Embracing Imperfection: Simulating Students with Diverse Cognitive Levels Using LLM-based Agents

Tao Wu[1], **Jingyuan Chen**[2†], **Wang Lin**[1], **Mengze Li**[3],
Yumeng Zhu[2], **Ang Li**[1], **Kun Kuang**[1], **Fei Wu**[1†],
[1] College of Computer Science and Technology, Zhejiang University,
[2] College of Education, Zhejiang University,
[3] Hong Kong University of Science and Technology
twu22@zju.edu.cn, jingyuanchen@zju.edu.cn

Abstract

Large language models (LLMs) are revolutionizing education, with LLM-based agents playing a key role in simulating student behavior. A major challenge in student simulation is modeling the diverse learning patterns of students at various cognitive levels. However, current LLMs, typically trained as "helpful assistants", target at generating perfect responses. As a result, they struggle to simulate students with diverse cognitive abilities, as they often produce overly advanced answers, missing the natural imperfections that characterize student learning and resulting in unrealistic simulations. To address this issue, we propose a training-free framework for student simulation. We begin by constructing a cognitive prototype for each student using a knowledge graph, which captures their understanding of concepts from past learning records. This prototype is then mapped to new tasks to predict student performance. Next, we simulate student solutions based on these predictions and iteratively refine them using a beam search method to better replicate realistic mistakes. To validate our approach, we construct the `Student_100` dataset, consisting of 100 students working on Python programming and $5,000$ learning records. Experimental results show that our method consistently outperforms baseline models, achieving 100% improvement in simulation accuracy and realism. Project page: https://mccartney01.github.io/student_sim.

1 Introduction

Artificial intelligence (AI) is transforming education by seamlessly integrating into teaching and learning (Xu et al., 2024a). Large language models (LLMs) play a central role in this shift, excelling in tasks such as personalized tutoring (Kwon et al., 2024), curriculum design (Erak et al., 2024), and adaptive assessment (Ross and Andreas, 2024).

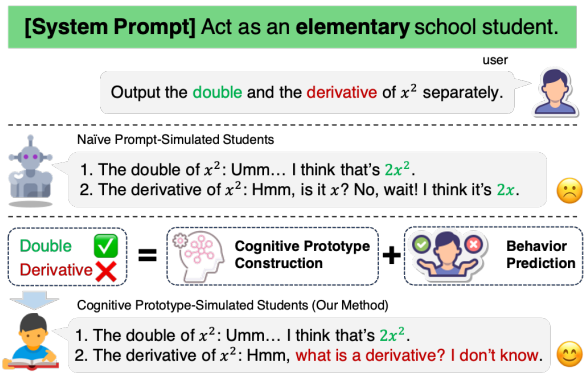

Figure 1: Existing LLM-based simulations struggle to accurately replicate behaviors at varying cognitive levels and produce overly advanced responses that undermine the validity of the simulation.

A key way LLMs drive these advancements is through simulation, where models typically adopt the role of a student. Such simulations provide researchers a cost-effective approach to evaluate teaching strategies (Yue et al., 2024), assess intelligent tutoring systems (Liu et al., 2024b), and enhance AI educational tools (Liu et al., 2024a).

Effective educational simulations must account for students' cognitive diversity (Xu et al., 2024b). High-performing students tend to demonstrate stronger comprehension and reasoning skills, while lower-performing students may make frequent errors. Thus, for LLM-based agents to effectively simulate student behavior, they must distinguish between different cognitive levels and generate responses that accurately reflect these differences. This means not only simulating the near-"perfect" solutions of high-achieving students but also generating "imperfections" of lower-performing students, including their typical mistakes.

However, recent studies highlight that current LLM-based agents struggle to replicate these "imperfections". They often overlook the typical mistakes made by lower-performing students and instead generate responses that are overly advanced

† Corresponding authors.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 9887–9908
July 27 - August 1, 2025

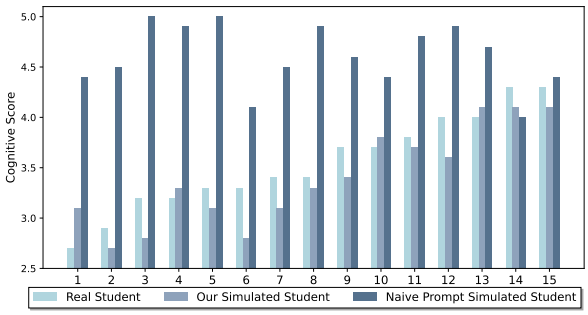

Figure 2: Cognitive scores of 15 different students. The naive prompt-based simulations tend to produce overly advanced responses. In contrast, our method adaptively simulates students and produces more approximated cognitive scores.

or unrealistic (He-Yueya et al., 2024; Markel et al., 2023; Aher et al., 2023). As shown in Figure 1, LLMs tend to overestimate the cognitive level of students with weaker abilities (*e.g.*, elementary school students), failing to replicate the expected error-prone behaviors that typically occur. Figure 2 further illustrates this issue, showing that LLMs frequently produce responses that exceed the cognitive capabilities of these students [1]. This limitation likely stems from the fact that LLMs are primarily trained to provide accurate and helpful solutions (Tan et al., 2023), rather than to mimic the nuanced error patterns characteristic of student learning.

A natural approach to addressing this challenge is fine-tuning LLMs on error-rich datasets. For instance, MalAlgoPy (Sonkar et al., 2024) defines 20 common equation transformation errors and trains models on misconception examples to reproduce these mistakes. While such fine-tuning can encourage models to generate errors, it also risks embedding incorrect knowledge, potentially degrading overall performance. Moreover, this method overlooks the personalized nature of student learning, as mistakes should be introduced adaptively based on a student's cognitive state.

To overcome these limitations, we propose a training-free approach based on cognitive prototypes to create LLM-based agents that accurately replicate student task-solving behavior. Our method begins by constructing a knowledge graph-based cognitive prototype from a student's past learning records, explicitly capturing their mastery of different knowledge concepts and serving as the foundation for precise cognitive-level analysis. Using this prototype, we predict how students will approach new tasks in a concept-aware manner. Specifically, our approach maps the cognitive prototype onto the new task, enabling a deeper, knowledge-concept-level assessment of the student's mastery. This allows for more precise predictions, including whether the student can correctly solve the task and what specific errors they are likely to make. Finally, we introduce a self-refinement approach to generate student solutions. The model continuously self-evaluates and iteratively refines its responses until they align precisely with the predicted behavior description. This not only ensures the generation of correct solutions but also improves the simulation of student mistakes, leading to more realistic, natural, and cognitively consistent student simulation.

To validate the effectiveness of our approach, we collect the `Student_100` dataset, consisting of $5,000$ learning records from 100 students working on Python programming tasks. Each student's records provide a unique experimental scenario for simulation. Experimental results show that our method significantly outperforms baseline models, producing more realistic and accurate student simulations. The contributions of this paper are:

- We identify the limitations of current LLMs in simulating students across different cognitive levels and collect a dataset for in-depth analysis.
- We propose a training-free framework using cognitive prototypes to generate realistic student behaviors and solutions at various cognitive levels.
- Extensive experimental results prove that our method overcomes the bias of behavioral simulations and can produce realistic simulations.

2 Related Work

2.1 LLM-based Education Simulation

Recent research explores using large language models (LLMs) to simulate educational roles, supporting teaching strategy evaluation (Markel et al., 2023; Luo et al., 2024) and AI teaching assistant development (Yue et al., 2024; Kwon et al., 2024). For instance, Wang et al. (2024b); Daheim et al. (2024) use LLMs for personalized error-correction, while Liu et al. (2024b) simulate students with different personalities in ITS. SocraticLM(Liu et al., 2024a) models students, teachers, and deans to support Socratic teaching and generate training data. Accurately replicating student behavior, including errors, is vital. Existing methods (Sonkar et al., 2024) improve error replication by training on error-

[1] Details about this figure are provided in Appendix B.

rich data, but risk embedding incorrect knowledge. To address this, we propose a training-free pipeline that predicts and refines student behavior simulations for more realistic and natural modeling.

2.2 Student Cognitive State Analysis

Diagnosing students' cognitive states and predicting their behavior are key challenges in education research. Existing methods fall into two categories: psychometrics-based and deep learning-based approaches (Jiang et al., 2022). Psychometrics-based methods (Stout, 2007; De La Torre, 2009) model the relationship between performance and knowledge state using empirical response functions, while deep learning methods (Su et al., 2018; Sun et al., 2022; Dong et al., 2025a,b; Lv et al., 2025; Zhou et al., 2025b; Lin et al., 2024) leverage neural networks but rely on implicit, parameterized knowledge, making it difficult to assess mastery of specific concepts (Wang et al., 2023b). This limitation restricts their effectiveness in student behavior prediction and simulation. To address this, we propose a knowledge graph-based approach to model students' cognitive prototypes. Our explicit, natural language-driven framework enables more precise behavior prediction and supports solution simulation.

3 Dataset Curation

Student simulation relies on the premise that a student's cognitive state remains stable over a short period. Based on this assumption, it effectively assesses the student's cognitive state using their past learning records and predicts their behavior on new tasks. The simulation then reproduces solution outcomes, such as accurately solving problems related to well-understood concepts or making reasonable errors on tasks involving less familiar concepts.

The simulation requires datasets with three key features: (1) a stable cognitive state for each student, (2) sequential task-solving records, and (3) detailed annotations, including task statements, solving behavior, and corresponding solutions. However, existing datasets often fail to meet all these criteria. Knowledge tracing datasets (Liu et al., 2023) primarily focus on correctness, lacking textual task statements, student behavior, and solutions (Hu et al., 2023). In contrast, error diagnosis datasets (Wang et al., 2024b; Daheim et al., 2024) offer rich textual information but lack annotated task-solving sequences, making it challenging to model students' cognitive states accurately.

To tackle these challenges, we develop a dataset, `Student_100`, specifically tailored for the student simulation task. The dataset originates from an online programming platform—PTA[2], comprising sequential programming records for each student. We choose programming as the task-solving scenario due to its complexity, error diversity and real-world applicability (Dai et al., 2024, 2025), making it a representative setting for student simulation[3]. To ensure a stable cognitive state, only records completed within a week are included in each sequence. Each raw record contains task statement, the solution (student-written code), and its correctness. To enhance the dataset, we recruit 10 well-trained annotators to provide detailed task descriptions and analyze student behavior (*i.e.*, evaluations of the solution) for each record.

For each student sequence, we select 40 records as "past learning records" and 10 as "simulation records". The final `Student_100` dataset comprises 100 students, each represented by 50 well-annotated task-solving records, forming a reliable foundation for student simulation tasks. Dataset statistics, a sample illustration and privacy protection are presented in Appendix A.

4 Methods

4.1 Overview

Student simulation requires accurately replicating the personalized behaviors and solutions of students with varying cognitive levels during task-solving. To achieve this, we propose a three-stage framework for student simulation. Firstly, given a student's M past learning records $P = \{P_i\}_{1\leq i\leq M} = \{(t_i, b_i, s_i)\}_{1\leq i\leq M}$, where t_i, b_i, and s_i represent tasks, behaviors, and solutions, respectively, we construct a cognitive prototype using a knowledge graph to explicitly capture the student's cognitive state (Section 4.2). Secondly, for a new task t_j ($M+1 \leq j \leq M+N$, where N is the number of tasks to be simulated), this prototype is applied to predict the student's expected behavior $\hat{b}_j$ (Section 4.3). Finally, based on the predicted behavior, the model generates the expected solution $\hat{s}_j$ using a beam search-based self-refinement method to ensure consistency (Section 4.4).

[2] https://pintia.cn/

[3] We provide a detailed discussion on the significance of programming tasks in Appendix A.3.

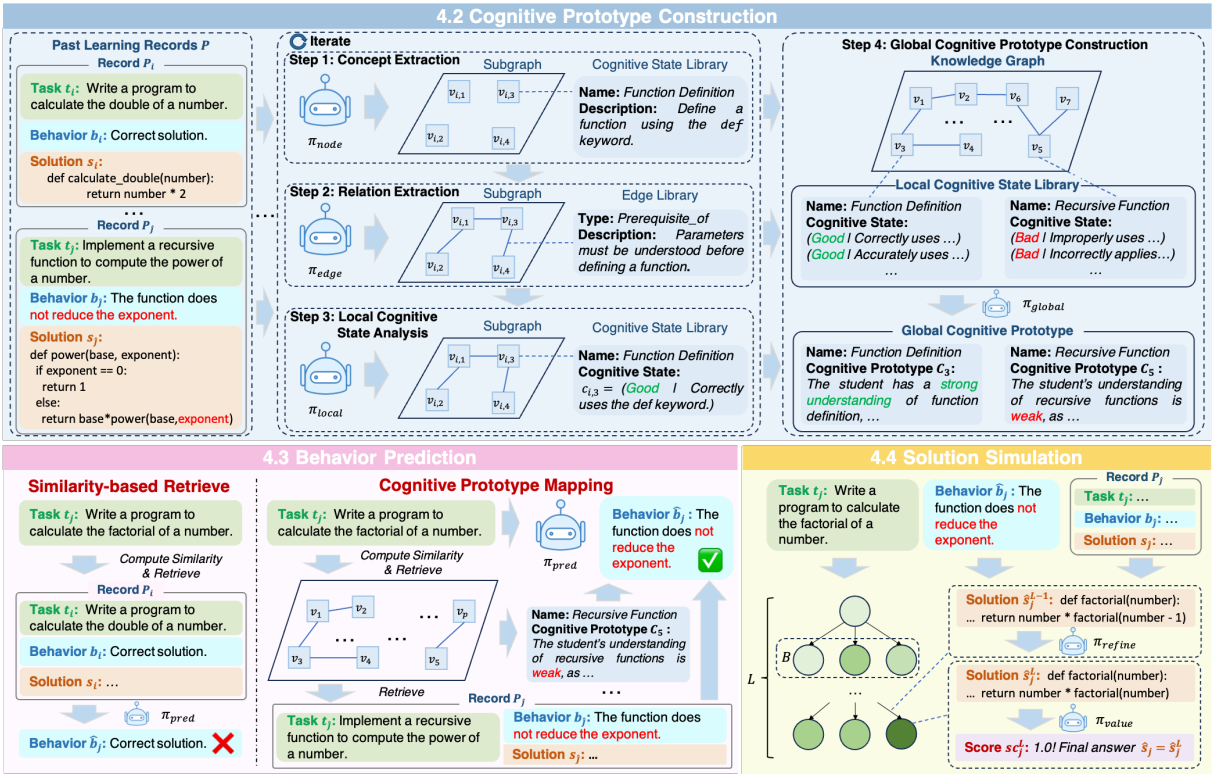

Figure 3: In the first stage, we construct a student cognitive prototype by iteratively building a knowledge graph from past learning records. This graph contains concepts, their relationships, and the local cognitive state libraries. After processing these records, we assess the student's mastery of each concept to create a global cognitive prototype. In the second stage, we use this prototype to predict behavior for new tasks. Unlike traditional methods, which rely on superficial similarities and risk incorrect retrieval, our approach maps the cognitive prototype to the task, identifying relevant concepts for accurate predictions. In the third stage, we employ a beam search-based self-refinement process to ensure the generated solution aligns with predicted behavior, improving simulation authenticity.

4.2 Cognitive Prototype Construction

A successful student simulation requires an interpretable representation of the student's cognitive state. Existing methods (Hu et al., 2023; Jiang et al., 2022) rely on implicit representations like neural networks, which lack interpretability and hinder behavior prediction. To address this, we propose constructing a cognitive prototype for each student using a knowledge graph based on their past learning records. As shown in Figure 3, each record $P_i = (t_i, b_i, s_i)$ is processed iteratively to extract relevant knowledge concepts and relationships, which are organized into a natural language-based knowledge graph to form the cognitive prototype. This iterative process involves 4 key steps:

Step 1: Concept Extraction. When students complete a task, they engage with knowledge concepts at various levels, from foundational concepts (*e.g.*, basic coding grammar) to advanced ones (*e.g.*, algorithm design). However, relying solely on the task statement t_i and student solution s_i often limits the extraction to foundational concepts, as advanced ones are rarely explicitly represented.

To address this, we use the model π_{desc} to generate a high-level task description $d_i = \pi_{desc}(t_i)$, which identifies reasoning strategies and advanced knowledge concepts required to complete the task. By integrating the task description d_i with the past learning record P_i, we employ π_{node} to extract comprehensive, multi-level knowledge concepts:

$$[v_{i,1},\ v_{i,2}, \cdots] = \pi_{node}(d_i, P_i) \quad (1)$$

where $[v_{i,1},\ v_{i,2}, \cdots]$ denote the extracted concepts.

Step 2: Relationship Extraction. As illustrated in Figure 3, after concept extraction, we analyze their relationships to form the edges of the knowledge graph. Inspired by previous work (Yang et al., 2024), we define four candidate relationships: "`Prerequisite_of`", "`Used_for`", "`Hyponym_of`", and "`Part_of`"[4]. The model π_{edge} then identifies related concept pairs and classifies them into one

[4]Detailed explanation about the 4 relationships are provided in Appendix C.1

of these types:

$$[e_{i,1},\ e_{i,2}, \cdots] = \pi_{edge}([v_{i,1},\ v_{i,2}, \cdots]) \quad (2)$$

where $[e_{i,1},\ e_{i,2}, \cdots]$ denote the extracted edges.

However, overlapping edges may arise and cause conflicts, where different relationship types are assigned to the same concept pair. To resolve this, we assign an edge library to each edge and handle inconsistencies after processing all past learning records. Details are provided in Appendix C.2. This ensures a consistent knowledge graph for further cognitive state analysis.

Step 3: Local Cognitive State Analysis. The knowledge graph constructed in previous steps includes only objective concepts and their relationships, without the student's personalized cognitive state. Thus, we evaluate the student's mastery of each concept based on their task performance. For each concept $v_{i,k}$, the model π_{local} examines the student's behavior and solution to detect any related mistakes, classifying mastery as either "`Good`" (no mistakes) or "`Bad`" (mistake detected).

$$[c_{i,1},\ \cdots] = \pi_{local}(d_i, P_i, [v_{i,1},\ \cdots]) \quad (3)$$

where $[c_{i,1},\ \cdots]$ denote the student's local cognitive state of each concept.

As shown in Figure 3, the local cognitive state for each concept is merged into its corresponding cognitive state library in the knowledge graph. This enriched library will support subsequent global cognitive state analysis, enabling a holistic assessment of the student's cognitive prototype.

Step 4: Global Cognitive Prototype Construction. After processing all past learning records, the knowledge graph is enriched with foundational nodes (*i.e.*, knowledge concepts), edges (*i.e.*, relationships), and local cognitive state analyses. To construct the global cognitive prototype, we evaluate the student's mastery of each concept based on the cognitive state library. For each concept v_k, the model π_{global} examines the frequency of "`Good`" and "`Bad`" classifications and generates a summary of the student's overall cognitive prototype:

$$C_k = \pi_{global}(v_k, [c_{k,1}, c_{k,2}, \cdots]) \quad (4)$$

where C_k denotes the student's global cognitive state of concept v_k.

This approach constructs a cognitive prototype for each student based on a knowledge graph. With nodes and edges expressed in clear natural language, the prototype will effectively support subsequent behavior prediction and solution simulation.

4.3 Behavior Prediction

Once the student's cognitive prototype is constructed, the next step is predicting their behavior on a new task. As shown in Figure 3, a common approach retrieves the most similar task from past learning records. However, this method often relies on textual similarity, overlooking underlying knowledge concepts. For example, when asked to "*Write a program to calculate the factorial of a number*", the model might retrieve a similar task like "*Write a program to calculate the double of a number*", which requires entirely different knowledge concepts and leads to inaccurate predictions.

To address this issue, we map the student's cognitive prototype onto the new task to predict their expected behavior. Specifically, for a given task t_j, the model π_{desc} generates a detailed description of the concepts assessed by the task. This description is compared to the nodes in the knowledge graph to compute similarity scores, and the top-p most relevant knowledge concepts are selected as the reference set, denoted as $[v_1, v_2, \cdots, v_p]$. Using these p concepts, we identify the past learning record $P_{\hat{j}}$ $(1 \leq \hat{j} \leq M)$ that includes the largest number of these concepts, which is deemed the most relevant task. By combining the student's overall cognitive states of these concepts with the context of $P_{\hat{j}}$, the model π_{pred} predicts the student's expected behavior of the given task:

$$\hat{b}_j = \pi_{pred}(t_j, [C_1, C_2, \cdots, C_p], P_{\hat{j}}) \quad (5)$$

where $\hat{b}_j$ denotes the predicted student behavior.

This behavior prediction leverages the student's cognitive prototype, ensuring predictions are based on deep conceptual understanding rather than superficial task statement similarities, and laying the foundation for further solution simulation.

4.4 Solution Simulation

After predicting the expected student behavior $\hat{b}_j$ for the new task t_j, we combine it with the retrieved past learning record $P_{\hat{j}}$ to simulate the solution $\hat{s}_j$ that aligns with the predicted behavior. Specifically, if the predicted behavior reflects a correct approach, the solution accurately resolves the task. However, if the behavior indicates potential errors, the solution should incorporate natural, intentional mistakes consistent with the predicted behavior. As demonstrated in Figure 3, to further refine the coherence and quality of the simulated solution, we employ a beam search-based self-refinement

Algorithm 1 Beam Search-Based Self-Refinement

Input: Expected Behavior $\hat{b}_j$, Retrieved Past Learning Record $P_{\hat{j}}$, Current task t_j, Refine Model π_{refine}, Value Model π_{value}, Max Iteration L, Beam Size B, Threshold δ

Output: Simulated Solution $\hat{s}_j$

$\hat{s}_j^1 = \pi_{refine}(t_j, \hat{b}_j, P_{\hat{j}})$, $l = 1$, $sc = 0$
while $l \leq L$ and $sc < \delta$ **do**
  for each $k \in [1, 2, \cdots, B]$ **do**
    $\hat{s}_j^{l,k} \sim \pi_{refine}(t_j, \hat{b}_j, P_{\hat{j}}, \hat{s}_j^l)$
    $sc^{l,k} \sim \pi_{value}(t_j, \hat{b}_j, P_{\hat{j}}, \hat{s}_j^{l,k}))$
  end for
  $\hat{k} = \text{Argmax}\ (sc^{l,1}, sc^{l,2}, \cdots, sc^{l,B})$
  $\hat{s}_j^{l+1} = \hat{s}_j^{l,\hat{k}}$, $sc = sc^{l,\hat{k}}$, $l \leftarrow l + 1$
end while
return $\hat{s}_j = \hat{s}_j^l$

method inspired by existing research (Chen et al., 2024; Zhang et al., 2024).

As shown in Algorithm 1, the process begins with an initial weak solution $\hat{s}_j^1$, which may not fully align with the predicted behavior $\hat{b}_j$. Then the solution is iteratively refined over L steps. At each iteration l ($1 \leq l \leq L$), the model π_{refine} updates the current solution $\hat{s}_j^l$ by leveraging the expected behavior and the retrieved past learning record to improve alignment. During each refinement step, B candidate solutions are sampled to increase the likelihood of achieving a successful refinement.

Each candidate is evaluated by model π_{value}, which assigns a score between 0 and 1 based on its alignment with the predicted behavior. The candidate solution with highest score is selected as the output for the current iteration and becomes the starting point for the next refinement step.

The process continues until either the maximum number of iterations L is reached or the value score exceeds a predefined threshold δ. By employing this beam search-based self-refinement method, we generate a final simulated solution $\hat{s}_j$ that accurately aligns with the expected behavior.

5 Experiments

5.1 Experiment Setup

We list the key parameter settings for our method in Table 1. The same model is employed across all components of our approach (*i.e.,* π_{desc}, π_{node}, π_{edge}, π_{local}, π_{global}, π_{pred}, π_{refine}, and π_{value}) to ensure consistency. Experiments are conducted using four representative LLMs—LLaMA-3.3-70B (Dubey et al., 2024), Claude-3.5-Sonnet (Anthropic, 2024), GPT-3.5 (Ouyang et al., 2022), and GPT-4o (OpenAI, 2023)—spanning different model families and scales. The temperature is fixed at 0 for reproducibility, except for π_{refine}, where diverse sampling is enabled to increase the likelihood of successful solution refinement.

Parameter	M	N	p	L	B	δ
Value	40	10	5	3	2	0.9

Table 1: Key parameter settings of our method.

Datasets. Given the substantial computational and financial costs associated with student simulation—with around 100 different experimental settings, as discussed in Section 5.2 and 5.3, each requiring approximately 20 minutes per student and multiple API calls—we conduct our analytical experiments on a randomly selected subset of 15 students (`Student_15`). This subset allows for comprehensive testing while ensuring the feasibility. For the main end-to-end comparison, we extend our analysis to all students in the `Student_100`, ensuring that our results are both robust and representative as in Appendix E.1.

To further validate the effectiveness of our method beyond Python programming, we additionally construct two new student groups—each comprising 5 students—for Java and C++ programming, respectively. These 10 students are built using metadata derived from the Codenet dataset (Puri et al., 2021), demonstrating our method's adaptability to datasets from different programming environments. Detailed experimental results are provided in Appendix E.3.

Baselines. For behavior prediction, we compare our prototype mapping approach with five baselines: 1) *Random*, which randomly selects a record from past learning records as a reference; 2) *Similarity*, which selects a record based on task statement similarities; 3) *Level* (Benedetto et al., 2024), which estimates a student's ability level based on the accuracy of their past learning records; 4) *Level+Random*, which incorporates *Random* and *Level*; 5) *Level+Similarity*, which incorporates *Random* and *Similarity*.

For solution simulation, we compare our beam search-based self-refinement method with two baselines: 1) *IO* (Yao et al., 2023), which requires models to directly output the solution with simulation instruct; 2) *CoT* (Wei et al., 2022), which indicates a Chain-of-Thought approach.

Behavior Prediction		Random			Similarity			Level			Level+Random			Level+Similarity			Prototype Mapping		
Solution Simulation		IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine
LLaMA-3.3-70B-Instruct	**Acc**	0.37	0.37	0.37	0.41	0.41	0.41	0.39	0.39	0.39	0.4	0.4	0.4	0.43	0.43	0.43	**0.61**	**0.61**	**0.61**
	Con$_1$	2.29	2.29	2.29	2.45	2.45	2.45	2.3	2.3	2.3	2.23	2.23	2.23	2.41	2.41	2.41	**2.99**	**2.99**	**2.99**
	Con$_2$	1.7	1.79	1.8	2.07	2.03	1.99	2.12	2.2	1.93	1.75	1.75	1.87	2.03	2.03	2.08	2.52	2.49	**2.69**
Claude-3.5-Sonnet	**Acc**	0.53	0.53	0.53	0.61	0.61	0.61	0.42	0.42	0.42	0.49	0.49	0.49	0.47	0.47	0.47	**0.65**	**0.65**	**0.65**
	Con$_1$	2.74	2.74	2.74	3.03	3.03	3.03	2.24	2.24	2.24	2.51	2.51	2.51	2.44	2.44	2.44	**3.09**	**3.09**	**3.09**
	Con$_2$	2.57	2.43	2.53	2.8	2.75	2.82	1.45	1.77	1.51	2.14	1.99	2.16	2.38	2.33	2.26	2.91	2.71	**2.99**
GPT-3.5	**Acc**	0.37	0.37	0.37	0.44	0.44	0.44	0.54	0.54	0.54	0.45	0.45	0.45	0.47	0.47	0.47	**0.56**	**0.56**	**0.56**
	Con$_1$	2.36	2.36	2.36	2.51	2.51	2.51	2.88	2.88	2.88	2.61	2.61	2.61	2.66	2.66	2.66	**2.99**	**2.99**	**2.99**
	Con$_2$	3.27	3.33	3.47	3.34	3.35	3.39	3.33	3.35	3.37	3.31	3.34	3.3	3.17	3.33	3.35	3.24	3.41	**3.49**
GPT-4o	**Acc**	0.45	0.45	0.45	0.47	0.47	0.47	0.44	0.44	0.44	0.47	0.47	0.47	0.47	0.47	0.47	**0.94**	**0.94**	**0.94**
	Con$_1$	2.55	2.55	2.55	2.62	2.62	2.62	2.44	2.44	2.44	2.58	2.58	2.58	2.57	2.57	2.57	**3.77**	**3.77**	**3.77**
	Con$_2$	2.49	2.45	2.31	2.55	2.86	2.65	2.27	2.5	2.45	2.27	2.29	2.11	2.24	2.36	2.29	3.32	3.5	**3.65**

Table 2: End-to-end comparison across 6 behavior prediction and 3 solution simulation methods. *Acc* and *Con$_1$* metrics both evaluate behavior descriptions, yielding identical values for the same behavior prediction method.

	π_{desc}, π_{node}, π_{edge}, π_{local}	π_{global}	π_{pred}	π_{refine}	π_{value}	**LLaMA-3.3-70B**			**Claude-3.5-Sonnet**			**GPT-3.5**			**GPT-4o**		
						Acc	**Con$_1$**	**Con$_2$**	**Acc**	**Con$_1$**	**Con$_2$**	**Acc**	**Con$_1$**	**Con$_2$**	**Acc**	**Con$_1$**	**Con$_2$**
1				✓	✓	0.41	2.45	1.99	0.61	3.03	2.82	0.44	2.51	3.39	0.47	2.62	2.65
2	✓		✓	✓	✓	0.53	2.82	2.16	0.64	3.09	2.63	0.45	2.61	3.12	0.66	3.13	2.89
3	✓	✓		✓	✓	-	-	1.98	-	-	2.52	-	-	3.26	-	-	2.7
4	✓	✓	✓			0.61	2.99	2.49	0.65	3.09	2.71	0.56	2.99	3.41	0.94	3.77	3.5
5	✓	✓	✓	✓		0.61	2.99	2.61	0.65	3.09	2.96	0.56	2.99	3.21	0.94	3.77	3.52
6	✓	✓	✓	✓	✓	0.61	2.99	2.69	0.65	3.09	3.09	0.56	2.99	3.49	0.94	3.77	3.65

Table 3: Ablation study of each component of our method. *Acc* and *Con$_1$* assess behavior prediction only and are thus unaffected by the changes in the solution simulation components (*i.e.*, π_{pred}, π_{refine}, and π_{value}).

Metrics. For behavior prediction, we first assess the model's accuracy (*Acc*) in determining whether a student can correctly solve the tasks. To evaluate the alignment between generated and ground truth behavior descriptions, we introduce an LLM-based metric (*Con$_1$*) that scores consistency on a scale from 1 to 5, using evaluations from o1-mini. Similarly, solution simulation is measured with *Con$_2$*, following the same o1-mini approach as *Con$_1$*.

Please note that since *Acc* and *Con$_1$* exclusively assess the accuracy and consistency of behavior prediction, their values are invariant across different solution simulation methods in Table 2 and 3. This design ensures a clean separation of evaluation for the two stages in our framework.

Please refer to Appendix D for experiment details and Appendix E for more experimental results.

5.2 End-to-End Performance Comparison

With 6 behavior prediction and 3 solution simulation methods, we conduct end-to-end comparison across all 18 unique configurations. The results, shown in Table 2, reveal the following key insights:

1) **Superior behavior prediction**. Our prototype mapping approach significantly outperforms existing methods, emphasizing the importance of precisely capturing a student's mastery of relevant concepts for more accurate behavior prediction.

2) **Enhanced solution simulation**. With our predicted behavior descriptions, our beam search-based self-refinement method consistently performs better, showing that iterative self-evaluation and optimization lead to more accurate solutions, consistent with the student's cognitive ability.

Beyond these improvements, the end-to-end comparison also reveals some innovative findings:

1) **Stronger LLMs benefit more from self-refinement**. We observe that more powerful models (*e.g.*, GPT-4o) exhibit greater performance gains with our method. We hypothesize that this is due to their superior self-evaluation capabilities, which allow them to generate more accurate assessments and constructive feedback throughout the refinement process. This iterative feedback mechanism systematically corrects imperfections in initial outputs, further enhancing simulation fidelity.

2) **Self-refinement relies on high-quality behavior descriptions**. While self-refinement generally improves solution simulation, we find that in some cases—particularly when behavior prediction quality is low—it underperforms compared to IO or CoT prompting. We hypothesize that this is because low-quality behavior descriptions can mislead the refinement process, causing the model to iteratively adjust solutions in the wrong direction, ultimately degrading simulation quality. This sug-

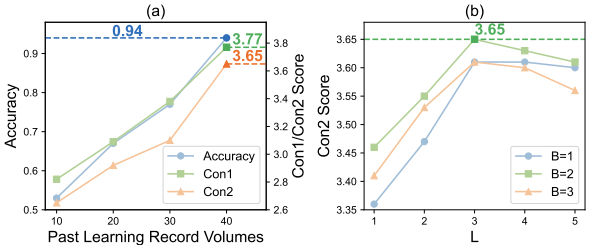

Figure 4: (a) Performance on different past learning record volumes. (b) Performance on different refinement iteration L and beam search sampling size B.

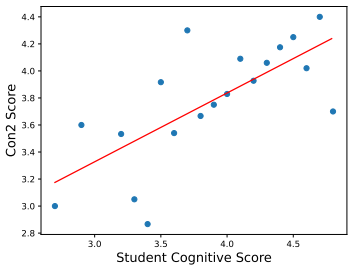

Figure 5: Simulation difficulty varies across students.

gests that self-refinement is most effective when built on accurate behavior predictions.

5.3 In-Depth Analysis

We further validate 4 vital issues as follows.

Effectiveness of each component. We evaluate the effectiveness of each component in Table 3. The components π_{desc}, π_{node}, π_{edge}, and π_{local} are considered as a whole, as ablating them individually would disrupt the entire graph-building process. 1) Removing the entire knowledge graph and relying solely on the most textually similar retrieved past learning record results in a performance drop (Row 1), validating *the importance of cognitive prototype construction.* 2) Without the global cognitive prototype construction, we use only the local cognitive state library for behavior prediction, leading to a decrease in performance (Row 2). This suggests that *global cognitive state construction is crucial to avoid overwhelming the model with excessive local records.* 3) Removing the behavior prediction component π_{pred} to directly generate student solutions leads to a performance drop (Row 3), highlighting *the importance of clear and accurate behavior descriptions in guiding solution simulation effectively.* 4) Removing the self-refinement process leads to poor performance in Row 4, indicating *the importance of iterative refinement*. However, Row 5 shows that even with only π_{refine}, without the self-evaluation component π_{value}, performance still decreases. This underscores that *self-refinement must be guided by self-evaluation to ensure the correct refinement direction.*

Method	Setting	Score
LLaMA-3.3-70B-Instruct		
The second best setting	Prototype Mapping + IO	2.88
Our method	Prototype Mapping + Refine	3.08
Claude-3.5-Sonnet		
The second best setting	Prototype Mapping + IO	3.24
Our method	Prototype Mapping + Refine	3.27
GPT-3.5		
The second best setting	Random + Refine	3.29
Our method	Prototype Mapping + Refine	3.3
GPT-4o		
The second best setting	Prototype Mapping + CoT	3.7
Our method	Prototype Mapping + Refine	3.75

Table 4: Human evaluation on solution simulation between our method and the second best setting.

Impact of past learning record volumes. As shown in Figure 4 (a), increasing the number of past learning records from 10 to 40 leads to consistent improvements in both behavior prediction and solution simulation. Specifically, the behavior prediction accuracy reaches 0.94 at 40 records, marking a substantial improvement. We attribute this trend to the enhanced accuracy of the cognitive prototype, which benefits from more past learning records that captures a more comprehensive view of the student's knowledge and behavior patterns. Notably, performance continues to improve steadily at 40 records, indicating that even larger past learning records could further refine student simulation quality. Investigating the effects of larger record volumes remains a direction for future research.

Impact of self-refinement iterations and beam search sampling size. We conduct a grid search to analyze the effects of the self-refinement iterations (L) and beam search sampling size (B), where L ranges from 1 to 5 and B ranges from 1 to 3. As shown in Figure 4 (b), the performance of solution simulation improves with an increasing iteration of refinements, highlighting the limitations of current LLMs in generating accurate student answers in a single pass. The self-refinement process effectively addresses initial weaknesses; however, when L exceeds 3, performance stabilizes or slightly declines. We speculate that this is because the models tend to over-correct the solutions, which introduces deviations from realistic solutions. Therefore, we select $L = 3$. A similar trend is observed for the beam size B: performance improves as B increases, but the gains become negligible beyond $B = 2$. To balance performance and simulation cost, we set the sampling size to $B = 2$.

Variability in simulation difficulty across stu-

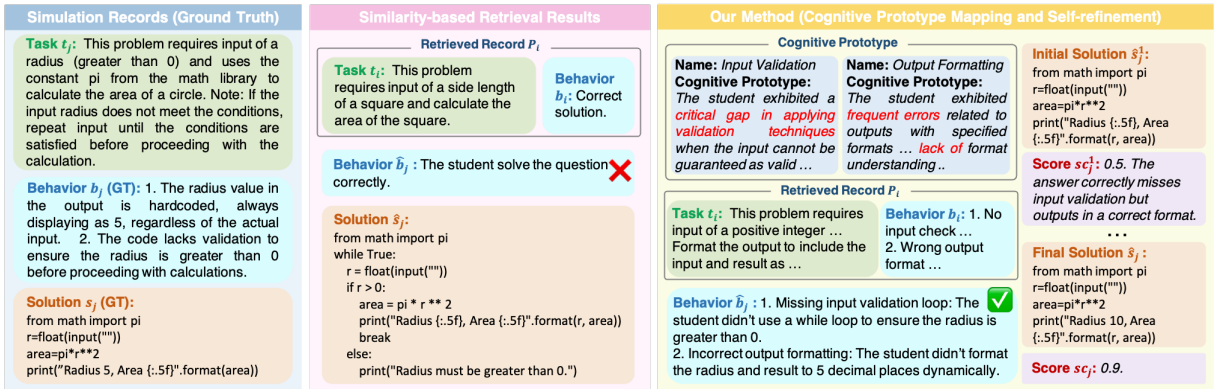

Figure 6: Examples of simulated results. Similarity-based retrieval methods rely on superficial task similarities, leading to inaccurate predictions. In contrast, our method maps the cognitive prototype to relevant concepts, ensuring accuracy. The self-refinement process then iteratively adjusts the solution for precise simulation.

dents. We further explore the difficulty differences in simulating individual students. We analyze the correlation between simulation quality (measured by the average Con_2 score) and student cognitive ability scores (described in Appendix A.2), as shown in Figure 5. The figure reveals a clear positive correlation between simulation quality and cognitive ability, indicating that simulating students with higher cognitive abilities is relatively easier.

This finding aligns with expectations—students with stronger cognitive abilities make fewer mistakes, and generating a fully correct solution is inherently easier than producing a realistic and natural simulation of a specific student's mistakes. This further highlights the challenge of accurately modeling students with lower cognitive abilities, as it requires the model to simulate not only correct responses but also plausible, individualized errors.

5.4 Human Evaluation

We further conduct a human evaluation study to demonstrate the effectiveness of our method. Given the substantial number of experimental settings involved in our comparisons and considering the cost of manual assessment, we compare our method against the second-best setting according to LLM evaluation results in Table 2.

We recruit 10 undergraduate students with solid Python programming skills to perform the evaluation. They rate each simulated solution on a 1–5 scale, consistent with the LLM-based evaluation rules, where higher scores indicate better simulation quality. The evaluation instructions provided to annotators are shown in Figure 15. Each solution is rated by two independent evaluators, and the final score is computed as their average.

As shown in Table 4, our method consistently receives higher human evaluation scores than the baseline across all model settings. This further validates the effectiveness and realism of our student simulation framework.

5.5 Case Study

As illustrated in Figure 6, similarity-based retrieval methods rely solely on textual similarities in task statements, leading to the retrieval of an inappropriate past learning record. This misleads the model into incorrectly predicting the student's behavior as "*The student solves the question correctly*", and consequently, generating a fully correct solution, which contradicts the real student's answer.

In contrast, our method maps the constructed student cognitive prototype to the task, accurately identifying the relevant knowledge concepts the student struggles with. By retrieving a correct past learning record, the model makes an accurate behavior prediction, explicitly describing the mistakes the student is likely to make. Building on this prediction, our self-refinement process iteratively adjusts the generated solution to reflect these predicted mistakes, ultimately producing a realistic and accurate simulation of the student's solution.

6 Conclusion

In this paper we introduce a training-free framework for student simulation. Our approach begins by constructing a cognitive prototype for each student based on past learning records to predict their behavior. It then employs a beam search-based self-refinement process to progressively improve the quality of simulated solutions. Experiments show that our framework simulates realistic and diverse students, enhancing the realism and utility of AI in education.

Limitations

In this section, we discuss the limitations of our work as follow:

- While our current validation is constrained to programming domains, primarily due to the accessibility of relevant data, the underlying simulation framework is theoretically applicable to a broader range of educational subjects, such as mathematics (Huang et al., 2025). We consider this a promising direction for future work.
- Our current simulation primarily focuses on textual and behavioral patterns without explicitly incorporating multimodal signals, such as visual or auditory cues (Wu et al., 2024; Wang et al., 2024a)that may also influence students' cognitive processes. While this unimodal setting enables clearer analysis of reasoning behaviors, we acknowledge that a more comprehensive understanding of student performance may benefit from integrating multimodal data. Exploring this dimension constitutes a valuable direction for future work.

Ethics Statement

As the use of large language models (LLMs) in education grows, it is crucial to consider the ethical implications of deploying such systems to simulate human-like student behaviors. While our work explores the potential of LLMs for simulating student cognitive states and generating educational scenarios, this research remains an early-stage exploration and is conducted solely within controlled experimental settings.

The simulated students in this study are designed for research and methodological evaluation rather than real-world teaching applications. These simulations are not intended to replace human students or educators but to provide insights into model capabilities. We strongly emphasize that any deployment of such models in educational contexts should involve careful oversight by educators to prevent misuse and to ensure ethical compliance.

Future work will focus on further improving the accuracy and realism of these simulations, reducing potential biases, and incorporating diverse perspectives to enhance their applicability. We remain committed to addressing ethical challenges and ensuring that our work supports responsible and beneficial advancements in AI-driven education.

Acknowledgements

This research was partially supported by grants from the National Natural Science Foundation of China (No.62037001, No.62307032), and the "Pioneer" and "Leading Goose" R&D Program of Zhejiang under Grant No. 2025C02022.

References

Gati V. Aher, Rosa I. Arriaga, and Adam Tauman Kalai. 2023. Using large language models to simulate multiple humans and replicate human subject studies. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 337–371. PMLR.

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku.

Luca Benedetto, Giovanni Aradelli, Antonia Donvito, Alberto Lucchetti, Andrea Cappelli, and Paula Buttery. 2024. Using llms to simulate students' responses to exam questions. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 11351–11368. Association for Computational Linguistics.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024. Alphamath almost zero: process supervision without process. *CoRR*, abs/2405.03553.

Nico Daheim, Jakub Macina, Manu Kapur, Iryna Gurevych, and Mrinmaya Sachan. 2024. Stepwise verification and remediation of student reasoning errors with large language model tutors. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 8386–8411. Association for Computational Linguistics.

Zhenlong Dai, Bingrui Chen, Zhuoluo Zhao, Xiu Tang, Sai Wu, Chang Yao, Zhipeng Gao, and Jingyuan Chen. 2025. Less is more: Adaptive program repair with bug localization and preference learning. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 128–136. AAAI Press.

Zhenlong Dai, Chang Yao, WenKang Han, Yuanying Yuanying, Zhipeng Gao, and Jingyuan Chen. 2024. Mpcoder: Multi-user personalized code generator with explicit and implicit style representation learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3765–3780. Association for Computational Linguistics.

Jimmy De La Torre. 2009. Dina model and parameter estimation: A didactic. *Journal of educational and behavioral statistics*, 34(1):115–130.

Zhiang Dong, Jingyuan Chen, and Fei Wu. 2025a. Knowledge is power: Harnessing large language models for enhanced cognitive diagnosis. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 164–172. AAAI Press.

Zhiang Dong, Liya Hu, Jingyuan Chen, Zhihua Wang, and Fei Wu. 2025b. Comprehend then predict: Prompting large language models for recommendation with semantic and collaborative data. *ACM Transactions on Information Systems*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Omar Erak, Omar Alhussein, Shimaa Naser, Nouf Al-abbasi, De Mi, and Sami Muhaidat. 2024. Large language model-driven curriculum design for mobile networks. *CoRR*, abs/2405.18039.

Joy He-Yueya, Noah D. Goodman, and Emma Brunskill. 2024. Evaluating and optimizing educational content with large language model judgments. In *Proceedings of the 17th International Conference on Educational Data Mining, EDM 2024, Atlanta, Georgia, USA, July 14-17, 2024*. International Educational Data Mining Society.

Liya Hu, Zhiang Dong, Jingyuan Chen, Guifeng Wang, Zhihua Wang, Zhou Zhao, and Fei Wu. 2023. Ptadisc: A cross-course dataset supporting personalized learning in cold-start scenarios. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Zihan Huang, Tao Wu, Wang Lin, Shengyu Zhang, Jingyuan Chen, and Fei Wu. 2025. Autogeo: Automating geometric image dataset creation for enhanced geometry understanding. *IEEE Transactions on Multimedia*.

Bo Jiang, Xinya Li, Shuhao Yang, Yaqi Kong, Wei Cheng, Chuanyan Hao, and Qiaomin Lin. 2022. Data-driven personalized learning path planning based on cognitive diagnostic assessments in moocs. *Applied Sciences*, 12(8):3982.

Soonwoo Kwon, Sojung Kim, Minju Park, Seunghyun Lee, and Kyuseok Kim. 2024. BIPED: pedagogically informed tutoring system for ESL education. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3389–3414. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Wang Lin, Yueying Feng, WenKang Han, Tao Jin, Zhou Zhao, Fei Wu, Chang Yao, and Jingyuan Chen. 2024. E^3: Exploring embodied emotion through A large-scale egocentric video dataset. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Wang Lin, Tao Jin, Wenwen Pan, Linjun Li, Xize Cheng, Ye Wang, and Zhou Zhao. 2023. TAVT: towards transferable audio-visual text generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 14983–14999. Association for Computational Linguistics.

Jiayu Liu, Zhenya Huang, Tong Xiao, Jing Sha, Jinze Wu, Qi Liu, Shijin Wang, and Enhong Chen. 2024a. Socraticlm: Exploring socratic personalized teaching with large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Zhengyuan Liu, Stella Xin Yin, Geyu Lin, and Nancy Chen. 2024b. Personality-aware student simulation for conversational intelligent tutoring systems. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 626–642. Association for Computational Linguistics.

Zitao Liu, Qiongqiong Liu, Jiahao Chen, Shuyan Huang, and Weiqi Luo. 2023. simplekt: A simple but tough-to-beat baseline for knowledge tracing. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Haohao Luo, Yang Deng, Ying Shen, See-Kiong Ng, and Tat-Seng Chua. 2024. Chain-of-exemplar: Enhancing distractor generation for multimodal educational question generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 7978–7993. Association for Computational Linguistics.

Xiangwei Lv, Guifeng Wang, Jingyuan Chen, Hejian Su, Zhiang Dong, Yumeng Zhu, Beishui Liao, and Fei Wu. 2025. Debiased cognition representation learning for knowledge tracing. *ACM Transactions on Information Systems*.

Julia M. Markel, Steven G. Opferman, James A. Landay, and Chris Piech. 2023. Gpteach: Interactive TA training with gpt-based students. In *Proceedings of the Tenth ACM Conference on Learning @ Scale, Copenhagen, Denmark, July 20-22, 2023*, pages 226–236. ACM.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Ruchir Puri, David S. Kung, Geert Janssen, Wei Zhang, Giacomo Domeniconi, Vladimir Zolotov, Julian Dolby, Jie Chen, Mihir R. Choudhury, Lindsey Decker, Veronika Thost, Luca Buratti, Saurabh Pujar, Shyam Ramji, Ulrich Finkler, Susan Malaika, and Frederick Reiss. 2021. Codenet: A large-scale AI for code dataset for learning a diversity of coding tasks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Alexis Ross and Jacob Andreas. 2024. Toward in-context teaching: Adapting examples to students' misconceptions. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 13283–13310. Association for Computational Linguistics.

Shashank Sonkar, Xinghe Chen, Naiming Liu, Richard G. Baraniuk, and Mrinmaya Sachan. 2024. Llm-based cognitive models of students with misconceptions. *CoRR*, abs/2410.12294.

William Stout. 2007. Skills diagnosis using irt-based continuous latent trait models. *Journal of Educational Measurement*, 44(4):313–324.

Yu Su, Qingwen Liu, Qi Liu, Zhenya Huang, Yu Yin, Enhong Chen, Chris H. Q. Ding, Si Wei, and Guoping Hu. 2018. Exercise-enhanced sequential modeling for student performance prediction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 2435–2443. AAAI Press.

Xia Sun, Xu Zhao, Bo Li, Yuan Ma, Richard F. E. Sutcliffe, and Jun Feng. 2022. Dynamic key-value memory networks with rich features for knowledge tracing. *IEEE Trans. Cybern.*, 52(8):8239–8245.

Xiaoyu Tan, Shaojie Shi, Xihe Qiu, Chao Qu, Zhenting Qi, Yinghui Xu, and Yuan Qi. 2023. Self-criticism: Aligning large language models with their understanding of helpfulness, honesty, and harmlessness. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: EMNLP 2023 - Industry Track, Singapore, December 6-10, 2023*, pages 650–662. Association for Computational Linguistics.

Dongsheng Wang, Jiequan Cui, Miaoge Li, Wang Lin, Bo Chen, and Hanwang Zhang. 2024a. Instruction tuning-free visual token complement for multimodal llms. In *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part LXXXI*, volume 15139 of *Lecture Notes in Computer Science*, pages 446–462. Springer.

Rose E. Wang, Qingyang Zhang, Carly Robinson, Susanna Loeb, and Dorottya Demszky. 2024b. Bridging the novice-expert gap via models of decision-making: A case study on remediating math mistakes. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 2174–2199. Association for Computational Linguistics.

Shulei Wang, Wang Lin, Hai Huang, Hanting Wang, Sihang Cai, WenKang Han, Tao Jin, Jingyuan Chen, Jiacheng Sun, Jieming Zhu, and Zhou Zhao. 2025. Towards transformer-based aligned generation with self-coherence guidance. *CoRR*, abs/2503.17675.

Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2021a. Denoising implicit feedback for recommendation. In *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021*, pages 373–381. ACM.

Wenjie Wang, Fuli Feng, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2021b. Clicks can be cheating: Counterfactual recommendation for mitigating clickbait issue. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 1288–1297. ACM.

Wenjie Wang, Xinyu Lin, Fuli Feng, Xiangnan He, Min Lin, and Tat-Seng Chua. 2022. Causal representation learning for out-of-distribution recommendation. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pages 3562–3571. ACM.

Ye Wang, Wang Lin, Shengyu Zhang, Tao Jin, Linjun Li, Xize Cheng, and Zhou Zhao. 2023a. Weakly-

supervised spoken video grounding via semantic interaction learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 10914–10932. Association for Computational Linguistics.

Zhifeng Wang, Wenxing Yan, Chunyan Zeng, Yuan Tian, and Shi Dong. 2023b. A unified interpretable intelligent learning diagnosis framework for learning performance prediction in intelligent tutoring systems. *International Journal of Intelligent Systems*, 2023(1):4468025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Tao Wu, Mengze Li, Jingyuan Chen, Wei Ji, Wang Lin, Jinyang Gao, Kun Kuang, Zhou Zhao, and Fei Wu. 2024. Semantic alignment for multimodal large language models. In *Proceedings of the 32nd ACM International Conference on Multimedia, MM 2024, Melbourne, VIC, Australia, 28 October 2024 - 1 November 2024*, pages 3489–3498. ACM.

Hanyi Xu, Wensheng Gan, Zhenlian Qi, Jiayang Wu, and Philip S. Yu. 2024a. Large language models for education: A survey. *CoRR*, abs/2405.13001.

Songlin Xu, Xinyu Zhang, and Lianhui Qin. 2024b. Eduagent: Generative student agents in learning. *CoRR*, abs/2404.07963.

Weicai Yan, Wang Lin, Zirun Guo, Ye Wang, Fangming Feng, Xiaoda Yang, Zehan Wang, and Tao Jin. 2025. Diff-prompt: Diffusion-driven prompt generator with mask supervision. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Rui Yang, Boming Yang, Sixun Ouyang, Tianwei She, Aosong Feng, Yuang Jiang, Freddy Lécué, Jinghui Lu, and Irene Li. 2024. Graphusion: Leveraging large language models for scientific knowledge graph fusion and construction in NLP education. *CoRR*, abs/2407.10794.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Murong Yue, Wijdane Mifdal, Yixuan Zhang, Jennifer Suh, and Ziyu Yao. 2024. Mathvc: An llm-simulated multi-character virtual classroom for mathematics education. *CoRR*, abs/2404.06711.

Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. 2024. Accessing GPT-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. *CoRR*, abs/2406.07394.

Yiyun Zhou, WenKang Han, and Jingyuan Chen. 2025a. Revisiting applicable and comprehensive knowledge tracing in large-scale data. *CoRR*, abs/2501.14256.

Yiyun Zhou, Zheqi Lv, Shengyu Zhang, and Jingyuan Chen. 2024. Cuff-KT: Tackling learners' real-time learning pattern adjustment via tuning-free knowledge state-guided model updating.

Yiyun Zhou, Zheqi Lv, Shengyu Zhang, and Jingyuan Chen. 2025b. Disentangled knowledge tracing for alleviating cognitive bias. In *Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025- 2 May 2025*, pages 2633–2645. ACM.

In this appendix, we present the following content:

Sample Illustration

Task : This problem requires input of a radius (greater than 0) and uses the constant pi from the math library to calculate the area of a circle. Note: If the input radius does not meet the conditions, repeat input until the conditions are satisfied before proceeding with the calculation.

Behavior : 1. The radius value in the output is hardcoded, always displaying as 5, regardless of the actual input. 2. The code lacks validation to ensure the radius is greater than 0 before proceeding with calculations.

Solution :

```
from math import pi
r=float(input(""))
area=pi*r**2
print("Radius 5, Area {:.5f}".format(area))
```

Figure 7: A sample illustration in our dataset.

A Dataset Statistics and Details

A.1 Privacy Protection

To ensure privacy protection, we excluded all personal and sensitive data, such as student names and email addresses, retaining only anonymized unique student IDs as individual identifiers. Furthermore, we confirmed that the use of user-generated data was explicitly authorized during the registration process, as outlined in the platform's terms of service and privacy policy.

A.2 Dataset Statistics

In Section 3, we construct a student simulation dataset, `Student_100`, comprising 100 students, each with 40 past learning records and 10 simulation records. Each record includes a Python programming task, the student's solving behavior (indicating correct or incorrect completion along with error analysis), and the corresponding solution (*i.e.*, student-written code). A sample is illustrated in Figure 7.

We further analyze the distribution of these students' cognitive level. Specifically, we use LLMs to score each student's solutions in the simulation records on a scale of 1 to 5, where higher scores indicate greater correctness. The scoring is evaluated using the o1-mini model, and the API version is o1-mini-2024-09-12. After scoring all solutions for a student, we calculate their average score to represent their cognitive ability. The distribution of these cognitive scores across the dataset is visualized in Figure 8.

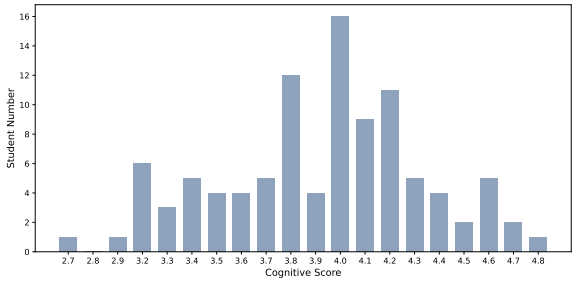

Figure 8: The distribution of student cognitive scores in our dataset.

Naive Prompt

You are to act as a {score}-point student (scores range from 1 to 5, with higher scores indicating better student ability). Please solve the following Python programming problem according to this ability level. Output your code directly.

Problem:
{problem}

Figure 9: Details of the naive prompt.

A.3 Significance of Programming Task for Student Simulation

In this section we would like to emphasize that programming tasks are sufficiently representative for the student simulation task for several reasons:

- Complexity and multi-leveled nature. Programming tasks inherently involve multiple levels of knowledge concepts, such as syntax, logical reasoning, and algorithm design. These tasks test not only a student's understanding of individual concepts but also their ability to integrate and apply these concepts to solve complex problems. This makes programming tasks an ideal scenario for modeling and simulating student cognitive states.
- Real-world applicability. Programming is a widely taught subject with high relevance in modern education and professional settings. Simulating student performance on programming tasks can provide practical insights into teaching strategies and learning patterns, making this work impactful in real-world educational contexts.
- Error diversity. Students often encounter diverse and challenging errors during program-

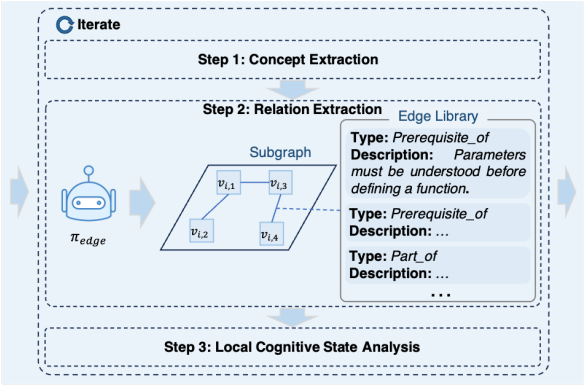

Figure 10: A detailed illustration of the edge library used in the cognitive prototype construction process. During the iterative process, each identified relationship is recorded in the corresponding edge's library. After processing all learning records, the relationship type that appears most frequently in each edge's library is selected as the final type for that edge.

ming, including syntax errors, logic errors, and runtime errors. These errors vary significantly in complexity, providing a rich testing ground for evaluating fine-grained behavior prediction and solution simulation.

- Interactive and iterative nature. Programming tasks typically require students to iteratively refine their solutions, reflecting a natural and realistic problem-solving process. This aligns closely with the goals of student simulation, where iterative behavior prediction and solution refinement are central.

We believe these characteristics make programming tasks a strong representative scenario for the student simulation problem.

A.4 Data Generalizability

The data used in this study was collected from an online programming platform similar to LeetCode. Despite the platform-specific origin of the data, it has been preprocessed to ensure it is not inherently tied to any particular platform. The preprocessing includes standardizing the format of problem statements, user solutions, and associated metadata, which makes the data format highly generalizable. As a result, similar data from other platforms (*e.g.*, LeetCode) can be processed into the same format, provided access to user submission records is available. This demonstrates the broader applicability of the proposed method beyond the platform used in this study.

Level Prompt

You will be shown a Python programming problem, and predict that whether a student of level {level} would correctly solve the problem. Please remind that a student's level ranges from 1 to 5, while 1 indicates the lowest ability and 5 indicates the highest ability. Directly output your prediction. If you think the student will make mistakes, provide possible details about the mistakes.

-Output Format-
Error Prediction: (Yes/No)

Error Description: (Your detailed analysis)

-Real Data-
Problem:
{problem}

Figure 11: Details of the Level prompt.

B Details on Figure 2

In Section 1, we use Figure 2 to illustrate that LLMs with naive prompts consistently generate correct answers, failing to simulate students with varying abilities, particularly those with weaker performance. In contrast, our simulation method more accurately reflects the distribution of student abilities.

The experiment proceeds as follows: we use the 15 students in the random selected `Student_15` subset (detailed in Section 5.1) to conduct this experiment. As explained in Appendix A.2, each student in the dataset is assigned a cognitive score evaluated by the LLM. This score serves as the basis for constructing naive prompts, guiding the model to respond in alignment with the corresponding ability level. The specific prompt is shown in Figure 9.

Using this naive prompt, the model generates simulated solutions. We then apply the scoring process described in Appendix A.2 to evaluate both the solutions generated by naive prompts and those produced by our method. The average score for each simulated student is calculated as their cognitive score. The results, presented in Figure 2, reveal that naive prompts consistently yield overly correct answers, failing to replicate weaker abilities. By contrast, our method closely matches the actual distribution of student abilities, providing a more realistic simulation.

C Method Details

C.1 Details on Concept Relationships

In Step 2 of Section 4.2, we describe the construction of edges in the knowledge graph and define

Input-Output (IO) Prompt

Given a Python programming problem, you need to write code to solve it. Your code must meet the following requirements:
1. Based on the provided error descriptions, your code must include and perfectly reproduce these errors.
2. Given a student's previous code, you can use the style of their historical code as a reference.

Directly output your code. At the end of your code, please include <e> to indicate its completion.

-Real Data-
Student's Past Code:
{eg_code}

New Programming Problem:
{question}

Error Descriptions:
{error_desc}

Code:

Figure 12: Details of the Input-Output (IO) prompt.

four types of relationships between the nodes (*i.e.*, concepts). The following provides an overview of these relationships:

- `Prerequisite_of`: This relationship indicates that one concept is either a defining characteristic of another or a necessary prerequisite of it. For example, "The ability to code is a prerequisite of software development."
- `Used_for`: This relationship signifies that one concept functions as a tool, resource, or method to achieve another. For instance, "Mathematics is used for solving engineering problems."
- `Hyponym_of`: This hierarchical relationship shows that one concept is a specific instance or subtype of another. For example, "A rectangle is a hyponym of a polygon."
- `Part_of`: This compositional relationship denotes that one concept is a component or integral part of a larger whole. For example, "A wheel is part of a car."

C.2 Edge Category Conflict Resolving

In Step 2 of Section 4.2, we construct edges in the knowledge graph by defining four types of relationships between nodes (*i.e.*, concepts). Since edges are extracted iteratively, overlapping edges may arise, potentially leading to conflicts where

Chain-of-Thought (CoT) Prompt

Given a Python programming problem, you need to write code to solve it. Your code must meet the following requirements:
1. Based on the provided error descriptions, your code must include and perfectly reproduce these errors.
2. Given a student's previous code, you can use the style of their historical code as a reference.

Directly output your code. At the end of your code, please include <e> to indicate its completion.

-Real Data-
Student's Past Code:
{eg_code}

New Programming Problem:
{question}

Error Descriptions:
{error_desc}

Make a plan then write. Your output should be of the following format:

Plan:
Your plan here.

Code:

Figure 13: Details of the Chain-of-Thought (CoT) prompt.

different relationship types are assigned to the same concept pair. To address this, we assign an edge library to each edge, as shown in Figure 10. Each identified relationship is added to the corresponding edge's library. After processing all learning records, we resolve conflicts by selecting the most frequently assigned relationship type in the edge library as the final type for each edge.

C.3 Scalability and Computational Feasibility

The scalability and computational feasibility of constructing the graph in our framework are key considerations. To address potential concerns about large-scale graphs, it is important to clarify that our approach is computationally manageable for the following reasons:

Upper limit on local knowledge state analyses. The construction of the global cognitive prototype (Step 4) evaluates a student's overall mastery of each knowledge concept by aggregating local cognitive state analyses from past learning records. The number of local analyses for each concept is directly bounded by the number of past records. In the experiments, the maximum number of local analyses is 17, which remains within a reasonable computational range. Additionally, the number of

Prompt for Con_1 metric

You are given a Python programming question, a student's answer code, and an analysis of errors found in the code. Then, I will provide a new error analysis. Your task is to evaluate the new analysis and determine how accurate and reasonable it is and how closely it matches the original error analysis. Please assign a score from 1 to 5, where:

• 1: The new analysis is completely unreasonable and unrelated to the actual error.
• 2: The new analysis attempts to describe the error but fails significantly or introduces unrelated issues.
• 3: The new analysis somewhat aligns with the actual error, but there are noticeable inaccuracies or omissions.
• 4: The new analysis is mostly accurate and aligns well with the actual error, with only minor differences or slight deviations.
• 5: The new analysis is fully accurate, natural, and matches the actual error explanation closely.

Additionally, please provide a brief explanation within 3 sentences justifying your score, focusing on the accuracy, relevance, and clarity of the new analysis in comparison to the original.

Format your output as (Score | Explanation), such as (5 | The new analysis is fully accurate). Do not output any other words.

-Real Data-
Question:
{question}

Student's answer code:
{gt_code}

Error Analysis:
{error_desc}

New Error Analysis:
{new_error_desc}

Output:

Figure 14: Details of the prompt for Con_1 metric.

past learning records is constrained to ensure stability in the student's cognitive state, an assumption critical to the student simulation task. Specifically, only records from a one-week period are considered, during which a student typically solves no more than 50 problems. This limitation ensures that the number of local analyses remains computationally manageable.

Upper limit on knowledge concepts. The relationship extraction process (Step 2) focuses on analyzing the relationships between knowledge concepts within each task to form the edges of the knowledge graph. To avoid redundancy and ensure manageability, the number of extracted knowledge concepts per task is limited to 15. This restriction makes the relationship extraction process computationally efficient, well within the capabilities of large language models.

C.4 Error Attribution and Traceability

To address the challenge of error attribution and tracing within the pipeline, our system logs each model invocation and saves the outputs of all intermediate stages. This design ensures full traceability, allowing for systematic failure analysis. When an error occurs in the final solution simulation, it is possible to trace outputs from previous stages, such as behavior prediction or retrieval, to pinpoint the root cause. This traceability supports targeted debugging and error analysis, isolating the impact of each stage on the system's overall performance.

D Experiment Details

D.1 Model Details

LLaMA-3.3-70B, released by Meta AI in December 2024, is a 70-billion-parameter multilingual large language model designed to advance AI applications in both research and industry. The Llama 3.3 instruction tuned text only model is optimized for multilingual dialogue use cases and outperforms many of the available open source and closed chat models on common industry benchmarks. The test model version is LLaMA-3.3-70B-Instruct.
Claude is a large language model developed by Anthropic, designed to generate human-like text and assist with a wide range of tasks, including answering questions, drafting content, and engaging in conversational interactions. Built with a focus on safety and alignment, Claude leverages advanced

Prompt for Con_2 metric

You are given a Python programming question, a student's answer code, and an analysis of errors found in the code. Then, I will provide a new code. Your task is to evaluate the new code and determine how well it reproduces the described errors. Please assign a score from 1 to 5, where:

- 1: The new code does not reproduce the described errors at all.
- 2: The new code attempts to reproduce some the described errors but fails significantly.
- 3: The new code somewhat reproduces some described errors but with significant differences.
- 4: The new code reproduces most the described errors well, with a little differences.
- 5: The new code fully and naturally reproduces the described errors.

Additionally, please provide a brief explanation within 3 sentences justifying your score, focusing on how closely the new code matches the nature of the described error.

Format your output as (Score | Explanation), such as (5 | The new code successfully reproduces the described errors). Do not output any other words.

-Real Data-
Question:
{question}

Student's answer code:
{gt_code}

Error Analysis:
{error_desc}

New Code:
{code}

Output:

Figure 15: Details of the prompt for Con_2 metric.

AI techniques to ensure responsible and effective communication, making it suitable for both casual and professional use. The tested API version is claude-3-5-sonnet-20241022.

GPT-3.5 is an advanced language model developed by OpenAI, designed to generate coherent and contextually relevant text across various domains. It builds on the capabilities of its predecessor, GPT-3, with improved understanding and responsiveness, making it highly effective for tasks like content creation, coding assistance, and conversational AI. Known for its versatility, GPT-3.5 is widely used in applications requiring natural language understanding and generation. The tested API version is gpt-3.5-turbo.

GPT-4o is OpenAI's generation language model, offering enhanced reasoning, creativity, and contextual understanding compared to its predecessors. It excels at complex tasks, such as advanced problem-solving, nuanced content creation, and in-depth conversational interactions. With improved alignment and multimodal capabilities, GPT-4o is designed to be more reliable, accurate, and adaptable across a wide range of applications. The tested API version is gpt-4o-2024-11-20.

D.2 Baseline Details

For behavior prediction, we compare our prototype mapping approach with five baselines:

- *Random*, which randomly selects a record from past learning records as a reference;
- *Similarity*, which selects a record based on task statement similarities;
- *Level* (Benedetto et al., 2024), which estimates a student's ability level based on the accuracy of their past learning records. Specifically, we first compute the accuracy of a student's past learning records and then normalize it to a range of 1 to 5, where 5 represents the highest cognitive level. As mentioned in Benedetto et al. (2024), this relative cognitive level simulation approach yields better results. We incorporate this level into the prompt to indicate the student's proficiency, enabling the model to perform behavior prediction accordingly. The prompt is shown in Figure 11.

Behavior Prediction		Random			Similarity			Level			Level+Random			Level+Similarity			Prototype Mapping		
Solution Simulation		IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine
LLaMA-3.3-70B-Instruct	**Acc**	0.29	0.29	0.29	0.35	0.35	0.35	0.34	0.34	0.34	0.32	0.32	0.32	0.36	0.36	0.36	**0.67**	**0.67**	**0.67**
	Con_1	2.26	2.26	2.26	2.44	2.44	2.44	2.28	2.28	2.28	2.24	2.24	2.24	2.43	2.43	2.43	**3.35**	**3.35**	**3.35**
	Con_2	2.01	2.02	2.03	2.22	2.23	2.16	2.65	2.38	1.79	2.21	2.02	2.02	2.47	2.31	2.14	2.75	2.73	**2.89**
Claude-3.5-Sonnet	**Acc**	0.49	0.49	0.49	0.55	0.55	0.55	0.27	0.27	0.27	0.35	0.35	0.35	0.36	0.36	0.36	**0.71**	**0.71**	**0.71**
	Con_1	2.87	2.87	2.87	3.07	3.07	3.07	1.81	1.81	1.81	2.25	2.25	2.25	2.31	2.31	2.31	**3.5**	**3.5**	**3.5**
	Con_2	2.93	2.77	2.77	3.1	3.01	3.01	1.62	1.72	1.65	2.41	2.28	2.27	2.6	2.56	2.46	3.22	3.32	**3.38**
GPT-3.5	**Acc**	0.37	0.37	0.37	0.36	0.36	0.36	0.5	0.5	0.5	0.42	0.42	0.42	0.45	0.45	0.45	**0.51**	**0.51**	**0.51**
	Con_1	2.47	2.47	2.47	2.47	2.47	2.47	2.87	2.87	2.87	2.66	2.66	2.66	2.73	2.73	2.73	**2.96**	**2.96**	**2.96**
	Con_2	3.47	3.56	3.61	3.53	3.51	3.57	3.54	3.58	3.59	3.47	3.53	3.59	3.34	3.58	3.58	3.46	3.58	**3.71**
GPT-4o	**Acc**	0.37	0.37	0.37	0.42	0.42	0.42	0.4	0.4	0.4	0.55	0.55	0.55	0.55	0.55	0.55	**0.89**	**0.89**	**0.89**
	Con_1	2.54	2.54	2.54	2.71	2.71	2.71	2.45	2.45	2.45	3.01	3.01	3.01	3.01	3.01	3.01	**3.9**	**3.9**	**3.9**
	Con_2	2.73	2.89	2.57	2.85	3.08	2.88	2.42	2.85	1.83	2.12	2.65	2.17	2.48	2.73	2.27	3.61	3.65	**3.83**

Table 5: End-to-end comparison across 6 behavior prediction and 3 solution simulation methods on the whole 100 students in `Student_100`.

- *Level+Random*, which incorporates *Random* and *Level*. We add the randomly selected past learning record into the *Level* prompt for behavior prediction;
- *Level+Similarity*, which incorporates *Random* and *Similarity*. We add the similarity-based retrieved past learning record into the *Level* prompt for behavior prediction.

For solution simulation, we compare our beam search-based self-refinement method with two baselines:

- *IO* (Yao et al., 2023), which combines the input with task instructions and few-shot input-output examples and requires models to directly output the simulated solution;
- *CoT* (Wei et al., 2022), which introduces a chain of thoughts that connects the input to the output, with each thought forming a coherent language sequence that acts as a meaningful intermediate step toward solving the problem.

Following Yao et al. (2023), we demonstrate the *IO* and *CoT* prompt in Figure 12 and 13.

D.3 Metric Details

Accuracy. We calculate the average accuracy of predicting whether each student correctly answers the corresponding 10 simulation records to reflect the behavior prediction quality.

Con_1. To assess the alignment and consistency between predicted and ground truth behavior descriptions, we introduce Con_1, a metric that utilizes a large language model (LLM) for evaluation. Con_1 assigns scores from 1 to 5, with higher values indicating stronger alignment with the ground truth. For fair and reliable comparisons, all evaluations use o1-mini (API version: o1-mini-2024-09-12).

Con_2. Similar to Con_1, Con_2 employs o1-mini to assign a score to each simulated solution, with higher scores indicating better consistency with the ground truth solutions.

Detailed prompts for Con_1 and Con_2 metric are provided in Figure 14 and 15.

E More Experimental Results and Analysis

E.1 End-to-end Comparison on `Student_100`

As shown in Section 5.2, we evaluate 6 behavior prediction methods and 3 solution simulation approaches, resulting in 18 unique configurations for comparison. We conduct end-to-end experiments across all configurations using the `Student_100` dataset, which includes the whole 100 students. The results, presented in Table 5, align with the findings in Section 5.2:

1) Evaluating a student's cognitive ability based solely on past learning accuracy is insufficient, as it does not capture mastery at the knowledge concept level. Even when selecting past records randomly or using text similarity retrieval, prediction accuracy remains low. This is because such methods often retrieve questions with similar wording but different underlying concepts, leading to misjudgments. In contrast, our cognitive prototype, constructed from a knowledge graph, accurately represents a student's mastery of relevant concepts, enabling more precise predictions and improving solution simulation quality.

2) Given the same prototype-mapped behavior descriptions, Table 5 shows that simulations based on simple IO or CoT prompts consistently underperform. This underscores the difficulty LLMs

Behavior Prediction		Random			Similarity			Level			Level+Random			Level+Similarity			Prototype Mapping		
Solution Simulation		IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine
LLaMA-3.3-70B-Instruct	ROUGE-L	12.63	12.23	18.92	12.74	12.49	21.32	12.55	12.39	20.31	12.49	13.11	19.28	13.95	11.88	20.94	15.76	13.59	**21.37**
	BLEU-4	0.59	0.63	2.18	0.69	0.78	2.77	0.49	0.52	1.41	0.69	0.64	2.31	0.79	0.82	2.86	0.93	0.86	**2.91**
Claude-3.5-Sonnet	ROUGE-L	21.24	20.96	23.67	21.8	25.72	24.61	18.58	20.99	22.67	20.24	20.06	21.51	23.59	26.55	26.37	24.5	24.32	**26.63**
	BLEU-4	2.47	2.39	3.76	3.75	2.96	4.53	1.37	1.88	3.07	2.41	2.49	3.56	3.27	4.11	4.77	4.25	3.17	**4.91**
GPT-3.5	ROUGE-L	23.59	19.94	25.76	25.59	19.08	26.34	20.44	19.23	24.13	22.68	19.51	25.41	22.85	21.38	26.61	23.88	20.91	**27.37**
	BLEU-4	3.28	1.8	4.37	4.78	2.51	5.6	2.3	1.75	4.21	3.0	2.56	4.53	4.52	2.6	5.38	4.19	2.65	**5.63**
GPT-4o	ROUGE-L	18.12	16.28	18.32	20.64	20.06	22.12	17.61	16.79	20.45	19.29	16.19	20.01	19.83	17.34	21.33	21.64	20.35	**22.28**
	BLEU-4	1.77	1.8	2.44	2.48	2.37	3.57	1.29	1.12	1.8	2.35	1.54	2.46	2.8	2.05	3.26	3.15	2.55	**4.59**

Table 6: End-to-end comparison results on the `Student_15` dataset for captioning metrics.

Behavior Prediction		Random			Similarity			Level			Level+Random			Level+Similarity			Prototype Mapping		
Solution Simulation		IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine
LLaMA-3.3-70B-Instruct	ROUGE-L	13.36	13.57	21.52	15.2	13.92	23.15	14.56	13.77	21.45	13.83	13.94	22.75	16.17	14.48	23.37	17.49	15.33	**24.09**
	BLEU-4	0.6	0.62	2.08	0.82	0.81	2.93	0.65	0.58	1.45	0.66	0.64	2.33	0.92	0.87	2.99	0.98	0.93	**3.06**
Claude-3.5-Sonnet	ROUGE-L	26.16	25.84	28.59	26.98	26.91	28.87	23.38	22.93	25.76	26.59	25.57	27.68	27.61	27.34	29.17	27.45	27.52	**29.95**
	BLEU-4	3.29	2.36	4.98	4.38	3.31	**5.94**	2.35	1.97	3.31	3.66	2.83	4.55	4.81	3.61	4.94	4.58	3.18	5.81
GPT-3.5	ROUGE-L	27.45	22.68	29.15	28.21	23.49	30.05	24.39	22.13	28.95	26.47	22.63	28.94	27.14	24.28	29.86	27.07	23.44	**30.34**
	BLEU-4	3.89	2.15	5.06	5.13	2.38	5.73	2.91	2.12	4.99	4.04	2.43	5.13	4.8	2.81	**5.95**	4.66	2.56	5.71
GPT-4o	ROUGE-L	22.99	20.96	22.51	23.97	22.65	25.15	18.92	18.67	21.82	21.14	19.44	22.2	22.51	20.77	23.18	24.9	22.76	**26.6**
	BLEU-4	2.32	1.84	2.54	3.36	2.33	3.47	1.26	1.14	1.83	1.97	1.4	2.17	2.74	2.12	2.85	3.35	2.24	**4.66**

Table 7: End-to-end comparison results on the `Student_100` dataset for captioning metrics.

face in generating accurate solutions for students with diverse cognitive abilities in a single attempt. Our beam search-based self-refinement method addresses this challenge by iteratively improving solutions through self-evaluation and optimization. By increasing sampling frequency, it enhances the probability of generating accurate, contextually consistent solutions, leading to higher-quality simulations.

E.2 Results for Captioning Metrics

We further evaluated the results of our solution simulation using captioning metrics, specifically ROUGE-L (Lin, 2004) and BLEU-4 (Papineni et al., 2002). The results, shown in Tables 6 and 7, indicate that our prototype mapping and self-refinement methods consistently achieve the best performance in most cases.

However, we believe that captioning metrics are not a suitable evaluation measure in the context of student simulation. The core focus of our task evaluation is to assess whether we can accurately and reasonably simulate the errors students are likely to make, rather than simply measuring text similarity. These metrics primarily evaluate low-level term similarity, which is limited in scope. Thus, they cannot effectively evaluate more complex aspects like the logical structure or syntax of code. For example, a student's incorrect code may differ from the correct version by just an indentation error, yet these metrics fail to capture such subtle distinctions. Therefore, we argue that captioning metrics do not provide a reasonable evaluation framework for our task and instead propose LLM-based metrics for assessment.

E.3 Results on Java and C++ programming

To further validate the effectiveness of our method beyond Python programming, we conduct additional experiments on two other programming subjects: Java and C++. We source new data from a new platform, CodeNet (Puri et al., 2021), which features tasks with a broad range of difficulty levels. Following the same expert annotation procedure of `Student_100` described in Section 3, we construct two new datasets, `Java_5` and `C++_5`, each containing 5 students with 40 past learning records and 10 test records per student.

We compare our method with baselines on these 2 datasets, and the results are shown in Table 8 and 9. Experimental results indicate that our method consistently outperforms all baselines on both datasets. This provides strong evidence for the robustness of our framework across different programming subjects, platforms, and student profiles.

E.4 In-Depth Analysis Details

In section 5.3, we analyze the impact of past learning volumes and the impact of self-refinement iterations and beam search sampling size. These experiments are conducted using the GPT-4o model and the results are shown in Figure 4. We provide the detailed performance in Table 11 and 12.

Behavior Prediction		Random			Similarity			Level			Level+Random			Level+Similarity			Prototype Mapping		
Solution Simulation		IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine
LLaMA-3.3-70B-Instruct	**Acc**	0.46	0.46	0.46	0.34	0.34	0.34	0.34	0.34	0.34	0.32	0.32	0.32	0.34	0.34	0.34	**0.68**	**0.68**	**0.68**
	Con_1	2.5	2.5	2.5	2.32	2.32	2.32	2.18	2.18	2.18	1.88	1.88	1.88	2.34	2.34	2.34	**3.3**	**3.3**	**3.3**
	Con_2	2.34	2.4	1.98	2.2	2.34	2.04	2.74	2.14	1.66	1.86	2.18	2.0	2.28	2.48	2.16	2.78	2.82	**2.84**
Claude-3.5-Sonnet	**Acc**	0.46	0.46	0.46	0.54	0.54	0.54	0.3	0.3	0.3	0.46	0.46	0.46	0.46	0.46	0.46	**0.56**	**0.56**	**0.56**
	Con_1	2.88	2.88	2.88	3.04	3.04	3.04	1.8	1.8	1.8	2.48	2.48	2.48	2.56	2.56	2.56	**3.3**	**3.3**	**3.3**
	Con_2	2.46	2.46	2.8	2.66	2.8	2.76	1.86	1.58	1.32	2.24	2.0	2.36	2.5	2.42	2.4	2.68	2.44	**2.9**
GPT-3.5	**Acc**	0.46	0.46	0.46	0.34	0.34	0.34	0.46	0.46	0.46	0.46	0.46	0.46	0.42	0.42	0.42	**0.52**	**0.52**	**0.52**
	Con_1	2.36	2.36	2.36	2.44	2.44	2.44	2.74	2.74	2.74	2.5	2.5	2.5	2.48	2.48	2.48	**2.92**	**2.92**	**2.92**
	Con_2	2.44	2.78	3.22	2.56	2.44	3.02	2.52	2.84	2.86	2.54	2.24	3.18	2.42	2.44	2.96	2.68	2.98	**3.28**
GPT-4o	**Acc**	0.3	0.3	0.3	0.5	0.5	0.5	0.42	0.42	0.42	0.66	0.66	0.66	0.56	0.56	0.56	**0.84**	**0.84**	**0.84**
	Con_1	2.48	2.48	2.48	2.9	2.9	2.9	2.34	2.34	2.34	3.56	3.56	3.56	3.28	3.28	3.28	**3.8**	**3.8**	**3.8**
	Con_2	2.28	2.5	2.6	2.74	2.78	2.74	2.28	2.8	1.9	2.44	2.64	2.4	2.24	2.52	2.1	3.24	3.38	**3.46**

Table 8: End-to-end comparison on `Java_5` dataset. *Acc* and *Con_1* metrics both evaluate behavior descriptions, yielding identical values for the same behavior prediction method.

Behavior Prediction		Random			Similarity			Level			Level+Random			Level+Similarity			Prototype Mapping		
Solution Simulation		IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine	IO	CoT	Refine
LLaMA-3.3-70B-Instruct	**Acc**	0.34	0.34	0.34	0.34	0.34	0.34	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	**0.62**	**0.62**	**0.62**
	Con_1	2.3	2.3	2.3	2.16	2.16	2.16	1.98	1.98	1.98	2.2	2.2	2.2	2.12	2.12	2.12	**3.0**	**3.0**	**3.0**
	Con_2	1.98	2.34	1.86	2.22	2.28	2.2	2.32	**2.52**	1.86	2.1	2.28	1.8	2.44	1.9	2.08	2.44	**2.52**	**2.52**
Claude-3.5-Sonnet	**Acc**	0.52	0.52	0.52	0.62	0.62	0.62	0.36	0.36	0.36	0.46	0.46	0.46	0.58	0.58	0.58	**0.66**	**0.66**	**0.66**
	Con_1	2.64	2.64	2.64	3.0	3.0	3.0	1.76	1.76	1.76	2.22	2.22	2.22	2.52	2.52	2.52	**3.16**	**3.16**	**3.16**
	Con_2	2.62	2.26	2.26	2.52	2.36	2.62	1.94	1.76	1.48	1.86	1.88	2.1	2.46	2.18	1.88	2.98	3.1	**3.16**
GPT-3.5	**Acc**	0.5	0.5	0.5	0.52	0.52	0.52	0.48	0.48	0.48	0.44	0.44	0.44	0.44	0.44	0.44	**0.58**	**0.58**	**0.58**
	Con_1	2.62	2.62	2.62	2.62	2.62	2.62	2.36	2.36	2.36	2.24	2.24	2.24	2.3	2.3	2.3	**2.7**	**2.7**	**2.7**
	Con_2	2.98	2.78	2.82	2.92	2.66	2.52	2.98	3.0	2.8	3.16	3.06	3.16	2.86	2.8	3.16	2.98	3.12	**3.18**
GPT-4o	**Acc**	0.5	0.5	0.5	0.54	0.54	0.54	0.42	0.42	0.42	0.54	0.54	0.54	0.68	0.68	0.68	**0.86**	**0.86**	**0.86**
	Con_1	2.5	2.5	2.5	2.68	2.68	2.68	2.08	2.08	2.08	2.84	2.84	2.84	3.2	3.2	3.2	**3.6**	**3.6**	**3.6**
	Con_2	2.64	3.04	3.1	3.14	3.3	2.9	2.42	3.08	2.04	2.32	2.7	2.24	2.6	2.8	2.8	3.32	3.4	**3.42**

Table 9: End-to-end comparison on `C++_5` dataset. *Acc* and *Con_1* metrics both evaluate behavior descriptions, yielding identical values for the same behavior prediction method.

E.5 Bad Case Analysis

For student behavior prediction and solution simulation, our method is able to accurately identify the relevant knowledge concepts that a problem tests by mapping the constructed student cognitive prototype to the task in most cases. However, this pipeline may occasionally fail, particularly when careless mistakes occur in the student's solution code. These rare but unavoidable mistakes, such as typos or incorrect code indentation (which can cause compilation errors in Python), may mislead the model's predictions. For example, if such mistakes appear in the simulation records of a student with otherwise high programming abilities, our method may incorrectly predict their behavior.

In fact, these occasional careless mistakes pose a challenge not only for our method but also for all baseline approaches, as they are equally susceptible to their influence (Lin et al., 2023; Wang et al., 2023a). We attribute this issue to the statistical nature of existing methods, which assess a student's cognitive level from a statistical perspective, making them ineffective at handling isolated errors.

We investigate whether increasing the number of retrieved past learning records could address this problem. The results, shown in Table 10, indicate that as the number of retrieved tasks increases, performance tends to decline. We speculate that retrieving more tasks introduces irrelevant information. Each retrieved task includes the problem statement, the student's solution, and expert error analysis, all of which are incorporated into the prompt for LLMs. As the number of retrieved tasks increases, the prompt becomes excessively long and redundant, overwhelming the model with too much information and leading to incorrect predictions.

Therefore, we recognize that predicting these occasional careless mistakes remains a challenging problem. Our method, however, is primarily designed to assess general cognitive proficiency—capturing a student's overall learning patterns rather than isolated, occasional errors. Fully addressing this issue would require additional methodological advancements, such as incorporating causal inference to mitigate biases and reduce

Number of Retrieved Tasks	LLaMA-3.3-70B			Claude-3.5-Sonnet			GPT-3.5			GPT-4o		
	Acc	Con_1	Con_2	Acc	Con_1	Con_2	Acc	Con_1	Con_2	Acc	Con_1	Con_2
1	0.61	2.99	2.69	0.65	3.09	2.99	0.56	2.99	3.49	0.94	3.77	3.65
2	0.6	3.02	2.25	0.63	3.1	2.61	0.42	2.53	3.15	0.73	3.24	3.04
3	0.61	2.97	2.4	0.62	3.05	2.72	0.47	2.63	3.19	0.7	3.24	3.12

Table 10: Experiments on different number of retrieved past learning records.

Past Learning Record Volumes	Acc	Con_1	Con_2
10	0.53	2.82	2.65
20	0.67	3.09	2.92
30	0.77	3.38	3.1
40	0.94	3.77	3.65

Table 11: Detailed performance on different number of retrieved past learning records.

B \ L	1	2	3	4	5
1	3.36	3.47	3.61	3.61	3.6
2	3.46	3.55	3.65	3.63	3.61
3	3.41	3.53	3.61	3.6	3.56

Table 12: Detailed performance on different refinement iteration L and beam search sampling size B.

the impact of confounding factors. We believe that this challenge extends beyond the scope of our current work and merits further exploration as an independent research direction.

Notably, similar challenges have been explored in other domains, such as recommendation systems (Wang et al., 2021a,b, 2022), where causal debiasing and counterfactual reasoning (Wang et al., 2025; Yan et al., 2025) have been employed to address issues like clickbait. We believe that effectively handling noise in educational data is an important avenue for future research, and we plan to explore this further in subsequent work.

E.6 Distinctions from Knowledge Tracing

At first glance, our task appears structurally similar to Knowledge Tracing (KT) (Zhou et al., 2025a, 2024), as both involve modeling student learning processes. However, direct comparison with KT methods is not entirely appropriate due to fundamental differences in objectives and generalization capabilities:

1) Differences in objectives. Student simulation requires not only predicting whether a student answers a question correctly but also diagnosing errors in detail and simulating realistic behaviors in the form of natural language. Our method generates explicit and interpretable descriptions of mistakes and solutions, whereas KT methods focus solely on correctness prediction and lack the ability to simulate detailed behaviors or provide natural-language explanations.

2) Generalization limitations. KT models rely on implicit parametric knowledge representations and problem indices without incorporating task-specific textual inputs (*e.g.*, problem statements). This reliance on training data restricts their ability to generalize to out-of-distribution (OOD) cases, such as the test data in our experiments. In contrast, our method leverages explicit cognitive prototypes, enabling robust performance in OOD scenarios and zero-shot tasks.

These distinctions highlight the fundamental differences between KT and student simulation, underscoring the need for tailored methodologies when modeling student behaviors in a more interpretable and generative manner.