Assessing Large Language Models for Automated Feedback Generation in Learning Programming Problem Solving

Priscylla Silva [1,3] Evandro Costa [2]

[1]Universidade de São Paulo [2]Universidade Federal de Alagoas [3]Instituto Federal de Alagas

Abstract

Providing effective feedback is important for student learning in programming problem-solving. In this sense, Large Language Models (LLMs) have emerged as potential tools to automate feedback generation. However, their reliability and ability to identify reasoning errors in student code remain not well understood. This study evaluates the performance of four LLMs (GPT-4o, GPT-4o mini, GPT-4-Turbo, and Gemini-1.5-pro) on a benchmark dataset of 45 student solutions. We assessed the models' capacity to provide accurate and insightful feedback, particularly in identifying reasoning mistakes. Our analysis reveals that 63% of feedback hints were accurate and complete, while 37% contained mistakes, including incorrect line identification, flawed explanations, or hallucinated issues. These findings highlight the potential and limitations of LLMs in programming education and underscore the need for improvements to enhance reliability and minimize risks in educational applications.

Context

- Providing **timely and accurate feedback** is crucial for learning programming.
- Large class sizes make it difficult for instructors to give **individualized feedback**.
- Large Language Models (LLMs) offer a promising alternative by generating natural-language feedback tailored to **students' needs**, but may **struggle with reasoning mistakes and hallucinated feedback**.

Goals

In this work, we systematically evaluate four LLMs to assess their ability to generate accurate and meaningful feedback for programming assignments.

Questions

- How accurately do LLMs determine whether a student's solution fulfills the requirements of a programming task?
- What are the common types of errors in student code that LLMs fail to identify?
- How well do different LLMs perform in identifying mistakes/bugs and generating useful feedback for student code?

Approach & Methodology

LLMs Evaluated

- **GPT-4o, GPT-4o-mini, GPT-4-Turbo, and Gemini-1.5-Pro**.
- Each model was tested on **45 student solutions** across **5 programming assignments**.
- **Evaluation Metrics**: Accuracy in identifying incorrect solutions and quality of feedback.

Dataset

- Student solutions collected from introductory programming courses.
- 45 solutions were submitted by 5 students, with some students providing multiple solutions for the same programming assignment.

Assignment	Total Solutions	Correct	Incorrect
Area of a Circle	14	05	09
Simple Sum	06	04	02
T-Shirts	10	02	08
Huaauhahhuahau	08	04	04
Grandpa's Balance	07	04	03
Total	45	19	26

Feedback

The prompt explicitly instructed the models to act as programming teachers, assess the correctness of the code, and generate JSON-formatted feedback. Feedback included a binary indicator (*is_correct*) and a list of hints, with each hint specifying:

- The line number of the issue,
- The code line containing the issue,
- A concise explanation of the problem.

More Information

Evaluation Setup

Two teaching assistants categorized feedback into:

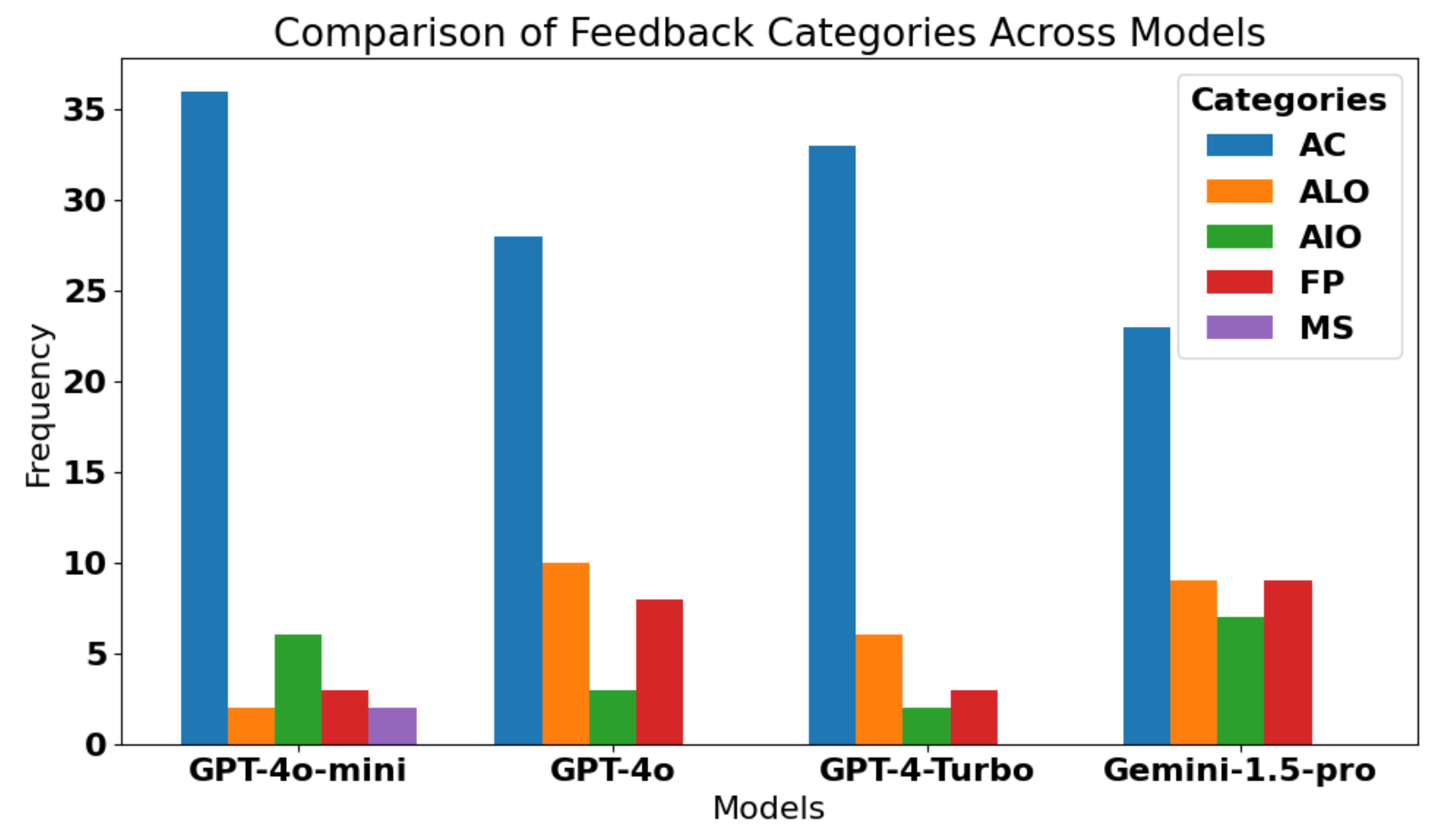
- **Accurate and Complete (AC)**: The feedback correctly identifies the appropriate line of code and explains the issue and/or how to resolve it. This is considered the ideal type of feedback, as it effectively supports students in identifying and understanding their mistakes.
- **Accurate Line Only (ALO)**: The feedback correctly identifies the line containing the issue but provides an incorrect or misleading explanation of the problem. Unlike False Positive (FP), the line identified does indeed contain a mistake, but the model misunderstands the nature of the issue.
- **Accurate Issue Only (AIO)**: The feedback describes the right issue but does not indicate the correct line of code. This misalignment can confuse students, as the explanation may not appear to relate to the line they are working on.
- **False Positive (FP)**: The feedback incorrectly identifies an issue where none exists. There is no actual error on the identified line, and the feedback message is entirely fake (hallucinated). This type of feedback can undermine student trust in the system.
- **Misleading Suggestions** (MS): The feedback points out a real error in the code, but both the line and the suggested fix are incorrect. While related to AIO, MS adds an additional layer of confusion by including a flawed or misleading resolution.

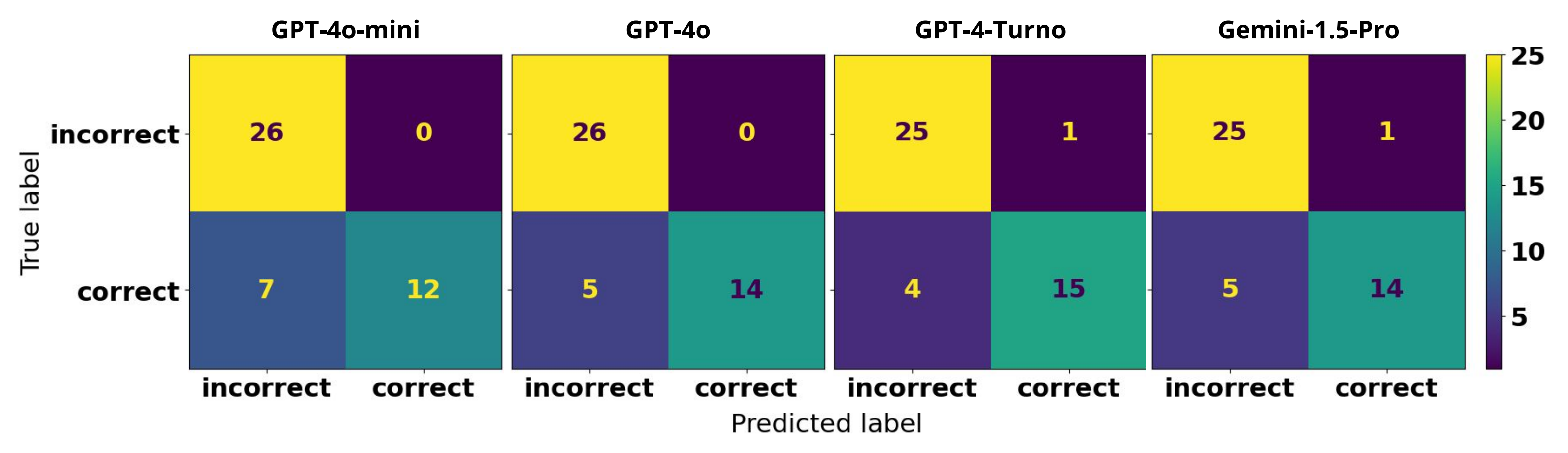
Key Findings & Results

- 63% of the feedback hints generated by the models were totally correct, meaning they accurately identified the problematic line and provided an appropriate explanation.
- 37% of the feedback contained some issues, such as pointing to the wrong line, providing an incorrect hint message, or hallucinating non-existent errors.
- Our findings reveal that although LLMs perform well in detecting syntactic and surface-level issues, they often fail to capture deeper reasoning mistakes in student code.

Table 1. Categorization of hints generated by each model.

Model	Categories					
	AC	ALO	AIO	FP	MS	Total
GPT-4o-mini	36	02	06	03	02	49
GPT-4o	28	10	03	08	00	49
GPT-4-Turbo	33	06	02	03	00	44
Gemini-1.5-pro	23	09	07	09	00	48
Total	120	27	18	23	02	190





Limitations & Future Work

- Limited dataset (45 solutions from 5 students) → Expand benchmark
- Prompt tuning → try different prompting strategies

Conclusion

- LLMs show potential for automated programming feedback but are prone to errors
- Models struggle with reasoning-based errors and hallucinate incorrect feedback