answer.*" (CAREC = 0.41109, CML2RI = 15.72040) contains precise and helpful mathematical content and is presented with clarity. In contrast, hints that are high on CML2RI but low on CAREC may flow well but provide little to no instructional value. These include feedback like "*Good job, your answer is correct!*" (CAREC = -0.10953, CML2RI = 16.50652), which is readable but not personalized and does not contain any contextual information.

Hints that have low CAREC and CML2RI scores require little cognitive processing but also have low cohesion, often sounding vague or awkward without delivering useful information. These hints look like "*Review your arithmetic when simplifying equations. When moving constants around, make sure you've accurately added or subtracted them correctly.*" (CAREC = -0.12792, CML2RI = -9.59537). Finally, hints with high CAREC and low CML2RI predict a high comprehension difficulty and low reading fluency. These hints are syntactically complex explanations that include high-level mathematical concepts, like "*You need to carefully analyze the initial problem and the characteristics of quadratic equations in order to correctly find the vertex of the parabola. In this case, you should try to calculate the y-coordinate of the vertex using the known x-intercepts and the y-intercept.*" (CAREC = 0.43234, CML2RI = -5.36947). This quadrant-based view could allow future developers to better tailor the hints to each student by asking GPT-4 to rephrase or include more personalized information. CAREC flags hints that may be conceptually challenging, while CML2RI flags hints that may be difficult to parse. These observations suggest that the measures capture meaningful differences in hint quality. However, since our analysis was limited to a single dataset and domain, further investigation is needed to establish generalizability to other instructional contexts.

Though most feedback included a positive tone, 14%-20% of the feedback was classified as negative. These are hints like "*Remember, the x-coordinate of the vertex of a parabola (line of symmetry) is the average of the x-intercepts. Try calculating this average again, and remember that division applies even when dealing with negative values.*" or "*Remember that the y-coordinate of the vertex of the parabola is the minimum or maximum value of the function. In this case, the vertex is a minimum, so its y-coordinate should be negative. Please re-check that your vertex is the lowest point on the graph.*". Much of the feedback rated as positive includes a motivational affirmation like "*Good job!*" or "*You're nearly there!*", but the feedback rated as negative lacks this positive reinforcement. The negative hints focus on correction and redoing work, containing phrases like 'recheck' or 'look again', which could make these hints feel

critical or reprimanding to students. Another possible explanation for the negative sentiment is the use of the mathematical term 'negative'. Though relevant and targeted, 24% of the hints that TextBlob categorized as negative contain the word 'negative' or 'wrong'. This may have contributed to the overall negative classification, even though the intention of the feedback was simply to point out mathematical concepts or errors.

One important finding is that even when GPT-4 did not accurately describe the student's mistake, it was still able to craft targeted or relevant feedback 73.7% of the time. By keeping the feedback to a description of the math concept or restating what form the answer should be for the field, GPT-4 was able to recover from its diagnostic inaccuracy. For example, when calculating the product of a and c for the trinomial $6x^2 - 23x + 20$, the student entered 26 instead of 120, having added instead of multiplied. The LLM diagnosed this error incorrectly as "*The student seems to have multiplied the coefficients of the first two terms of the polynomial (a and b: 6 and -23) instead of multiplying the coefficient of the first term and the constant term (a and c: 6 and 20), which is the correct process.*". However, the following hint is general enough that even though it was derived from an incorrect assessment, it still contains accurate directions for the student: "*Remember, when factoring by grouping, the product 'ac' is found by multiplying the coefficient of the* x^2 *term (a) and the constant term (c). So for this problem, you would multiply 6 (coefficient of* x^2*) and 20 (the constant term). Please try again.*".

In addition, general feedback that orients students toward their goal may be more effective than error explanations. McKendree (1990) demonstrated that goal-focused feedback was more effective than explaining why a solution was incorrect. However, their incorrect explanations were always addressing a logical error. As shown in Section "Study 1: Diagnosing Student Errors", only 30% of student errors can be attributed to a logical mistake. It is unclear whether a general, goal-oriented hint would still be effective when students understand the logic but misrepresent the form.

Even though GPT-4 was only able to diagnose a mistake approximately 80% of the time, we are seeing much better results for useful feedback. Only 5% of generated feedback was factually wrong and 7% contained the correct answer – notably lower error rates than the prior study. GPT-4 was better able to construct relevant and appropriate feedback than it was able to describe student errors. Given that the prompt stated not to give away the correct answer directly, and that typical ITS feedback is delivered in a sequence ending with a bottom-out hint, only about 10% of the generated feedback (both feedback with inaccurate directions and bottom-out hints) should be reviewed for accuracy and alignment with the intended instructional sequence. With respect to **RQ2**, the vast majority of feedback GPT-4 generated was helpful and informative according to our human evaluators.

The feedback that GPT-4 produces tends to explain broader concepts or provide guidance about the structure of the answer, generating correct and potentially useful hints even when it did not identify what the student did wrong. This approach allows the LLM to offer useful hints related to the topic at hand and could potentially help students refine their answers without necessarily pinpointing their specific error. GPT-4 will often restate the formula or method ("*Remember, the goal is to find two numbers that multiply to 'c' (-8 in this case) and add up to 'b' (2 in this case). Use these numbers to fill the brackets following this structure: (x + number1)(x + number2).*"), describe

missing parts of the student's answer ("*Remember to set your denominator equal to zero in order to find the values of x that are undefined for the equation. Try adding '=0' to your equation.*"), or redefine the mathematical concept ("*Think about the one-to-one property of exponents: if two exponents with same base are equal, then their powers must be equal as well. Try to equate the powers of the given equation, and then solve for x.*"). This approach means that GPT-4 can generate relevant and applicable feedback even when the LLM is not perfectly aligned with the student's mistake. By focusing on the broader context, GPT-4 can still provide useful hints.

The tendency to focus on broader concepts rather than the student's specific mistake is closely tied to how GPT-4 was instructed to generate feedback. The prompt instructed GPT-4 to "*Provide 1-2 sentences of feedback that will help guide the student to correct their mistake, without giving away the full answer.*" (see Appendix A.3). The sentence limitation likely contributed to the concise, concept-focused nature of many of the hints, as the model often defaulted to rephrasing definitions, restating procedural steps, or describing the structure of the expected answer. The instruction to avoid giving the full answer appears to have had only moderate success – 7% of hints still contained the correct answer. Perhaps rephrasing the prompt to remove the word '*not*' would increase the effectiveness of this instruction. Truong et al. (2023) found that Large-Language Models fail to reason under negation, even when scaling up the model's size. Only through instruction tuning did the authors see improvement in negation tasks. Future work should refine this prompt with the intention of producing fewer bottom-out hints.

The prompt also instructed that "*The goal is to help the student learn and figure out how to solve the problem on their own, not to simply give them the correct answer. Provide targeted feedback that addresses their specific mistake.*". This constraint may have encouraged more general or overly cautious hints, given that 35% of the hints were vague on the specific correction students needed to make. It also reflects the trade-off between simplicity and safety: the prompt may be suppressed more directive feedback that risks being incorrect. Future work could explore variants of the prompt that are dynamically adjusted based on diagnostic confidence, or allow for more gradation in specificity, similar to the Apprentice Tutors orientation, instrumental, and bottom-out hints.

While prompt design can influence level of detail, it does not fully resolve the challenges of generating feedback that is both accurate and personalized. One of the biggest appeals of using a Large Language Model is its potential to create student-specific guidance personalized to each learner. Unfortunately, 34.2% of the feedback generated was quite general, incorrect, or a bottom-out hint. General hints can still be good for learning, but they contain the same information that pre-coded hints would when creating bug rules. In addition, ITS hinting strategies are based on pedagogical best practices (Aleven et al., 2016), and may have a specific sequence. As such, they would not accidentally deliver a bottom-out hint prematurely and would never deliver a hint with incorrect directions. Though only 4.8% of the feedback is factually incorrect, this risk is significantly greater than the 0% incorrect feedback of ITS. This raises the question of whether the feedback is good enough to use directly with learners, where correctness is critical, and if we can detect that a hint is inappropriate before

displaying it to the student. In the next study, we explore methods for evaluating the informativeness and helpfulness GPT-generated feedback automatically.

Study 3: Automated Feedback Evaluation

We have demonstrated in the prior study that while GPT-4 can generate useful, context-aware hints, it can also produce responses that are misleading, incorrect, or unhelpful. Students rely on accurate and constructive feedback to guide their learning, so any amount of unhelpful information presents a challenge to using feedback from large language models. Currently, human oversight is required to filter out poor-quality responses, which is time-consuming for ITS authors and limits the scalability of LLM-generated feedback. To use the full potential of LLMs effectively within ITS, it is essential to distinguish high-quality hints from low-quality hints without constant human intervention.

The goal of this study is to determine if an LLM can autonomously assess the quality of its own feedback. We explore several different metrics that use LLMs to measure the helpfulness and informativeness of the hints without human intervention. With reliable self-evaluation, ITS could provide scalable, autonomous feedback generation, giving students personalized, real-time support. We aim to evaluate quality assurance mechanisms that could make LLM-generated feedback a self-sustaining component of ITS.

Methods

McNichols et al. (2023) used LLMs to generate math multiple choice questions, distractor answers and explanatory feedback for the distractors. They outlined a procedure for automatically evaluating the quality of feedback using LLMs. First, prompt the LLM to adjust the incorrect answer using the problem description and feedback. If it can generate the correct answer, this indicates the feedback was helpful. Next, prompt the LLM to predict the incorrect answer from the problem and feedback. If it can reproduce the incorrect answer, this indicates the feedback is informative. We evaluated the feedback generated from student errors using these two metrics.

We used the same prompts from McNichols et al. (2023) with one change. Intelligent Tutor transactions may be challenging for LLMs to update incorrect answers since many of the fields are partial answers or intermediate steps, rather than a final simplified form. Just like the feedback prompt, we included the problem interface, the skill, and the field the student was working on. This way the LLM has all the context needed to update or predict the answer to an intermediate step. The prompts can be found in Appendices A.4 and A.5.

Prior work (Pardos & Bhandari, 2024) indicates that repeated prompting will converge on an answer that is more correct than a single response. For a subset of the transactions, we ran each test ten times and selected the most common (modal) answer to see if accuracy increased. We also examined the frequency of correct answers.

Phung et al. (2024) took a similar approach to evaluating programming hints in their GPT4Hints-GPT3.5Val system. They prompted a less advanced version of GPT

to update the buggy program, acting a simulated student. They compared the LLM's success when it is given an explanation of the error to when the LLM is given no extra information. The explanation needs to be informative and specific enough that more simulated students are able to update the incorrect answer successfully with the explanation than without. They claim that the feedback is helpful and informative if for ten (n) prompts, the number of simulated students who can update the incorrect answer with the error explanation (n_2) is greater than the number of simulated students who can update the incorrect answer without the explanation (n_1), and $\frac{n_2}{n} > 0.5$ or $\frac{n_2}{n} > \frac{n_1}{n} + 0.25$. These last two requirements enforce that only quality feedback will pass and n_2 is sufficiently great. When n_1 is very low, nearly any feedback could pass the $n_2 > n_1$ requirement, so they ensure that the ratio of correct solutions produced with the explanation exceeds the threshold of 50% or is at least 25% higher than the ratio of correct answers generated without the explanation. We also performed this test, using the update answer prompt in Appendix A.4 ten times on GPT-3.5 with the explanation instead of the hint, and then removed the explanation for another ten prompts.

Results

Table 6 describes the number of feedback messages that passed the McNichols et al. (2023) tests. Of the 1303 feedback samples, only 40.7% of the time was the LLM able to adjust the incorrect answer to the correct answer, and only 30.2% of the time was the LLM able to predict the student's incorrect answer. Using these tests in practice would remove nearly 80% of the generated feedback, even though only 4.2% of the feedback generated was factually incorrect.

When we examine the breakdown by feedback type in Table 7, GPT-4 performs the best when adjusting the answer and predicting the student's answer for hints that contain the correct answer. This type of feedback often takes the form "*You entered X when you should have entered Y*", embedding both the correct answer and the student's incorrect answer. While these types of hints performed well on the automated tests, their success may accurately reflect how helpful or instructive they really are. They simply include the information the tests are looking for.

Even when the hint is targeted to the student's mistake, GPT-4 could only adjust the answer to the correct answer 54% of the time and predict the incorrect answer 37% of the time. We do see that GPT-4 performed worst on feedback that gave incorrect directions. Only 3.1% of incorrect feedback passed both tests; that is, two out of the 57 incorrect hints passed. This indicates that if these tests were used in practice, students would receive incorrect feedback 0.15% of the time. However, because of the low

Table 6 Percent of feedback messages that successfully adjusted the incorrect answer to a correct one, predicted the student's incorrect answer from the feedback, successfully generated both, or passed at least one test

Evaluation Criterion	Percent Passing
Adjusts Student Answer	40.7%
Predicts Student Answer	30.2%
Passes Both Tests	21.4%
Passes At Least One Test	49.5%

Table 7 Pass rates for GPT-4 feedback evaluation, broken down by feedback type

Feedback Type	Correctly Adjusted	Correctly Predicted	Both	Either
Incorrect	23.4%	9.4%	3.1%	29.7%
Gave Answer	62.1%	45.3%	37.9%	69.5%
General	23.4%	24.5%	15.4%	51.3%
Targeted	53.8%	37.2%	30.0%	60.9%

pass rate of all other types of hints, students would rarely receive any GPT-generated feedback - approximately 21% of incorrect answers would receive a personalized hint.

The predicted student answer pass rate is quite low, and one reason could be how frequently GPT-4 self-selected to not predict. Even for feedback that specifically called out the student's mistake, 38.4% of the time GPT-4 responded that there was insufficient information to predict what the student's answer might have been. Table 8 outlines how often GPT-4 said it did not have enough information to predict or update an answer. We do see that GPT-4 more readily predicted and adjusted when the feedback contained the correct answer, aligning with the previous finding that the feedback that passes these tests most frequently are hints that contain the correct answer. Hints that contained broader information about the goal of the step earned the highest rate of refusal. More often than not, GPT-4 stated that these hints did not have enough information to predict or adjust the answer.

Unfortunately, these methods did not totally align with human evaluation of the feedback. Feedback that human evaluators marked as incorrect still passed the adjusted answer test 23.2% of the time and the predicted answer test 9.4% of the time, though it only passed both tests 3.1% of the time. Feedback that specifically addressed a student's mistake only passed both tests 30% of the time, while bottom-out feedback passed 38% of the time. According to GPT-4 and these evaluation metrics, the most informative and helpful hint is one that gives away the correct answer.

We also had GPT-4 predict and adjust the answers ten times on the Factoring, Rational Equation, and Quadratic Equations datasets to see if, given multiple chances, GPT-4 would converge on an accepted answer. This subset is roughly one third of the entire dataset and includes a variety of mathematical topics as well as both long multi-step and short tutor interfaces. In the subset, there are 237 Targeted hints, 97 General hints, 42 hints that contain the correct answer, and 10 Incorrect hints for a total of 386 hints. We see in Table 9 that with more attempts, GPT-4 was able to generate the correct response more frequently, but the increase in accuracy is small, especially

Table 8 GPT-4 refusal rates by feedback type. This table describes cases where GPT-4 declined to revise or predict the student's answer

Feedback Type	Refused to Adjust	Refused to Predict	Both	Either
Incorrect	9.4%	26.3%	4.7%	40.6%
Gave Answer	1.1%	26.3%	1.1%	26.3%
General	16.8%	58.1%	7.7%	67.1%
Targeted	6.8%	38.4%	2.8%	42.4%

Table 9 Count of each passed test for the subset of 393 transactions

Metric	Single Response	Modal Response	Average Matches of 10
Adjusted Answer	172 (43.8%)	191 (48.6%)	4.28
Predicted Answer	136 (34.6%)	140 (35.6%)	1.97

for predicting the student's incorrect response. The single response column was the first answer that GPT-4 provided, and the modal response column represents the most common answer from ten prompts. On average, GPT-4 could generate two matching student responses and four correct answers out of ten opportunities.

We see the difference in success rates when using a single response or the modal response in Figs. 10 and 11 broken down by feedback type. For both tests, we see improvement for Targeted and General feedback, though the gains for Targeted hints are very small. For hints that contains the correct answer, the modal answer performs slightly worse than a single random answer, either converging on a reasonable but non-matching answer, refusing to predict, or questioning the validity of the stated correct answer. When the feedback contained incorrect directions, in a few cases, GPT-4 could generate the correct answer with multiple opportunities. Given that LLMs are already probabilistic text generators, one attempt appears to be no better than multiple attempts in most cases. Because we did not see a drastic improvement to the pass rate, we opted not to perform this test on the entire dataset.

We now examine the results from the Phung et al. (2024) tests. These tests used an earlier model of GPT to act as a simulated student. The authors used GPT-3.5 as the simulated student in these tests because it is more error-prone and may better model the behavior of a struggling learner. We simulate ten students using GPT-3.5, first asking them to update an incorrect answer when given an explanation of the error, and then taking away the explanation. If the number of successes with the explanation (n_2) is sufficiently greater than the number of successes without the explanation (n_1),

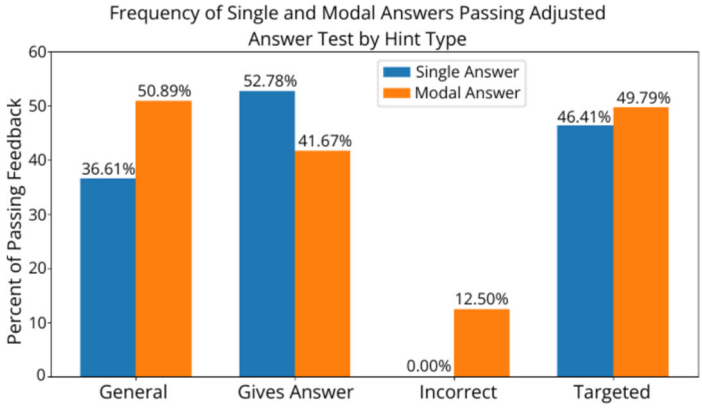

Fig. 10 Percent of feedback messages that passed the adjusted answer test when using a single or modal answer on the Factoring, Rational Equation, and Quadratic Equations data subset. We see modest improvements when using the modal answer for Targeted and General feedback, though we also see more Incorrect feedback passing

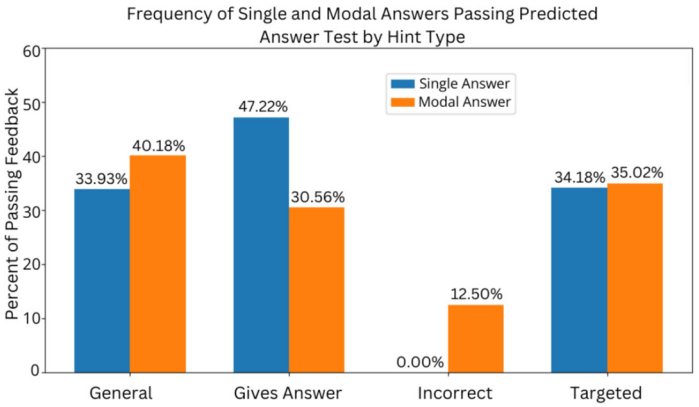

Fig. 11 Percent of feedback messages that passed the predicted answer test when using a single or modal answer on the Factoring, Rational Equation, and Quadratic Equations data subset. We see modest improvements when using the modal answer for Targeted and General feedback, though we also see more Incorrect feedback passing

then the explanation is informative. Sufficiently greater means that GPT-3.5 is at least 50% successful with the explanation ($\frac{n_2}{n} \geq 0.5$), and is at least 25% more successful with the explanation than without ($\frac{n_2}{n} \geq \frac{n_1}{n} + 0.25$). These thresholds aim to ensure that the feedback is not just slightly better than nothing but is meaningfully helpful. We applied this method to see if an LLM could evaluate the accuracy of the error diagnoses analyzed in Section "Study 1: Diagnosing Student Errors".

Table 10 describes how each metric fairs when the GPT-generated error explanation correctly describes the student's error, and when GPT-4 failed to accurately describe what the student did wrong. The results show a clear distinction between correct and incorrect explanations. When GPT-4 accurately identified the student's error, the explanation is more likely to meet the informativeness thresholds. Nearly half the explanations met either threshold, whereas only 21.4% of incorrect explanations met the thresholds. Interestingly, 83% of incorrect explanations were more informative than no explanation while only 78.6% of correct explanations were more informative than no explanation. This suggests that even flawed explanations can sometimes provide helpful context – perhaps by introducing relevant concepts – even if the spe-

Table 10 Percent of feedback messages that passes Phung et al. (2024) tests, broken down by whether GPT-4 was able to correctly diagnose the student's error

Evaluation Metric	Mistake Identified	Mistake Not Identified
Improved with Explanation ($\frac{n_2}{n} \geq \frac{n_1}{n}$)	78.6%	83.0%
At Least 50% with Explanation ($\frac{n_2}{n} \geq 0.5$)	46.0%	19.7%
Improved by at least 25% ($\frac{n_2}{n} \geq \frac{n_1}{n} + 0.25$)	24.9%	8.3%
Meets Either Threshold (above 50% OR 25% improvement)	48.2%	21.4%
Improves and Meets a Threshold ($\frac{n_2}{n} \geq \frac{n_1}{n}$ AND either threshold above)	37.6%	17.9%

Table 11 Average number of matching correct answers generated out of 10 chances with and without the diagnosed mistake

	With Explanation	Without Explanation
Correct Diagnosis	4.198880 (SD = 3.987204)	3.010271 (SD = 3.668620)
Incorrect Diagnosis	1.775424 (SD = 3.152235)	1.631356 (SD = 2.881288)

When GPT-4 was unable to correctly describe an issue with the student's response, including the error explanation did not drastically increase the number of matches

cific diagnosis is wrong. Combining all the criteria, we see that 37.6% of accurate explanations passed these tests while only 17.9% of inaccurate explanations met the criteria. This suggests that accurate error diagnosis is a strong indicator of whether an explanation will actually help the simulated GPT-3.5 student update to the correct answer. These methods could potentially flag an incorrect diagnosis in the pipeline before it becomes an incorrect hint.

Table 11 shows that providing an explanation generally improved the model's ability to correct the student's response, but the benefit was greater with an accurate explanation. When GPT-4 correctly identified the student's mistake, the average number of correct revisions increased from 3 to 4.2 when the explanation was included. However, the standard deviations are quite large, indicating inconsistent performance whether or not the explanation was included. In contrast, when the diagnosis was incorrect, the explanation had a minimal effect. The average slightly improved from 1.63 to 1.78. The standard deviations are somewhat smaller as well, suggesting that performance was not only weaker overall, but consistently worse. This suggests that accurate explanations improved the simulated student performance, and could help real students correct their mistakes. Meanwhile, an incorrect explanation adds little value.

Figure 12 shows how frequently GPT-3.5 was able to update the student answers according to the kind of mistake the student made, comparing success with and without

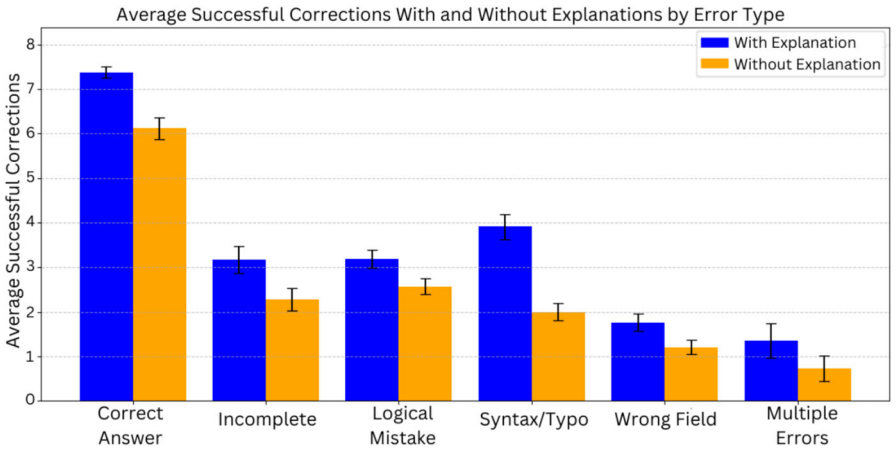

Fig. 12 Average number of successful corrections with and without the explanation by Error Type. Error bars represent the standard error of the mean

the GPT-4-generated explanation of the student's mistake. Across every error category, explanations led to more successful corrections. The largest gains occurred with syntax/typo errors, potentially because the responses are correct save for an incorrect character or two. The LLM saw the most success with answers that were already correct, though notably more success with an explanation. For more complex errors, like logical mistakes, incomplete responses, out-of-order problem-solving, and combinations of errors, the overall correction rates were lower. As noted in Study 1, error explanations rarely noted multiple student errors when present and these are the errors where GPT-3.5 had the least success.

Discussion

Our results showed that only 40.7% of hints were helpful enough that GPT-4 was able to adjust the incorrect student response to the correct one, and 30.2% were informative enough that GPT-4 could predict the incorrect student answer based on the feedback. We see similar results when comparing GPT-4's ability to update the student's answer with and without an explanation of what went wrong, where only 38% of the explanations were helpful enough to update more frequently. These figures suggest that either the GPT-generated feedback is not as informative or helpful as originally assessed, or these evaluation metrics do not align with human evaluation.

The relatively low pass rates of incorrect hints does indicate that GPT-4 is following the incorrect directions to update the student's answer, rather than solving the problem directly. This suggests that the evaluation methods are successfully identifying when feedback is misleading or inaccurate. These methods appear to be only assessing the logical consistency and applicability of the feedback. However, targeted feedback, which directly addresses specific student errors, only passed these tests 30% of the time. Consistently, bottom-out hints perform best in passing the evaluation metrics. As these hints usually contain both the correct answer and the student's answer, GPT-4 can succeed by regurgitating prompt information rather than following the steps instructed by the hint. The feedback that GPT-4 finds the most helpful with these metrics is feedback that gives away the answer.

The low pass rates for both general feedback and targeted hints could be explained by two factors. Students can make multiple mistakes in each answer and the short 1-2 sentence hint only targets one mistake, making it hard to update the student's answer to a correct answer. And, while the targeted feedback does call out the student's mistake, the full extent of the mistake cannot be anticipated, making it difficult to predict the exact student response.

When students make multiple mistakes, such as a student entering $(x + 4)(3 + 3)$ when the correct answer is $(x + 4)(x + 3) = 0$, the student has swapped one x for a 3 (Syntax/Typo error) and they are missing the $= 0$ to turn this expression into an equation (Incomplete error). Only one mistake was included in the diagnosis: "*The student mistook 'x+3' as '3+3' when factoring the trinomial. This suggests a simple transcription error or misunderstanding in the process of factoring.*". The feedback then only addressed this typo: "*Check your factored form again and think about how the 'x' term should be included.*" Subsequently, GPT-4 was unable to both adjust the

incorrect answer and predict the other mistake, even if this feedback could lead a student to taking a step toward a correct answer.

Another type of student answer that is hard to predict is an excess of characters. When a student answers `673^{^{^{14}}}+` with four exponent carets, the exact format of the student answer is difficult to predict unless it is explicitly stated in the feedback. Feedback that asks the student to try again with only one caret is clear about what mistake the student has made but it is not clear that the student used more than two carats. For another layer of difficulty, the correct answer is 673^{33}. The student has added the exponents instead of multiplying them, making it nearly impossible to capture the entirety of the student's mistakes in a short sentence in a way that their answer is recoverable.

When we gave GPT-4 ten opportunities to adjust the incorrect answers and predict the original mistakes, the results showed minimal improvement in the pass rates. Rather than converging on a correct answer, GPT-4 often defaulted to refusing to predict or settled on a plausible but incorrect answer. The average count of correct matches across multiple attempts remained relatively low, and never greater than half. Giving GPT-4 more opportunities to improve did little to improve its ability to pass these tests.

One interesting finding is that GPT-3.5 performed better on student responses that GPT-4 had previously diagnosed correctly, even when no explanation was provided. This suggests that these particular student responses are inherently easier to interpret and revise. It is likely that these student answers fit more cleanly into patterns that GPT models have in their training data. In contrast, when GPT-4 misdiagnosed a student's mistake, including the (incorrect) explanation did not improve its ability to correct student errors. This suggests that harder-to-diagnose responses are also harder to fix, regardless of additional context. Similar performance across both conditions – correct diagnosis versus incorrect – indicates that there is something inherent to student responses that make them easier or harder for GPT models to process. When GPT-4 can diagnose the mistake, reinforcing GPT-3.5 with an explanation is not as necessary for the model to generate the correct answer. When GPT-4 cannot diagnose the mistake, GPT-3.5 will also not be able to generate the correct response. Certain student responses are easier for GPT models to update because they align with patterns the models can handle, while others require a deeper level of contextual understanding.

One of the key limitations of these evaluation methods is the mismatch between human judgment and automated evaluation methods. While we approved of the targeted feedback generated by GPT-4, it often failed to pass the automated tests. The evaluation criteria are often overly eager to reject feedback. The feedback that we believed was personalized and focused on the student's specific mistake was frequently rejected by the automated methods. This discrepancy suggests that the evaluation system may not be fully capturing the value of the feedback. The main limitation of these evaluation metrics is that they only evaluate if the student could use the feedback to turn an incorrect answer into a correct answer. As we have seen, student errors often compound and the feedback usually address only one error. The evaluation should capture if the feedback could move a student toward the correct answer – if the fixed answer is 'less incorrect' than it used to be, rather than matching against full correctness.

In addition, the low pass rates of the hints raise concerns about the practicality of deploying this system in real educational settings. While the promise of instant, per-

sonalized hints is appealing, the reality is that hints would be delivered too infrequently and the quality of the hints would be too inconsistent to meet the expectations of a tutoring system. The pass rates for hints, even after giving GPT-4 multiple attempts, were not high enough to guarantee reliable feedback for students. The most enticing part of using a large language model to generate feedback is that it can be instantaneous and personalized, but the feedback is only helpful if we can accurately assess its quality. These results demonstrate that neither GPT-3.5 nor GPT-4 are currently reliable at determining whether a hint is helpful or informative.

Discussion - Putting it All Together

These studies set out to investigate three core research questions to assess the ability of LLMs to support ITS. Across these three studies, we see that GPT-generated feedback has the potential to address scalability challenges in providing personalized feedback. We found that:

- GPT-4 can diagnose student errors with an average accuracy of 87.8%.
- Only 4.8% of feedback generated by GPT-4 contained incorrect directions.
- Though human evaluators approved of over 90% of the generated feedback, more than half the feedback failed to pass LLM-based evaluation metrics, and the metrics did not fail all incorrect hints.

These results demonstrate how LLMs could be used to automate feedback creation within large-scale educational systems. Large language models have the potential to:

- Enhance error identification in ITS to give a more personalized, specific diagnosis.
- Provide specific directions on how to correct each student's unique error.
- Pick up where ITSs leave off, creating adaptive feedback when student interactions extend beyond what human instructional designers can prepare for.

The biggest challenge for this kind of feedback is how to deliver it to students without requiring human intervention for every interaction.

Feedback that is targeted, context-specific, and personalized is not passing the automated tests as frequently as human evaluation says it should. There is a gap between the quality of feedback GPT-4 can generate and the ability of the evaluation methods tested here to properly assess it. The misalignment between human judgment and automated methods indicates that the feedback validation process needs refining. Without a reliable way to evaluate the feedback, there is a risk of delivering incorrect feedback to students. While GPT-4 has the potential to scale feedback delivery, we need more robust assessment mechanisms to ensure that students consistently receive accurate guidance.

In addition, this study used data from a College Algebra course, a structured subject with well-defined problem types and lots of written internet explanations. While our results show promise in math-focused ITS environments, further research is needed to determine whether similar outcomes would hold across other subjects or with different language models. GPT-4's ability to generate relevant, accurate feedback is encouraging. We anticipate that LLMs trained on similar datasets would perform comparably

on structured domains, but empirical validation is necessary. Further research should confirm the consistency of these findings across academic subjects and model architectures.

Future work on LLM-based feedback can explore methods to improve both the quality of feedback and its alignment with pedagogical best practices. Given the success of additional context to mistake diagnosis, one improvement would be to add more context. Instead of only seeing the student's response in isolation, giving the LLM historical student interaction data may help the LLM narrow into specific misconceptions, or identify behaviors like hint abuse or guess-and-check. If the designer incorporated information about the student's previous entries or other incorrect answers into the prompt, the LLM might be able to generate more personalized feedback. The performance of GPT-4 depended heavily on the design and content of the prompt, which included detailed interface information and skill estimates. This level of prompt engineering may not generalize to systems where such metadata is unavailable.

We also want to explore additional techniques for quality assurance. Rather than assessing the LLM's ability to turn an incorrect answer into a correct one, perhaps measuring improvement in the incorrect answer would be a better gauge for helpfulness. Relaxing the constraints would allow more hints to pass the quality checks, but it may also allow more incorrect responses to reach students. If we can better separate high-quality hints from incorrect or unhelpful ones, ITS could more easily provide personalized support to a large number of students – achieving scalable feedback without sacrificing accuracy. Importantly, we did not evaluate learning outcomes with actual students. While the feedback appears helpful and appropriately targeted, further studies involving learners are needed to validate its pedagogical impact. It is possible that students are resilient to the small amount of incorrect feedback that GPT-4 generates. By assessing the impact these hints have on student learning, we can better tailor the quality control metrics to remove hints students find less helpful.

One common method for validating generated responses is to run the same prompt multiple times and select the most frequent (modal) answer. However, using the modal answer for feedback evaluation was not a significant improvement upon a single answer. A more effective approach might be implementing a feedback loop where the model revises its feedback based on additional information (such as the helpfulness evaluation results) to provide a more accurate response. Some more recent models like o1 https://openai.com/index/introducing-openai-o1-preview/ may already have these self-referential mechanisms. While feedback quality was generally high, there remains a non-trivial risk of incorrect hints. Until automated validation improves, human oversight or hybrid approaches may still be necessary to ensure instructional integrity.

Finally, it is important to align LLM-based instruction with pedagogical best practices. Although GPT-4 frequently produces accurate and well-structured hints, it lacks an internal model of instruction. Embedding LLMs within the structured framework of ITSs offers a way to constrain feedback generation and ensure alignment with learning goals. Future work should evaluate the impact of such feedback on student learning and continue refining quality control mechanisms.

Conclusion

While the LLM-based system as a whole demonstrated a reasonable ability to diagnose student errors, with an average accuracy above 80%, its performance varied significantly depending on the problem type, complexity, and student error. GPT-4 performed well with simpler, shorter problems like exponents and radical operations but struggled with more involved tasks that required multiple steps or context beyond the surface level. It also struggled to describe student responses when there was more than one error in the answer. GPT-4's effectiveness depends heavily on the problem structure and context provided.

However, even when GPT-4 misdiagnosed a student error, it was still able to craft useful, albeit sometimes generic, feedback. GPT-4 can craft relevant, albeit generic, feedback when given enough of the problem context, without referencing the student's response. But, over 10% of the generated feedback contained incorrect directions or gave away the correct answer. While the feedback was comprehensible to a general audience and used a positive tone, it was also overly general, and perhaps no better than pre-written feedback.

We also see a need for more robust evaluation methods for generated feedback. In the Apprentice Tutors system, 20% of student interactions were incorrect answers, each of which is a learning opportunity. With only 40% of the hints passing either test, in practice, students would infrequently receive helpful, informative feedback to their errors, and only 30% of the hints that students would actually see would be personalized to student input. The ability to reliable identify incorrect or inappropriate feedback, such as a bottom-out hint received before a high-level hint, is essential before unsupervised use.

We present these studies not only as an evaluation of current GPT capabilities, but also as a model for future research. By combining granular error analysis, prompt engineering, and automated analyses, we offer a framework for how others can evaluate and refine LLM-generated feedback. As language models continue to evolve, this kind of multifaceted evaluation will be key to safely and effectively integrating them into educational environments. Taken together, these studies show that while GPT-4 can offer scalable, personalized feedback, strong validation methods or domain-specific training will be required to avoid presenting learners with incorrect directions. Intelligent Tutoring Systems are already quite effective at enhancing student learning. Together with the natural language mastery of LLMs, we can build responsive, personalized learning systems.

Appendix A Prompts

A.1 Prompt Version 1

You will be provided with a problem, the correct answer to the problem, and a student's answer to the problem. Your task is to identify the specific mistake the student made and provide a helpful hint so they can correct their mistake.

Here is the problem the student was trying to solve:

<problem >
{{problem}}
</problem>
Here is the correct answer to the step:
<correct_answer >
{{correct}}
</correct_answer>
Here is the student's answer:
<student_answer>
{{student_input}}
</student_answer>

First, carefully analyze the student's answer and identify the specific mistake they made if they made a mistake, whether it was a logical error in their approach or a syntax error in how they wrote their answer. Describe the mistake in 1-2 sentences inside <analysis> tags.

Now, provide 1-2 sentences of feedback that will help guide the student to correct their mistake, without giving away the full answer. Write the feedback inside <feedback> tags.

Format your final output like this:
<analysis>
Insert your analysis of their mistake here
</analysis>
<feedback>
Insert your 1-2 sentence feedback here
</feedback>

Remember, the goal is to help the student learn and figure out how to solve the problem on their own, not to simply give them the correct answer. Provide targeted feedback that addresses their specific mistake.

A.2 Prompt Version 2

You will be provided with a problem, the correct answer to the problem, and a student's answer to the problem. Your task is to identify the specific mistake the student made and provide a helpful hint so they can correct their mistake.

Here is the problem the student was trying to solve:
<problem>
{{problem}}
</problem>
The initial problem is {{initial_problem}}.
The student entered a solution in the {{field}} field.
Here is the correct answer to the step:
<correct_answer>
{{correct}}
</correct_answer>
Here is the student's answer:

<student_answer>
{{student_input}}
</student_answer>

First, carefully analyze the student's answer and identify the specific mistake they made if they made a mistake, whether it was a logical error in their approach or a syntax error in how they wrote their answer. Describe the mistake in 1-2 sentences inside <analysis> tags.

Now, provide 1-2 sentences of feedback that will help guide the student to correct their mistake, without giving away the full answer. Write the feedback inside <feedback> tags.

Format your final output like this:

<analysis>
Insert your analysis of their mistake here
</analysis>
<feedback>
Insert your 1-2 sentence feedback here
</feedback>

Remember, the goal is to help the student learn and figure out how to solve the problem on their own, not to simply give them the correct answer. Provide targeted feedback that addresses their specific mistake.

A.3 Final Prompt

You will be provided with a problem, the correct answer to the problem, and a student's answer to the problem. Your task is to identify the specific mistake the student made and provide a helpful hint so they can correct their mistake.

Here is the problem the student was trying to solve:

<problem>
{{problem}}
</problem>

The initial problem is {{initial_problem}}.

The student entered a solution in the {{field}} field.

Here is the correct answer to the step:

<correct_answer>
{{correct}}
</correct_answer>

Here is the student's answer:

<student_answer>
{{student_input}}
</student_answer>

We believe there is a {{kc_estimate}} probability that the student has learned the skill "{{skill_name}}".

First, carefully analyze the student's answer and identify the specific mistake they made if they made a mistake, whether it was a logical error in their approach or a

syntax error in how they wrote their answer. Describe the mistake in 1-2 sentences inside <analysis> tags.

Now, provide 1-2 sentences of feedback that will help guide the student to correct their mistake, without giving away the full answer. Write the feedback inside <feedback> tags.

Format your final output like this:

<analysis>
Insert your analysis of their mistake here
</analysis>
<feedback>
Insert your 1-2 sentence feedback here
</feedback>

Remember, the goal is to help the student learn and figure out how to solve the problem on their own, not to simply give them the correct answer. Provide targeted feedback that addresses their specific mistake.

A.4 Evaluation - Adjust Answer Prompt

Here is the problem the student was trying to solve:

<problem>
{{problem_html}}
</problem>
The initial problem is {{initial_problem}}.
The student entered a solution in the {{field}} field.
A student gives the answer: "{{student_input}}"
Their teacher gives this feedback: "{{feedback}}"

Using this feedback, describe how to adjust the student's answer, then adjust the student's answer.

Write your updated answer at the end of the output using the template: "Updated Answer: [updated answer]". Only include the answer in its final form.

If it is not possible to update the student's answer based on the feedback, respond with "Invalid Feedback".

A.5 Evaluation - Predict Student Answer Prompt

Here is the problem the student was trying to solve:

<problem>
{{problem_html}}
</problem>
The initial problem is {{initial_problem}}.
The student entered a solution in the {{field}} field.
The correct answer to the {{field}} field is: "{{correct_answer}}"

A student gives an answer, to which their teacher gives the feedback: "{{feedback}}"

Using this feedback, describe how to predict what the student's answer was, then predict what the student's incorrect answer was.

Write your final prediction at the end of the output using the template: "Student Answer: [the student's answer]". Only include the answer in its final form.

If it is not possible to determine what the student's answer was based on the feedback, respond with "Insufficient Information".

Author Contributions J.R. and C.M. designed the study procedures. J.R. and A.A. analyzed, categorized, and interpreted the data. J.R. wrote the initial draft of the manuscript. C.M. and A.A. contributed comments and revisions. All authors reviewed and approved of the final manuscript. J.R. revised the manuscript based on reviewer suggestions. C.M. and A.A. contributed comments and revisions. All authors reviewed and approved of the final manuscript.

Funding This work was funded by the NSF National AI Research Institutes Program (Award #2112532). The opinions expressed are those of the authors and do not represent the views of the sponsoring agency.

Data Availability Student transaction data from the Apprentice Tutor platform used in this study will be made publicly available upon publishing at DataShop: https://pslcdatashop.web.cmu.edu/

Code Availability All prompts used are available in full in the Appendix.

Declarations

Competing Interests The authors declare no competing interests.

References

Aleven, V., McLaren, B. M., & Sewall, J. (2009). Scaling up programming by demonstration for intelligent tutoring systems development: An open-access web site for middle school mathematics learning. *IEEE Transactions on Learning Technologies, 2*(2), 64–78.

Aleven, V., Roll, I., McLaren, B. M., & Koedinger, K. R. (2016). Help helps, but only so much: Research on help seeking with intelligent tutoring systems. *International Journal of Artificial Intelligence in Education, 26*, 205–223.

Anderson, J., & Pelletier, R. (1991). *A development system for model-tracing tutors*

Azaiz, I., Kiesler, N., & Strickroth, S. (2024). Feedback-generation for programming exercises with gpt-4. In: *Proceedings of the 2024 on Innovation and Technology in Computer Science Education* (vol. 1, pp. 31–37).

Azevedo, R., Witherspoon, A., Graesser, A., McNamara, D., Chauncey, A., Siler, E., Cai, Z., Rus, V., & Lintean, M. (2009). Metatutor: Analyzing self-regulated learning in a tutoring system for biology. In: *Artificial Intelligence in Education*, (pp. 635–637). IOS Press

Berglund, L., Tong, M., Kaufmann, M., Balesni, M., Stickland, A.C., Korbak, T., & Evans, O. (2023). *The reversal curse: Llms trained on" a is b" fail to learn" b is a"*. arXiv:2309.12288

Butgereit, L. (2024). Using gpt-4 to tutor technical subjects in non-english languages in africa. In: *International conference on information technology-new generations*, (pp. 35–40). Springer

Butgereit, L., Martinus, H., & Abugosseisa, M.M. (2023). Prof pi: tutoring mathematics in arabic language using gpt-4 and whatsapp. In: *2023 IEEE 27th international conference on intelligent engineering systems (INES)*, (pp. 000161–000164). IEEE

Butler, A. C., Godbole, N., & Marsh, E. J. (2013). Explanation feedback is better than correct answer feedback for promoting transfer of learning. *Journal of Educational Psychology, 105*(2), 290.

Calo, T., & Maclellan, C. (2024). Towards educator-driven tutor authoring: Generative ai approaches for creating intelligent tutor interfaces. In: *Proceedings of the Eleventh ACM Conference on Learning@ Scale*, (pp. 305–309).

Choi, J.S., & Crossley, S.A. (2021). Arte: Automatic readability tool for english. *NLP Tools for the Social Sciences*. linguisticanalysistools.org.

Corbett, A. T., & Anderson, J. R. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-adapted Interaction, 4*, 253–278.

Crossley, S. A., & McNamara, D. S. (2008). Assessing l2 reading texts at the intermediate level: An approximate replication of crossley, louwerse, mccarthy & mcnamara (2007). *Language Teaching, 41*(3), 409–429.

Crossley, S. A., Skalicky, S., & Dascalu, M. (2019). Moving beyond classic readability formulas: New methods and new models. *Journal of Research in Reading, 42*(3–4), 541–561.

Deeva, G., Bogdanova, D., Serral, E., Snoeck, M., & De Weerdt, J. (2021). A review of automated feedback systems for learners: Classification framework, challenges and opportunities. *Computers & Education, 162*, 104094.

Dziri, N., Lu, X., Sclar, M., Li, X.L., Jiang, L., Lin, B.Y., Welleck, S., West, P., Bhagavatula, C., Le Bras, R., et al. (2024). Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, *36*.

Enders, N., Gaschler, R., & Kubik, V. (2021). Online quizzes with closed questions in formal assessment: How elaborate feedback can promote learning. *Psychology Learning & Teaching, 20*(1), 91–106.

Erickson, D., Holderness, D. K., Olsen, K. J., & Thornock, T. A. (2022). Feedback with feeling? how emotional language in feedback affects individual performance. *Accounting, Organizations and Society, 99*, 101329. https://doi.org/10.1016/j.aos.2021.101329

Finn, B., Thomas, R., & Rawson, K. A. (2018). Learning more from feedback: Elaborating feedback with examples enhances concept learning. *Learning and Instruction, 54*, 104–113.

Frieder, S., Pinchetti, L., Griffiths, R.-R., Salvatori, T., Lukasiewicz, T., Petersen, P., & Berner, J. (2024). Mathematical capabilities of chatgpt. *Advances in Neural Information Processing Systems*, *36*

Gendron, G., Bao, Q., Witbrock, M., & Dobbie, G. (2023). *Large language models are not strong abstract reasoners*. arXiv:2305.19555

Gilman, D. A. (1969). Comparison of several feedback methods for correcting errors by computer-assisted instruction. *Journal of Educational Psychology, 60*(6p1), 503.

Hattie, J., & Timperley, H. (2007). The power of feedback. *Review of Educational Research, 77*(1), 81–112.

Heffernan, N. T., & Heffernan, C. L. (2014). The assistments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education, 24*, 470–497.

Howard, N. R. (2021). "how did i do?": Giving learners effective and affective feedback. *Educational Technology Research and Development, 69*, 123–126.

Johnson, W. L. (1990). Understanding and debugging novice programs. *Artificial Intelligence, 42*(1), 51–97.

Knifong, J. D., & Holtan, B. (1976). An analysis of children's written solutions to word problems. *Journal for Research in Mathematics Education, 7*(2), 106–112.

Kumar, H., Rothschild, D.M., Goldstein, D.G., & Hofman, J.M. (2023). *Math education with large language models: Peril or promise*? Available at SSRN 4641653

Lee, K., Firat, O., Agarwal, A., Fannjiang, C., & Sussillo, D. (2018). *Hallucinations in neural machine translation*

Loria, S., et al. (2018). textblob documentation. *Release 0.15, 2*(8), 269.

Lovett, M.C. (1998). Cognitive task analysis in service of intelligent tutoring system design: A case study in statistics. In: *International Conference on Intelligent Tutoring Systems*, (pp. 234–243). Springer

Lu, X., & Wang, X. (2024). Generative students: using llm-simulated student profiles to support question item evaluation. In: *Proceedings of the eleventh ACM conference on learning@ Scale*, (pp. 16–27)

MacLellan, C. J., & Koedinger, K. R. (2022). Domain-general tutor authoring with apprentice learner models. *International Journal of Artificial Intelligence in Education, 32*(1), 76–117.

Mandernach, B. J. (2005). Relative effectiveness of computer-based and human feedback for enhancing student learning. *The Journal of Educators Online, 2*(1), 1–17.

Marsh, E. J., Lozito, J. P., Umanath, S., Bjork, E. L., & Bjork, R. A. (2012). Using verification feedback to correct errors made on a multiple-choice test. *Memory, 20*(6), 645–653.

McKendree, J. (1990). Effective feedback content for tutoring complex skills. *Human-Computer Interaction, 5*(4), 381–413.

McNichols, H., Feng, W., Lee, J., Scarlatos, A., Smith, D., Woodhead, S., & Lan, A. (2023). *Automated distractor and feedback generation for math multiple-choice questions via in-context learning*. NeurIPS.

Mollick, E., & Mollick, L. (2024). Instructors as innovators: A future-focused approach to new ai learning opportunities, with prompts. arXiv:2407.05181

Nesbit, J.C., Adesope, O.O., Liu, Q., & Ma, W. (2014). How effective are intelligent tutoring systems in computer science education? In: *2014 IEEE 14th International Conference on Advanced Learning Technologies*, (pp. 99–103). IEEE

Newman, M. A. (1977). An analysis of sixth-grade pupil's error on written mathematical tasks. *Victorian Institute for Educational Research Bulletin, 39*, 31–43.

Olms, C., Jakstat, H. A., & Haak, R. (2016). The implementation of elaborative feedback for qualitative improvement of shade matching—a randomized study. *Journal of Esthetic and Restorative Dentistry, 28*(5), 277–286.

Paaßen, B., Hammer, B., Price, T. W., Barnes, T., Gross, S., & Pinkwart, N. (2018). The continuous hint factory - providing hints in vast and sparsely populated edit distance spaces. *Journal of Educational Data Mining,10*(2018), 1–35. arXiv:1708.06564

Pal Chowdhury, S., Zouhar, V., & Sachan, M. (2024). Autotutor meets large language models: A language model tutor with rich pedagogy and guardrails. In: *Proceedings of the Eleventh ACM Conference on Learning@ Scale*, (pp. 5–15).

Pardos, Z.A., Tang, M., Anastasopoulos, I., Sheel, S.K., & Zhang, E. (2023). Oatutor: An open-source adaptive tutoring system and curated content library for learning sciences research. In: *Proceedings of the 2023 Chi Conference on Human Factors in Computing Systems*, (pp. 1–17).

Pardos, Z. A., & Bhandari, S. (2024). Chatgpt-generated help produces learning gains equivalent to human tutor-authored help on mathematics skills. *Plos One, 19*(5), 0304013.

Phung, T., Pădurean, V.-A., Cambronero, J., Gulwani, S., Kohn, T., Majumdar, R., Singla, A., & Soares, G. (2023). Generative ai for programming education: Benchmarking chatgpt, gpt-4, and human tutors. In: *Proceedings of the 2023 ACM Conference on International Computing Education Research-Volume 2*, (pp. 41–42).

Phung, T., Pădurean, V.-A., Singh, A., Brooks, C., Cambronero, J., Gulwani, S., Singla, A., & Soares, G. (2024). Automating human tutor-style programming feedback: Leveraging gpt-4 tutor model for hint generation and gpt-3.5 student model for hint validation. In: *Proceedings of the 14th Learning Analytics and Knowledge Conference*, (pp. 12–23).

Rau, M. A., Michaelis, J. E., & Fay, N. (2015). Connection making between multiple graphical representations: A multi-methods approach for domain-specific grounding of an intelligent tutoring system for chemistry. *Computers & Education, 82*, 460–485.

Razzaq, L., Patvarczki, J., Almeida, S. F., Vartak, M., Feng, M., Heffernan, N. T., & Koedinger, K. R. (2009). The assistment builder: Supporting the life cycle of tutoring system content creation. *IEEE Transactions on Learning Technologies, 2*(2), 157–166.

Ritter, S., Anderson, J. R., Koedinger, K. R., & Corbett, A. (2007). Cognitive tutor: Applied research in mathematics education. *Psychonomic Bulletin & Review, 14*, 249–255.

Rivers, K., & Koedinger, K. R. (2017). Data-driven hint generation in vast solution spaces: a self-improving python programming tutor. *International Journal of Artificial Intelligence in Education, 27*, 37–64.

Roest, L., Keuning, H., & Jeuring, J. (2024). Next-step hint generation for introductory programming using large language models. In: *Proceedings of the 26th Australasian Computing Education Conference*, (pp. 144–153).

Shahri, H., Emad, M., Ibrahim, N., Rais, R.N.B., & Al-Fayoumi, Y. (2024). Elevating education through ai tutor: Utilizing gpt-4 for personalized learning. In: *2024 15th Annual undergraduate research conference on applied computing (URC)*, (pp. 1–5). IEEE

Shute, V. J. (2008). Focus on formative feedback. *Review of Educational Research, 78*(1), 153–189.

Smith, G., Gupta, A., & MacLellan, C. (2024). *Apprentice Tutor Builder: A Platform For Users to Create and Personalize Intelligent Tutors*. arxiv:2404.07883

Stamper, J., Barnes, T., Lehmann, L., & Croy, M. (2008). The hint factory: Automatic generation of contextualized help for existing computer aided instruction. In: *Proceedings of the 9th International Conference on Intelligent Tutoring Systems Young Researchers Track*, (pp. 71–78).

Tafazoli, D., María, E. G., & Abril, C. A. H. (2019). Intelligent language tutoring system: Integrating intelligent computer-assisted language learning into language education. *International Journal of Information and Communication Technology Education (IJICTE), 15*(3), 60–74.

Thomas, D.R. (2003). A general inductive approach for qualitative data analysis

Truong, T.H., Baldwin, T., Verspoor, K., & Cohn, T. (2023). Language models are not naysayers: An analysis of language models on negation benchmarks. arxiv:2306.08189

Valmeekam, K., Olmo, A., Sreedharan, S., & Kambhampati, S. (2022). Large language models still can't plan (a benchmark for llms on planning and reasoning about change). In: *NeurIPS 2022 Foundation Models for Decision Making Workshop*

Valmeekam, K., Marquez, M., Sreedharan, S., & Kambhampati, S. (2023). On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems, 36*, 75993–76005.

VanLehn, K. (2006). The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education, 16*(3), 227–265.

Wan, T., & Chen, Z. (2024). Exploring generative ai assisted feedback writing for students' written responses to a physics conceptual question with prompt engineering and few-shot learning. *Physical Review Physics Education Research, 20*(1), 010152.

Weitekamp, D., Harpstead, E., & Koedinger, K.R. (2020). An interaction design for machine teaching to develop ai tutors. In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, (pp. 1–11).

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems, 35*, 24824–24837.

Xiao, R., Hou, X., & Stamper, J. (2024). Exploring how multiple levels of gpt-generated programming hints support or disappoint novices. In: *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*. CHI '24. ACM. https://doi.org/10.1145/3613905.3650937

Yamkovenko, S. (2023). *Sal Khan's 2023 TED talk: AI in the classroom can transform education*. Khan Academy. Accessed: 26 -Sept-2024