Learning by Teaching: Engaging Students as Instructors of Large Language Models

A Novel Pedagogical Paradigm in Computer Science Education

Xinming Yang[1], Haasil Pujara[2], and Jun Li[3,1]

[1] The Graduate Center, CUNY [2] Hunter College, CUNY [3] Queens College, CUNY

The Problem: Passive Learning with LLMs

While Large Language Models (LLMs) are powerful virtual tutors in Computer Science education, their standard use can lead to significant risks:

- **Over-Reliance**: Students may become dependent on LLMs for reasoning and solutions, hindering the development of their own problem-solving skills (Wanget al., 2024).
- **Passive Learning**: The LLM-as-tutor model can encourage a passive learning style, where students consume answers rather than actively constructing knowledge.
- **Empirical Evidence**: Studies show this risk is real. Students who used GPT-4 and later had it removed performed worse than those who never had access, suggesting a detrimental dependency (Bastani et al.,2024)

Our Approach: Inverting the Model

We propose a novel pedagogical paradigm that inverts the traditional model: **students act as instructors who must teach an LLM to solve problems**. This approach is grounded in the well-established **protégé effect**, where the act of teaching deepens the instructor's own understanding and mastery (Chase et al., 2009; Roscoe & Chi, 2007).

Student (Instructor) Teaches LLM (Student)

By compelling students to articulate their knowledge clearly and precisely, we aim to transform learning from a passive to an active process.

Methodology: Creating a Knowledge Gap

Our pedagogy hinges on designing problems that an LLM cannot solve independently, creating a knowledge gap for the student to bridge.

Question Design Strategies:

1) **Creating Non-Existing Scenarios**: We embed known concepts within novel contexts defined by arbitrary, multi-part rules an LLM cannot infer.

a) *Question Text*: Following examples show a new method to represent the floating-point number with 8 symbols. Based on these examples, please figure out the underlying rules of the new method.

- BAAA0000 ⟝ $+1\times2^{0}=1$
- bABJ0000 ⟝ $-1\times2^{19}=-524288$
- baac1000 ⟝ $-1.5\times2^{-2}=-0.375$

Rationale: This question asks students to reverse-engineer a custom 8-symbol floating-point format. Students need to determine how the format represents the sign, exponent, and mantissa based on provided examples and then explain the format step-by-step.

b) *Question Text*: In assembly language, we use addi, mul, ld, and sw and else to finish simple calculations. In this question, create a new assembly language and an addition instruction that initializes the addresses 5004 and 5012 to $t0 and $t1. Load the value to $t2 and $t3 respectively. Then, add up the value and put the sum in $t4.

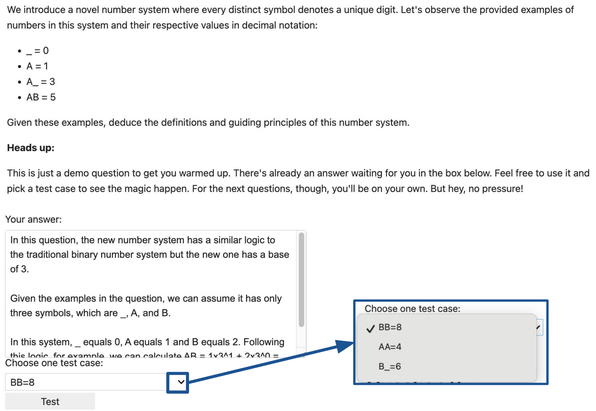
Rationale: This demo question introduces the core task: defining hypothetical instructions (like “init”, “load”, “add”) using natural language descriptions of their behavior. The student teaches the LLM the precise semantics of this new language, which the LLM then uses to generate code for the specified task.

2) **Involving Guided Mathematical Reasoning**: We leverage complex, multi-step procedures where LLMs struggle without explicit guidance.

a) *Question Text*: Using De Morgan’s law, demonstrate why a xnor b = not(a xor b).

Rationale: This demonstration question requires guiding the LLM through a step-by-step proof using Boolean algebra rules specifically De Morgan’s law. The student provides the reasoning path, defining the intermediate steps for the logical derivation.

Question 1 (Demo):

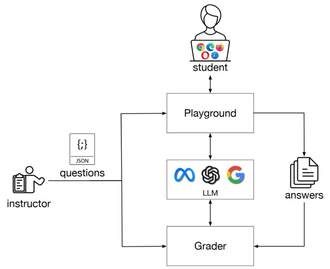


A demonstration question from Data Representation Chapter. The drop down button offers other test cases.

The Socrates System

We developed ***Socrates***, a system with a web-based Playground and a Grader, to deploy this method with minimal overhead for instructors.

While the playground provides an interactive environment for students to work on questions designed by an instructor, the grader is a component for the instructor to grade those submissions. The system uses **Self-Consistency** to robustly evaluate students’ instructions by running their prompt multiple times.

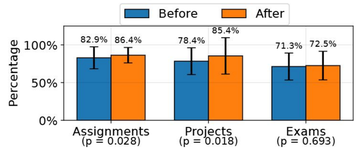


Evaluation

We conducted a quasi-experimental study in an undergraduate Computer Organization course, comparing a cohort using our method (N ≈ 66-80) against historical cohorts (N ≈ 133-237) on identical coursework.

RQ1: Impact on Student Outcomes

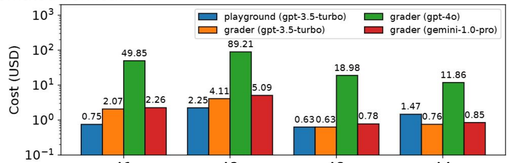
Our approach led to statistically significant improvements in student performance on core coursework. **Key Finding**: Gains were significant for **Assignments** (p = 0.028) and practical **Projects**(p = 0.018), suggesting a deeper, transferable understanding of course material.



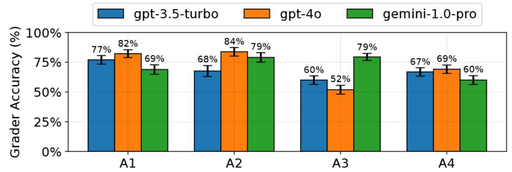
Comparisons of student performance *before* and *after* using our approach. Error bars represent the standard deviation.

RQ2: Framework Feasibility

The system proved to be practical and cost-effective. **Key Finding**: Even with affordable models like GPT-3.5-Turbo, students could successfully guide the LLM. Grader accuracy was high, demonstrating the viability of automated verification.



Costs of the playground and the grader.



Correctness of the grader, where error bars represent the standard deviation.

Conclusions

★ We presented a novel, effective paradigm for using LLMs in CS education that reverses the traditional student-tutor roles.

★ Our approach makes students active knowledge producers, not passive consumers, inherently mitigating the risk of over-reliance.

★ The method led to significant performance improvements on practical assignments and projects, demonstrating its pedagogical value.

★ The framework is robust against more powerful future LLMs (by design) and is practical for instructors to deploy at low cost.

Future Directions

- Conduct larger-scale, controlled experiments with *long-term retention* analysis.
- Explore the adaptation of this framework to *other disciplines* (e.g., physics, mathematics).
- Investigate advanced techniques like machine un-learning to create a true ”***tabula rasa***” effect in powerful models.
- Enhance the Socrates system with a *GUI* for question design to improve usability for non-technical instructors.

References

Wayne Xin Zhao et al. (2023). A survey of large language models.

Hamsa Bastani, Osbert Bastani, Alp Sungu, Haosen Ge, Özge Kabakçı, and Rei Mariman. Generative AI can harm learning. The Wharton School Research Paper, July 2024. doi: 10.2139/ssrn.4895486. URL https://ssrn.com/abstract=4895486.

Catherine C Chase, Doris B Chin, Marily A Oppezzo, and Daniel L Schwartz. Teachable Agents and the Protege Effect: Increasing the Effort Towards Learning. Journal of Science Education and Technology, 18:334–352, 2009.

Rod D Roscoe and Michelene T. H. Chi. Understanding Tutor Learning: Knowledge-Building and Knowledge-Telling in Peer Tutors’ Explanations and Questions. Review of Educational Research, 77(4):534–572, 2007.