STUDENTSIM: TRAINING LLM-BASED STUDENT SIMULATORS

Anonymous authors
Paper under double-blind review

ABSTRACT

AI tutors are most useful when they adapt to each student's strengths, weaknesses, and preferred guidance, but feedback about which guidance works for whom is sparse, slow, and costly to collect. Student simulators can provide this feedback at machine timescales. Existing approaches capture only part of what tutor training requires: state-tracking models fit learner behavior but cannot respond to natural-language guidance, while prompted LLMs respond fluently but do not reliably reproduce an individual student's competence and characteristic errors. We present STUDENTSIM, a training framework that turns sparse per-student records into individualized simulators through pooled domain training followed by per-student specialization. STUDENTSIM combines structured behavioral grounding with a language-model interface, enabling each simulator to mirror a student's responses and revise them under tutor guidance. We also introduce STUDENTSIMEVAL, a standardized protocol over 60 students in chess, second-language English writing, and mathematics. It evaluates *behavioral fidelity* ($\mathcal{F} \uparrow$), agreement between the simulator's responses and the real student's responses, and *guidance responsiveness* ($\mathcal{R} \uparrow$), whether the simulator reaches the response targeted by guidance. Across all three domains, STUDENTSIM outperforms behavior-prediction and prompted-LLM baselines on both metrics. In chess, STUDENTSIM reaches $\mathcal{F} = 0.51$ and $\mathcal{R} = 0.91$, compared with 0.23 and 0.72 for GPT-5.4 and 0.45 and 0.27 for Maia2. To demonstrate downstream use for AI tutor improvement, we use a trained STUDENTSIM as the reward in tutor reinforcement learning. Expert chess players rate the resulting tutor as more accurate, better-guided, and more personalized than a no-RL baseline and a tutor trained with GPT-5.4 simulator feedback.

1 INTRODUCTION

An AI tutor is most useful when it can adapt to differences in students' background knowledge, characteristic errors, and responses to instruction. Learning such adaptation from real interactions requires recruiting diverse students and repeatedly observing how each responds to different teaching approaches, which is slow and costly. This limits the pace at which adaptive tutors can be evaluated and improved.

Student simulators offer proxy feedback at machine timescales (Fig. 1). Educational measurement distinguishes what a learner produces independently from what the learner can achieve with support, the zone of proximal development (VYGOTSKY, 1980) that dynamic assessment operationalizes (Grigorenko & Sternberg, 1998). Motivated by this distinction, a simulator for tutor training must satisfy two requirements (Fig. 2). **Behavioral fidelity** measures how well it reproduces a student's own responses, including characteristic strengths and mistakes. **Guidance responsiveness** measures whether it updates those responses in the direction of tutor guidance. These properties provide the personalized starting point and capacity to be taught that tutor optimization requires. A pool of such simulators can stand in for a diverse population of teachable students.

Existing approaches usually satisfy only one requirement. State-tracking models, including knowledge tracing (Corbett & Anderson, 1995; Piech et al., 2015; Ghosh et al., 2020) and human-style behavior prediction (McIlroy-Young et al., 2020; 2022; Tang et al., 2024), learn from human behavior but lack an input pathway for tutor explanations. Prompted LLM simulators (Owoicho et al., 2023; Terragni et al., 2023) respond fluently to guidance, but textual descriptions do not reliably constrain

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

them to an individual's competence and characteristic errors. The challenge is therefore to build per-student simulators that are both behaviorally faithful and responsive to guidance.

We address this gap with STUDENTSIM, a framework for building individualized simulators from sparse records. Its two-stage pipeline (Fig. 3) first pools domain records to learn shared response patterns and how responses change under guidance, then specializes that base on each student's own records. This combination grounds the simulator in individual behavior while retaining a language interface for tutor guidance. STUDENTSIMEVAL places our models and prior families on the same footing: it defines fixed per-student train and held-out splits for 60 students across chess, second-language English writing, and mathematics, and evaluates every method on behavioral fidelity ($\mathcal{F}$) and guidance responsiveness ($\mathcal{R}$). We provide benchmark-construction and evaluation code in the anonymous supplementary material.

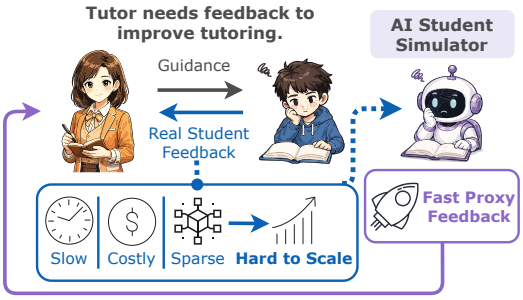
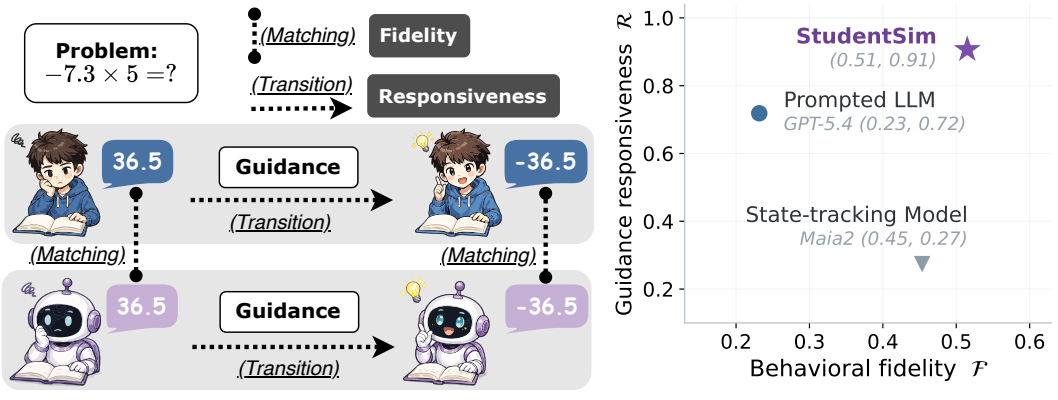


Figure 1: **Real-student feedback is hard to scale; a student simulator supplies it at machine timescales.**

Across all three domains, STUDENTSIM outperforms the evaluated baselines on both metrics, on average and for most individual students. In chess, STUDENTSIM reaches $\mathcal{F} = 0.51$ and $\mathcal{R} = 0.91$, compared with 0.23 and 0.72 for GPT-5.4 (OpenAI, 2026) and 0.45 and 0.27 for Maia2 (Tang et al., 2024). The comparison exposes complementary limitations: prompted GPT-5.4 responds to guidance but poorly reproduces individual behavior, while Maia2 models human moves but cannot consume natural-language guidance. STUDENTSIM is strong on both axes, and the same pattern holds in L2 and mathematics (Sections 5.2 and 5.3).

We also give a proof of concept that tests whether simulator feedback can improve an AI tutor. Using a trained STUDENTSIM as the reward in chess tutor reinforcement learning produces a tutor that expert human raters score as more accurate, better-guided, and more personalized than both the supervised-finetuned tutor without RL and a tutor trained with GPT-5.4 simulator feedback (Sec. 6). This result shows that faithful and responsive student simulators can provide useful proxy feedback for tutor optimization.



(a) What a simulator is scored on

(b) Where each method lands, on chess

Figure 2: **Scoring a student simulator.** Education tracks a learner's progress through what they produce independently and how far they advance once support arrives, the zone of proximal development (VYGOTSKY, 1980) that dynamic assessment operationalizes (Grigorenko & Sternberg, 1998). We score whether a student simulator can serve as a feedback source for a tutor, which rests on two abilities mirroring those signals: *behavioral fidelity*, behaving like the student it stands for, and *guidance responsiveness*, moving where the tutor's guidance indicates (a). On chess (b), STUDENTSIM leads on both, the prompted LLM on responsiveness alone, and the state-tracking model on fidelity alone, and the same ranking holds in L2 and math (Table 2 and Table 3).

Our contributions are: *i)* a formulation of personalized student simulation around the jointly required properties of behavioral fidelity and guidance responsiveness; *ii)* a two-stage pooled-then-specialized training framework that realizes both properties from sparse per-student data; *iii)* STUDENTSIMEVAL, a standardized three-domain protocol over 60 students, under which STUDENTSIM outperforms behavior-prediction and prompted-LLM baselines; and *iv)* a proof-of-concept RL study in which STUDENTSIM feedback yields the best-rated tutor in an expert human evaluation.

2 RELATED WORK

Prior work spans student modeling, LLM-based simulation, and tutor optimization. Cognitive-state and behavior-prediction models, including knowledge tracing and human-play move predictors, estimate learner behavior but do not accept natural-language guidance (Corbett & Anderson, 1995; Piech et al., 2015; Ghosh et al., 2020; McIlroy-Young et al., 2020; Tang et al., 2024). LLM-prompted simulators add a conversational interface (Owoicho et al., 2023; Terragni et al., 2023) and have supported tutoring dialogues, multi-agent classrooms, application-specific simulation, and learner-data generation (Macina et al., 2023; Zhang et al., 2024; Gao et al., 2025; Xu et al., 2025; 2024; Lu & Wang, 2024; Markel et al., 2023). Yet textual competence descriptions weakly constrain individual behavior, motivating validity studies from architectural, fidelity-benchmark, and teacher-facing perspectives (Yuan et al., 2026; Scarlatos et al., 2026; Wu et al., 2025; Martynova et al., 2025; Mannekote et al., 2024; Louie et al., 2024; Peng & Shang, 2024; Samuel et al., 2025; Wang et al., 2024d). Tutor-side methods learn from expert dialogues or generic LLM judges (Chevalier et al., 2024; Zheng et al., 2023), and recent work uses rubric feedback, knowledge tracing, or prompted students as reward signals (Scarlatos et al., 2025b; Dinucu-Jianu et al., 2025; Maurya et al., 2025); complementary studies develop pedagogical benchmarks and classroom or expert evaluations (Macina et al., 2025; Wang et al., 2024b;c; Team et al., 2024).

To our knowledge, STUDENTSIM is the first to turn individualized student simulation into an end-to-end, testable feedback system for improving AI tutors. It *i)* formalizes per-student simulation as the joint requirement of behavioral fidelity and guidance responsiveness; *ii)* trains simulators on real per-individual learner records so that they both reproduce a student's own behavior and respond to natural-language tutor guidance; and *iii)* uses the simulator learned within the same framework as the reward source for optimizing an AI tutor (Sec. 6). Sec. A provides a detailed per-strand survey.

3 PROBLEM FORMULATION

Let $\{\pi_1, \ldots, \pi_N\}$ be N real students and $\{M_1, \ldots, M_N\}$ be trained student simulators, where M_i is meant to approximate π_i. For each student i, we hold out two kinds of recorded data. *Single-turn records*, $S_i = \{(x, m)\}$, pair a problem x with the student's actual response m; S_i serves to measure whether the simulator's prediction matches the student's response. *Multi-turn records*, $T_i = \{(x, m, \tau, m^*)\}$, additionally include the tutor guidance τ provided to address the student's wrong response m on x and the canonical corrected response m^* that the guidance is steering toward (e.g., the engine-recommended move in chess, the corrected fragment in second-language English writing, or the correct answer in mathematics); T_i serves to measure whether the simulator successfully updates toward m^* after reading guidance. A useful population must answer two questions: does each simulator match its target student's behavior on S_i, and does it produce the corrected response on T_i after reading guidance?

Behavioral fidelity. Per-student behavioral fidelity, $\mathcal{F}_i$, scores simulator M_i on S_i: how well its response on a held-out problem matches student π_i's recorded response. Each of three domains tested in STUDENTSIMEVAL instantiates this match notion to its own response space. In chess, fidelity is move-prediction accuracy: how often the simulator's predicted move is the same move the student actually played. In second-language English writing, fidelity is error-profile match: whether the simulator's generated essay matches the student's overall error rate and issue-type profile. In mathematics, where problems are four-way multiple-choice, fidelity is how faithfully the simulator's prediction matches the option the student actually selected. The population score is $\mathcal{F} = \frac{1}{N}\sum_i \mathcal{F}_i$. Appendix C gives the per-benchmark formula and operational recipe; Appendix F.3 gives the design rationale for the per-domain operationalization.

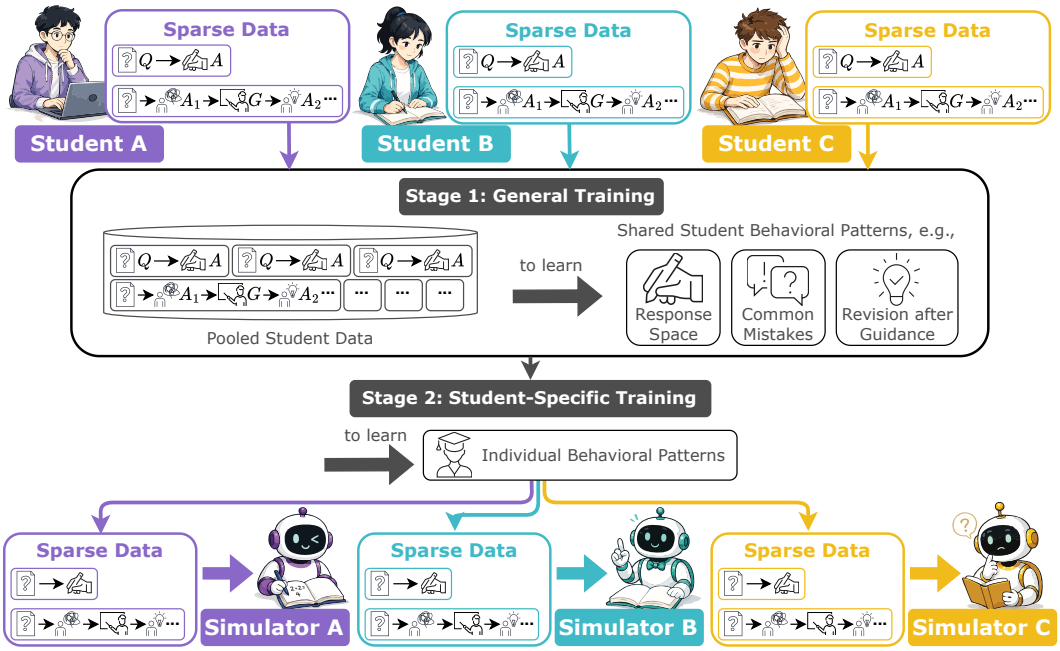


Figure 3: **Two-stage training pipeline for StudentSim.** Stage 1 pools the sparse records of many students in a domain and trains one base simulator on what they share. Stage 2 specializes that base to a single student on that student's own records, yielding one simulator per student.

Guidance responsiveness. Per-student guidance responsiveness, $\mathcal{R}_i$, scores simulator M_i on T_i using multi-turn records (x, m, τ, m^*), where x is the problem, m is the student's initial response, τ is the tutor guidance addressing it, and m^* is the response that guidance is steering toward. $\mathcal{R}_i$ asks: after reading (x, m, τ), does the simulator update to m^*? The population score is $\mathcal{R} = \frac{1}{N}\sum_i \mathcal{R}_i$. This isolates the simulator's update under guidance: behavioral fidelity already captures the student-specific answer patterns present before the intervention, and $\mathcal{R}$ captures whether the simulator takes up the guidance and follows it to the endpoint it steers toward. That is the operative property when the simulator serves as a reward model for tutor training (Section 6), since a static imitator that ignores guidance offers no signal about whether an intervention moved the student. Appendix F.1 details this target choice.

Orthogonality. $\mathcal{F}$ and $\mathcal{R}$ measure separable capabilities: a faithful but unresponsive simulator describes the student without responding to guidance, and a responsive but unfaithful one follows guidance from the wrong starting state. Therefore, the high-$\mathcal{F}$, high-$\mathcal{R}$ corner is the target.

4 StudentSim Training Pipeline

Per-student records are sparse. In the second-language English writing corpus (Öksüz et al., 2026), for example, a learner writes three essays at the median, and more than two thirds write five or fewer (Appendix B). Collecting real-user records is slow and costly, so sparse per-user data is common in interaction-based systems (Schein et al., 2002; Lee et al., 2019; Bhattacharjee & Wayllace, 2025; Li et al., 2017). Training a large language model end-to-end for each student on so little data would overfit and repeatedly relearn shared domain structure (Dodge et al., 2020; Finn et al., 2017). We therefore use two stages (Fig. 3).

Stage 1: pooled training. We train one domain-specific base simulator on records pooled across students. It learns common mistakes, the domain's response format, and how natural-language guidance changes a response, yielding broad student-like behavior before individual specialization.

Stage 2: per-student specialization. Starting from the Stage 1 base, we adapt an independent simulator M_i using only student i's responses and tutor-guidance interactions. This stage captures

individual mistake patterns while retaining the ability to advance toward the target of an intervention under varied guidance.

Why two stages. Stage 1 supplies shared structure that sparse individual records cannot reliably identify, while Stage 2 grounds that structure in one student's behavior. In chess, replacing pooled records with repeated records from one student lowers performance (Appendix D.1).

Student state grounding and interactive realization. STUDENTSIM separates structured grounding from interactive realization. A structured state summarizes prior behavior, including piece usage and playing style in chess, grammatical error patterns in second-language English writing, and skill-level performance in mathematics. This state grounds cognitive fidelity. A language model realizes it as an interactive object and provides the interface for reacting to natural-language tutor guidance. Training aligns expressed competence with the state evidence while teaching guided revision toward the intended response. Behavioral fidelity evaluates the grounding, and guidance responsiveness evaluates whether the simulator reaches the criterion response under guidance.

5 RESULTS

This section instantiates the framework on chess, second-language English writing (L2), and mathematics (math). Sec. 5.1 specifies data sources, tasks, guidance types, training setup, and baselines. Sec. 5.2 reports behavioral fidelity $\mathcal{F}$. Sec. 5.3 reports guidance responsiveness $\mathcal{R}$.

5.1 EXPERIMENT SETUP

Data sources, tasks, and student populations. Each domain provides single-turn records (problem $\rightarrow$ student response) used to learn a student's baseline behavior and multi-turn records (problem $\rightarrow$ wrong response $\rightarrow$ tutor guidance $\rightarrow$ canonical corrected response) used to learn whether the simulator reaches the criterion response under tutor guidance, instantiating the (S_i, T_i) schema from Sec. 3. All three corpora consist of records from real human students. For chess, the corpus is drawn from Lichess (May 2025), a fully open-source online chess platform on which players record real games against human opponents; each record pairs a board position with the player's actual move. The simulator task is to predict a specific player's next move and, after natural-language coach guidance, to produce the move targeted by the guidance and identified as strongest by a chess program. For second-language English writing (L2), we use the EFCAMDAT corpus (Geertzen et al., 2014), a collection of essays by English-as-foreign-language learners alongside tutor corrections; the simulator task is to generate essays whose error patterns match a specific learner and, after tutor feedback, to produce the teacher's corrected fragment. For mathematics (math), we use a foundational-assistance corpus (Worden et al., 2026) of open-ended problems annotated with student answers and correct answer; the simulator task is to predict a specific student's answer option and, after a tutor explanation, to produce the audited correct answer. In all three domains, Stage 1 pretrains on a larger pool of students and Stage 2 specializes per-student simulators on a student-specific subset; per-student evaluation records are disjoint from training data. STUDENTSIMEVAL evaluates a fixed roster of 30 chess players, 15 L2 learners, and 15 math students, each on a per-student held-out set that is frozen and scored identically for every method; these per-student held-out sizes are the last two columns of Table 1, and the full evaluation protocol is in Appendix B.1. Table 1 summarizes the per-domain Stage 1 and Stage 2 training data scale. Construction of the multi-turn corpora (real-teacher-annotated for L2, controllable LLM-tutor-generated for chess and math under fixed style templates) is detailed in Appendix F.2; for chess and math, the LLM determines only the wording of the guidance, while the response it steers toward is fixed in advance by the chess program or an audited answer key.

Guidance types. Effective tutoring adjusts the level of support to what the learner needs at that moment, giving more when the student is struggling and stepping back as the student takes over (Wood et al., 1976). Each domain therefore includes guidance types that span that range of support: chess uses error remediation (diagnosing the error and explaining the correction), comparative (contrasting the played move with stronger alternatives), strategic (outlining the plan the chess position calls for), and Socratic (asking questions that guide the student's reasoning); L2 uses point-based (scoping feedback to one error occurrence) and rule-based (explaining the underlying grammatical rule) correction; math uses error remediation, Socratic, and conceptual explanation (teaching the principle

Table 1: **Per-domain training data scale: Stage 1 (pooled training), Stage 2 (per-student specialization), and held-out evaluation per Stage-2 student.** Stage 1 / Stage 2 *instances* count training records (single-turn plus multi-turn pooled); the multi-turn ratio is the fraction of multi-turn records (T_i) sampled into training batches, with the remainder single-turn (S_i). The held-out columns count per-student evaluation records: S is the single-turn split, and T the multi-turn split. Each count is the same for every student in the domain, except math T (mean shown, range in parentheses): each student has too few wrong-answer problems for a uniform cap to leave enough per student, so we keep all of them, and the count varies with how many the student first answered incorrectly.

Domain	Stage 1 (pooled)		Stage 2 (per-student)		Multi-turn ratio	Held-out (per-student)	
	students	instances	students	instances		S	T
chess	100	100,000	30	1,000	0.20	5,000	4,000
L2	200	7,800	15	73	0.20	26	40
math	200	23,400	15	153	0.20	66	59 (21–99)

Table 2: **Behavioral fidelity ($\mathcal{F}$) by domain.** The naive baseline is Maia2 for chess and untrained Qwen3-4B-Instruct for L2 and math; untrained Qwen is omitted for chess because it cannot reliably produce legal moves. Appendix E.4 reports standard deviations across training seeds.

Domain	$\mathcal{F}$ Definition	Naive Baseline	GPT-4o	GPT-5.4	StudentSim
Chess (30 players)	top-1 accuracy (↑)	0.4535	0.2163	0.2316	**0.5150**
L2 (15 learners)	error-profile match (↑)	0.5130	0.4718	0.5141	**0.5624**
Math (15 students)	K=4 MC accuracy (↑)	0.4919	0.5121	0.6121	**0.6384**

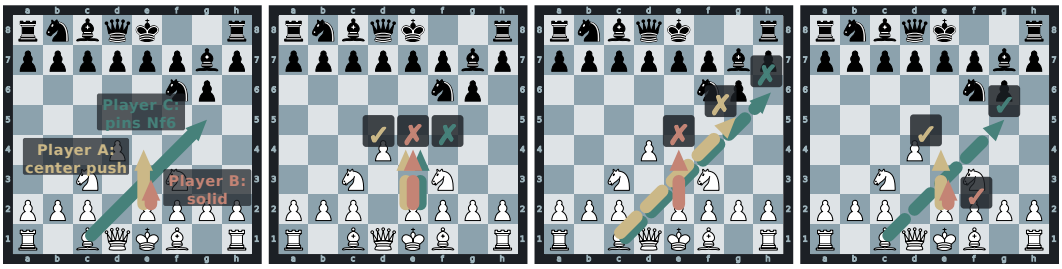
behind the error). Error remediation covers the direct-correction end of that range, the appropriate tutor action when a learner is stuck or repeatedly failing (Koedinger & Aleven, 2007). A tutor trained against the simulator has to learn when to use that level of support, so the training environment must include students who respond to it. At the other end, Socratic and conceptual modes correspond to the case where the tutor asks questions or supplies a principle without stating which response is correct, so the learner must work the correction out for themselves. The simulated student must then reason through those hints to arrive at the correction, making these modes the hardest case for guidance responsiveness. Multi-turn records mix these guidance types uniformly during training so the simulator learns to follow any of them at evaluation time; per-mode breakdown is in Appendix C.7.

Training setup. All domains use Qwen3-4B-Instruct (Yang et al., 2025). Stage 1 trains a domain LoRA adapter on pooled records with a 0.2 multi-turn ratio; Stage 2 continues from it on each student's records to produce one adapter per student. Both stages share the base model, LoRA architecture, and optimizer; only batch size and training length vary with dataset scale. Decoding is greedy ($T = 0$). Results are means over three independent runs; standard deviations, the two-stage ablation, and full hyperparameters are in Appendices E.4, D.1, and E.

Baselines. We compare three baseline families: *i)* GPT-4o (OpenAI, 2024) and GPT-5.4 (OpenAI, 2026), prompted with each student's profile, problem, and guidance; *ii)* Qwen3-4B-Instruct, prompted identically but without domain training, for L2 and math; and *iii)* Maia2 (Tang et al., 2024) for chess, a human-style move predictor conditioned on FEN and player ELO. We omit untrained Qwen for chess because it cannot reliably predict legal moves and its near-zero $\mathcal{F}$ would be uninformative. Chess has a fixed, narrow scope and dedicated behavior engines, but L2 and math have no analogous specialized model; the closed LLMs are therefore the strongest available references. All baselines use the same held-out records as our trained simulators. Appendix F.5 gives per-domain details.

5.2 Behavioral Fidelity Results

The trained simulators outperform every reference baseline on $\mathcal{F}$ across all three domains (Table 2). A higher $\mathcal{F}$ has a different concrete meaning in each benchmark: in chess it means the simulator predicts the player's actual next move more accurately; in L2 it means the simulator's essays match, more closely, the rate and category profile of the mistakes a particular learner still makes on a writing prompt; in math it means the simulator more often selects, among the answer choices, the specific



(a) Ground truth (b) Maia2 (c) GPT-5.4 (d) STUDENTSIM

Figure 4: **Behavioral fidelity on a held-out chess position.** Three players choose distinct moves (a). Maia2 collapses them to the Elo-modal `e4` (b), and GPT-5.4 misses all three (c); STUDENTSIM reproduces every player's move (d). Solid arrows show ground truth and dashed arrows show predictions. Across all 30 players, $\mathcal{F}$ is **0.51** for STUDENTSIM, 0.45 for Maia2, 0.23 for GPT-5.4, and 0.22 for GPT-4o.

answer (correct or not) that this student actually gave. We use chess as the case study below because per-student behavior distinction shows up most directly at a single board position.

Fig. 4 shows one held-out position where three real players in our test set choose three different moves. Each move reflects a different style: center push, solid, and tactical pin. The trained simulators reproduce all three actual moves, one per player. This is the capability per-student training is meant to add: at a fixed input, the simulator's output reflects the specific student it was trained on rather than a population-level prediction. Neither baseline matches this resolution. Maia2 conditions on player ELO only and collapses all three players onto its ELO-modal `e4`. GPT-5.4 prompted with each player's textual history also misses all three, because a textual description of past games does not constrain move choice at per-student resolution.

5.3 GUIDANCE RESPONSIVENESS RESULTS

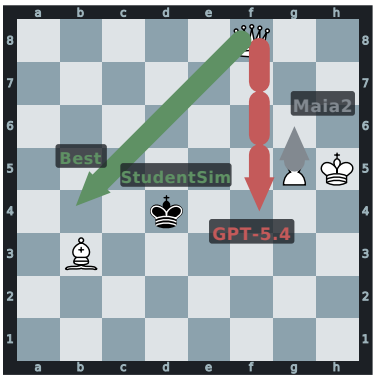
Table 3: **Guidance responsiveness ($\mathcal{R}$) by domain.** The naive baseline is Maia2 for chess and untrained Qwen3-4B-Instruct for L2 and math. Appendix E.4 reports standard deviations across training seeds.

Domain	$\mathcal{R}$ Definition	Naive Baseline	GPT-4o	GPT-5.4	STUDENTSIM
Chess (30 players)	corrected-move rate (↑)	0.2721	0.7655	0.7186	**0.9067**
L2 (15 learners)	fragment-rewrite match (↑)	0.0200	0.3883	0.5950	**0.6417**
Math (15 students)	answer-correction rate (↑)	0.6132	0.6940	0.7099	**0.9181**

Across all three domains, the trained simulators outperform every baseline on $\mathcal{R}$ (Table 3); the gap is largest on open-ended modes where the tutor prompts the student to reason through questions or principles. A higher $\mathcal{R}$ has a different meaning in each benchmark: in chess, after textual guidance, the simulator moves from its initial wrong move toward the response the guidance was intended to elicit; in L2, after teacher feedback flags the offending fragment, it supplies the exact correction; in math, after the tutor explains the reasoning behind the error, it revises the wrong numerical answer toward the correct one. The case study illustrates the strictest chess mode.

Fig. 5 examines Socratic guidance, the strictest chess mode because the tutor poses a sequence of questions rather than stating the corrected move. In this held-out position, the player chose the blunder `g5g6`. The questions steer the student's reasoning toward a forcing queenside check and ultimately to the engine target `f8b4`, although the guidance never names the destination square. To respond correctly, the simulator must take the student's role and derive the answer by following the Socratic reasoning path.

Maia2 cannot read the guidance and repeats the blunder. GPT-5.4 produces the queen move `f8f4` but does not carry the guided reasoning to its conclusion. Our per-player 4B simulator works through the same sequence of questions in the student's role and derives `f8b4`. That a 4B trained simulator completes this Socratic reasoning chain where a much larger prompted model does not is evidence



White to move. FEN
`5Q2/8/8/6PK/3k4/1B6/8/8 w`

Socratic guidance (abridged):

"Compare `1. g5g6` with a forcing queen move that gives check from the queenside. After such a check, where is Black's king forced to go, and do you gain a useful tempo? . . . Is it stronger to push the pawn immediately, or to cut off the enemy king with the queen first?"

The guidance never names the destination square.

Source	Move	Follows guidance?
Player's actual move	`g5g6`	n/a
Guidance target	`f8b4`	n/a
Maia2 (no guidance support)	`g5g6`	×
GPT-5.4 (after guidance)	`f8f4`	×
StudentSim (Ours) (after guidance)	`f8b4`	✓

Figure 5: **Guidance following on a held-out chess position.** The Socratic prompt guides the student through a sequence of questions whose reasoning leads from the player's `g5g6` blunder to `f8b4`, without naming the destination square. Maia2 repeats the blunder, GPT-5.4 does not follow the guided reasoning to its conclusion, and StudentSim derives `f8b4` from the questions. Across the test set, $\mathcal{R}$ is **0.91** for StudentSim, 0.77 for GPT-4o, 0.72 for GPT-5.4, and 0.27 for Maia2.

that multi-turn training adds guidance-following capability rather than merely copying explicit hints. Per-mode breakdowns are in Appendix C.7.

6 Tutor RL with Student Simulator Feedback

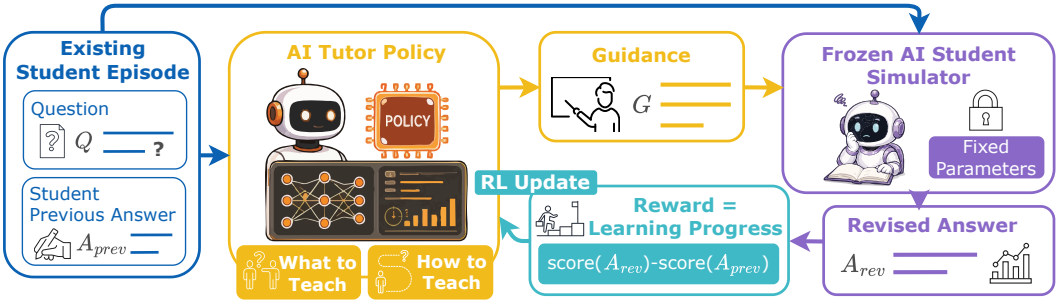


Figure 6: **Tutor RL with a frozen student simulator.** Improvement in the simulator's move quality supplies the reward for updating the tutor policy.

We give a chess proof of concept comparing StudentSim feedback with no RL and GPT-5.4 simulator feedback. The claim is that a reward system built around a faithful, responsive, and extensible simulator provides a practical signal for tutor optimization, not that we produce a best-in-class tutor. Chess isolates this question because Stockfish (The Stockfish developers, 2025) provides a precise reward independent of simulator behavior; free-form domains need domain-specific rewards beyond the present scope (Appendix F.6). We evaluate all conditions with an expert human study (Appendix E.5).

Setup. The loop is shown in Fig. 6. Each episode draws a chess problem Q and recorded wrong move A_{prev} from real student records; the tutor proposes guidance, a frozen simulator emits A_{rev}, and a precomputed Stockfish lookup rewards improvement over A_{prev} (full formulation in Appendix E). All conditions use the same Qwen3-VL-8B tutor (Bai et al., 2025), rendered-board input, SFT starting point, and GRPO procedure (Shao et al., 2024). We compare the reward systems each simulator interface can practically support:

- **No RL**: the SFT tutor itself, a lower-bound control.

- **GPT-5.4 simulator**: GPT-5.4 plays the student after reading the tutor's guidance, and the shared move-quality reward scores its revised move. This closed, fixed-behavior baseline requires a frontier-model API at every rollout.
- **STUDENTSIM (Ours)**: the reward uses our trained STUDENTSIM with two extensible heads mounted on the same backbone, all reading the tutor's explanation. The shared term scores the revised move's quality improvement; a personalization head scores the intended teaching style; and a perception head penalizes board-state misdescriptions, such as a wrong square or hallucinated piece. The heads act as multiplicative gates on move quality (Appendix E.5). The simulator and heads are small, open, locally served, and customizable.

For this proof of concept, we intentionally use pooled Stage-1 feedback to train a general tutor; per-student Stage-2 environments would tie optimization to personal records and raise privacy and consent concerns.

Expert Human Study. After tutor model optimization, competitive chess players rate each tutor's generated guidance, blind to presentation order, on three axes: accuracy (the fraction of a tutor's responses with no misleading factual error), guidance quality (a 1–5 rating of how well the response teaches), and personalization (a 1–5 rating in the context of a Socratic-leaning student). Among the rewards we study, our STUDENTSIM reward produces the best-rated tutor on all three axes (Table 4). The GPT-5.4-simulator reward stays below even the no-RL baseline on accuracy, held back by a markedly higher rate of severe factual errors. Full study details are in Appendix E.5.

Table 4: **Expert ratings of the AI chess tutors.** Means over 74 annotations from 8 evaluators.

Condition	Accuracy (%)	Guidance (1–5, ↑)	Personalization (1–5, ↑)
No RL	75.7	2.99	2.80
GPT-5.4 reward	71.6	3.08	2.42
STUDENTSIM (Ours)	**90.5**	**3.31**	**3.93**

Scope and interpretation. Holding the tutor policy, SFT initialization, and GRPO fixed, the human study compares two deployable rewards: integrated STUDENTSIM, whose faithful and responsive open backbone supports teaching-style and board-state-perception heads, and closed GPT-5.4, which cannot support them. The human-study results therefore demonstrate the effectiveness of the integrated STUDENTSIM-based reward system.

7 CONCLUSION

STUDENTSIM is a training framework for per-student AI simulation built around two requirements that must hold together: behavioral fidelity ($\mathcal{F}$), matching a student's own responses, and guidance responsiveness ($\mathcal{R}$), updating where tutor guidance leads. To measure both on a common footing, STUDENTSIMEVAL casts public learner corpora across chess, L2 writing, and mathematics into a standardized per-student protocol on which any simulator, from our framework or a prior family, is fit and scored on the same held-out records. Under this protocol, our pooled-then-specialized pipeline yields a reference family of 60 individualized simulators strong on both axes across all three domains, outperforming domain-specific state tracking (weak on $\mathcal{R}$) and prompt-only LLM role-play (weak on $\mathcal{F}$). As a proof of concept that simulation can improve tutors, a frozen STUDENTSIM used as the reward in a chess tutor RL loop yields a tutor that experts rate as more accurate, better-guided, and more personalized than tutors trained with a frontier-LLM-simulator reward or no RL.

Looking ahead, behavioral fidelity and guidance responsiveness capture a student's state and its one-step update under guidance. Extending simulation to the fuller dynamics of how students acquire, retain, and forget knowledge across many tutor interactions or through self-learning is the deeper foundation of a simulator's value for AI tutor optimization and the direction STUDENTSIM opens toward.

Ethics Statement

This work uses pre-existing records of human behavior from three sources with different access conditions. The Lichess game database is publicly released under CC0. EFCAMDAT requires an academic affiliation and a signed user agreement, and FoundationalASSIST requires acceptance of its Responsible Use Agreement. We use each source under its corresponding terms, do not redistribute restricted source records, replace source identifiers with paper-local pseudonyms in reported examples, and report evaluation results in aggregate.

Per-student behavioral models introduce risks beyond those of aggregate student modeling. They could be repurposed for student profiling, ranking, or differential treatment, and they may reproduce biases in the populations represented by their source corpora. The evaluated populations reflect online chess players, learners from the eleven most represented nationalities in the cleaned EFCAMDAT corpus, and students working within the grades 6 through 8 United States mathematics curriculum represented by FoundationalASSIST. We therefore scope the resulting models and artifacts to research on student simulation and tutor evaluation rather than student-facing assessment or consequential educational decisions. We also use the population-level Stage-1 simulator, rather than a per-student Stage-2 adapter, in tutor reinforcement learning so that policy optimization is not tied to one learner's behavioral record.

The expert evaluation was conducted under a protocol approved by our institution's Institutional Review Board, and all evaluators provided informed consent. Competitive chess players assessed model-generated guidance under blinded and order-shuffled conditions. The study did not deploy the resulting tutor to students or evaluate real-student learning outcomes. Establishing such outcomes requires a separate controlled deployment study with appropriate consent and oversight.

Reproducibility Statement

We provide the information needed to reconstruct the experiments from the authorized source data. Appendix B documents the data sources and access conditions, preprocessing procedures, record schemas, chronological train and held-out splits, evaluated student populations, and per-student sample counts. Appendix C gives the exact definitions and aggregation procedures for behavioral fidelity and guidance responsiveness. Appendix E reports the model releases, LoRA configurations, optimizer and scheduler settings, random seeds, decoding parameters, tutor-RL configuration, reward definitions, human-evaluation protocol, and compute usage. Appendix D and Appendix E.4 report the supporting ablations and variation across training runs. Restricted source corpora are not redistributed, but their access routes and the processing needed to reconstruct the benchmark are documented. An anonymized implementation, configuration files, and benchmark-construction scripts are provided in the supplementary material.

AI Use Statement

Generative AI tools were used to generate the natural-language tutor guidance in the semi-synthetic multi-turn datasets for chess and mathematics. Specifically, GPT-4o and GPT-5.1 generated tutor guidance for chess, and GPT-5.4 generated tutor guidance for mathematics, as described in Appendix F.2. Generative AI tools were also used to polish the language of the manuscript. The authors reviewed the final manuscript and take responsibility for its content, results, and claims.

References

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen

Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Indronil Bhattacharjee and Christabel Wayllace. Cold start problem: An experimental study of knowledge tracing models with new students, 2025. URL https://arxiv.org/abs/2505.21517.

Hao Cen, Kenneth Koedinger, and Brian Junker. *Learning Factors Analysis – A General Method for Cognitive Model Evaluation and Improvement*, pp. 164–175. Springer Berlin Heidelberg, 2006. ISBN 9783540351603. doi: 10.1007/11774303_17. URL http://dx.doi.org/10.1007/11774303_17.

Weihua Cheng, Hanwen Du, Chunxiao Li, Ersheng Ni, Liangdi Tan, Tianqi Xu, and Yongxin Ni. Uncertainty-aware knowledge tracing. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(27):27905–27913, April 2025. ISSN 2159-5399. doi: 10.1609/aaai.v39i27.35007. URL http://dx.doi.org/10.1609/aaai.v39i27.35007.

Alexis Chevalier, Jiayi Geng, Alexander Wettig, Howard Chen, Sebastian Mizera, Toni Annala, Max Jameson Aragon, Arturo Rodríguez Fanlo, Simon Frieder, Simon Machado, Akshara Prabhakar, Ellie Thieu, Jiachen T. Wang, Zirui Wang, Xindi Wu, Mengzhou Xia, Wenhan Xia, Jiatong Yu, Jun-Jie Zhu, Zhiyong Jason Ren, Sanjeev Arora, and Danqi Chen. Language models as science tutors, 2024. URL https://arxiv.org/abs/2402.11111.

Youngduck Choi, Youngnam Lee, Junghyun Cho, Jineon Baek, Byungsoo Kim, Yeongmin Cha, Dongmin Shin, Chan Bae, and Jaewe Heo. Towards an appropriate query, key, and value computation for knowledge tracing. In *Proceedings of the Seventh ACM Conference on Learning @ Scale*, L@S '20, pp. 341–344. ACM, August 2020. doi: 10.1145/3386527.3405945. URL http://dx.doi.org/10.1145/3386527.3405945.

Albert T. Corbett and John R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modelling and User-Adapted Interaction*, 4(4):253–278, 1995. ISSN 1573-1391. doi: 10.1007/bf01099821. URL http://dx.doi.org/10.1007/bf01099821.

David Dinucu-Jianu, Jakub Macina, Nico Daheim, Ido Hakimi, Iryna Gurevych, and Mrinmaya Sachan. From problem-solving to teaching problem-solving: Aligning llms with pedagogy using reinforcement learning, 2025. URL https://arxiv.org/abs/2505.15607.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping, 2020. URL https://arxiv.org/abs/2002.06305.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017. URL https://arxiv.org/abs/1703.03400.

Weibo Gao, Qi Liu, Linan Yue, Fangzhou Yao, Rui Lv, Zheng Zhang, Hao Wang, and Zhenya Huang. Agent4edu: Generating learner response data by generative agents for intelligent education systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(22):23923–23932, April 2025. doi: 10.1609/aaai.v39i22.34565. URL https://doi.org/10.1609/aaai.v39i22.34565.

Jeroen Geertzen, Theodora Alexopoulou, and Anna Korhonen. Automatic linguistic annotation of large scale l2 databases: The ef-cambridge open language database (efcamdat). In *Selected Proceedings of the 2012 Second Language Research Forum: Building Bridges between Disciplines*, pp. 240–254. Cascadilla Proceedings Project, 2014. URL https://www.lingref.com/cpp/slrf/2012/abstract3100.html.

Aritra Ghosh, Neil Heffernan, and Andrew S. Lan. Context-aware attentive knowledge tracing, 2020. URL https://arxiv.org/abs/2007.12324.

Aritra Ghosh, Jay Raspat, and Andrew Lan. *Option Tracing: Beyond Correctness Analysis in Knowledge Tracing*, pp. 137–149. Springer International Publishing, 2021. ISBN 9783030782924. doi: 10.1007/978-3-030-78292-4_12. URL http://dx.doi.org/10.1007/978-3-030-78292-4_12.

Elena L. Grigorenko and Robert J. Sternberg. Dynamic testing. *Psychological Bulletin*, 124 (1):75–111, July 1998. ISSN 0033-2909. doi: 10.1037/0033-2909.124.1.75. URL https://doi.org/10.1037/0033-2909.124.1.75.

Ilya Gusev. Pingpong: A benchmark for role-playing language models with user emulation and multi-model evaluation, 2024. URL https://arxiv.org/abs/2409.06820.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Hang Jiang, Xiajie Zhang, Xubo Cao, Cynthia Breazeal, Deb Roy, and Jad Kabbara. Personallm: Investigating the ability of large language models to express personality traits. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 3605–3627. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-naacl.229. URL http://dx.doi.org/10.18653/v1/2024.findings-naacl.229.

Kenneth R. Koedinger and Vincent Aleven. Exploring the assistance dilemma in experiments with cognitive tutors. *Educational Psychology Review*, 19(3):239–264, July 2007. ISSN 1573-336X. doi: 10.1007/s10648-007-9049-0. URL https://doi.org/10.1007/s10648-007-9049-0.

Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. Melu: Meta-learned user preference estimator for cold-start recommendation, 2019. URL https://arxiv.org/abs/1908.00413.

Unggi Lee, Jiyeong Bae, Dohee Kim, Sookbun Lee, Jaekwon Park, Taekyung Ahn, Gunho Lee, Damji Stratton, and Hyeoncheol Kim. Language model can do knowledge tracing: Simple but effective method to integrate language model and knowledge tracing task, 2024. URL https://arxiv.org/abs/2406.02893.

Xiujun Li, Zachary C. Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. A user simulator for task-completion dialogues, 2017. URL https://arxiv.org/abs/1612.05688.

Lichess. Lichess open database: standard rated games (may 2025 snapshot). https://database.lichess.org/standard/lichess_db_standard_rated_2025-05.pgn.zst, 2025. Database portal: https://database.lichess.org/. Released under Creative Commons CC0 1.0.

Jiayu Liu, Zhenya Huang, Tong Xiao, Jing Sha, Jinze Wu, Qi Liu, Shijin Wang, and Enhong Chen. Socraticlm: exploring socratic personalized teaching with large language models. *Advances in Neural Information Processing Systems*, 37:85693–85721, 2024a.

Naiming Liu, Zichao Wang, Richard Baraniuk, and Andrew Lan. Open-ended knowledge tracing for computer science education. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3849–3862. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.emnlp-main.254. URL http://dx.doi.org/10.18653/v1/2022.emnlp-main.254.

Zhengyuan Liu, Stella Xin Yin, Geyu Lin, and Nancy F. Chen. Personality-aware student simulation for conversational intelligent tutoring systems. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 626–642. Association for Computational Linguistics, 2024b. doi: 10.18653/v1/2024.emnlp-main.37. URL http://dx.doi.org/10.18653/v1/2024.emnlp-main.37.

Zitao Liu, Qiongqiong Liu, Jiahao Chen, Shuyan Huang, Boyu Gao, Weiqi Luo, and Jian Weng. Enhancing deep knowledge tracing with auxiliary tasks. In *Proceedings of the ACM Web Conference 2023*, WWW '23, pp. 4178–4187. ACM, April 2023. doi: 10.1145/3543507.3583866. URL http://dx.doi.org/10.1145/3543507.3583866.

F. M. Lord. *Applications of Item Response Theory To Practical Testing Problems*. Routledge, November 2012. ISBN 9781136557248. doi: 10.4324/9780203056615. URL http://dx.doi.org/10.4324/9780203056615.

Ryan Louie, Ananjan Nandi, William Fang, Cheng Chang, Emma Brunskill, and Diyi Yang. Roleplay-doh: Enabling domain-experts to create llm-simulated patients via eliciting and adhering to principles, 2024. URL https://arxiv.org/abs/2407.00870.

Xinyi Lu and Xu Wang. Generative students: Using llm-simulated student profiles to support question item evaluation, 2024. URL https://arxiv.org/abs/2405.11591.

Xiang Luo, Zhiwen Tang, Jin Wang, and Xuejie Zhang. Duetsim: Building user simulator with dual large language models for task-oriented dialogues, 2024. URL https://arxiv.org/abs/2405.13028.

Jakub Macina, Nico Daheim, Sankalan Pal Chowdhury, Tanmay Sinha, Manu Kapur, Iryna Gurevych, and Mrinmaya Sachan. Mathdial: A dialogue tutoring dataset with rich pedagogical properties grounded in math reasoning problems, 2023. URL https://arxiv.org/abs/2305.14536.

Jakub Macina, Nico Daheim, Ido Hakimi, Manu Kapur, Iryna Gurevych, and Mrinmaya Sachan. Mathtutorbench: A benchmark for measuring open-ended pedagogical capabilities of llm tutors, 2025. URL https://arxiv.org/abs/2502.18940.

Amogh Mannekote, Adam Davies, Jina Kang, and Kristy Elizabeth Boyer. Can llms reliably simulate human learner actions? a simulation authoring framework for open-ended learning environments, 2024. URL https://arxiv.org/abs/2410.02110.

Julia M. Markel, Steven G. Opferman, James A. Landay, and Chris Piech. Gpteach: Interactive ta training with gpt-based students. In *Proceedings of the Tenth ACM Conference on Learning @ Scale*, pp. 226–236. Association for Computing Machinery, July 2023. doi: 10.1145/3573051.3593393. URL https://doi.org/10.1145/3573051.3593393.

Daria Martynova, Jakub Macina, Nico Daheim, Nilay Yalcin, Xiaoyu Zhang, and Mrinmaya Sachan. Can llms effectively simulate human learners? teachers' insights from tutoring llm students. In *Proceedings of the 20th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2025)*, pp. 100–117. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.bea-1.8. URL http://dx.doi.org/10.18653/v1/2025.bea-1.8.

Kaushal Kumar Maurya, KV Aditya Srivatsa, Kseniia Petukhova, and Ekaterina Kochmar. Unifying ai tutor evaluation: An evaluation taxonomy for pedagogical ability assessment of llm-powered ai tutors, 2025. URL https://arxiv.org/abs/2412.09416.

Reid McIlroy-Young, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Aligning superhuman ai with human behavior: Chess as a model system, 2020. URL https://arxiv.org/abs/2006.01855.

Reid McIlroy-Young, Russell Wang, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Learning models of individual behavior in chess. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1253–1263. Association for Computing Machinery, August 2022. doi: 10.1145/3534678.3539367. URL https://doi.org/10.1145/3534678.3539367.

Benjamin D. Nye, Arthur C. Graesser, and Xiangen Hu. Autotutor and family: A review of 17 years of natural language tutoring. *International Journal of Artificial Intelligence in Education*, 24(4):427–469, December 2014. ISSN 1560-4292. doi: 10.1007/s40593-014-0029-5. URL https://doi.org/10.1007/s40593-014-0029-5.

OpenAI. GPT-4o System Card. https://openai.com/index/gpt-4o-system-card/, aug 2024.

OpenAI. GPT-5.4 Thinking System Card. https://openai.com/index/gpt-5-4-thinking-system-card/, mar 2026.

Paul Owoicho, Ivan Sekulić, Mohammad Aliannejadi, Jeffrey Dalton, and Fabio Crestani. Exploiting simulated user feedback for conversational search: Ranking, rewriting, and beyond, 2023. URL https://arxiv.org/abs/2304.13874.

Shalini Pandey and George Karypis. A self-attentive model for knowledge tracing, 2019. URL https://arxiv.org/abs/1907.06837.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, pp. 1–22. ACM, October 2023. doi: 10.1145/3586183.3606763. URL http://dx.doi.org/10.1145/3586183.3606763.

Joon Sung Park, Carolyn Q. Zou, Jonne Kamphorst, Niles Egan, Aaron Shaw, Benjamin Mako Hill, Carrie Cai, Meredith Ringel Morris, Percy Liang, Robb Willer, and Michael S. Bernstein. Llm agents grounded in self-reports enable general-purpose simulation of individuals, 2024. URL https://arxiv.org/abs/2411.10109.

Philip I. Pavlik, Jr., Hao Cen, and Kenneth R. Koedinger. Performance factors analysis–a new alternative to knowledge tracing. In *Artificial Intelligence in Education*, volume 200 of *Frontiers in Artificial Intelligence and Applications*, pp. 531–538. IOS Press, 2009. doi: 10.3233/978-1-60750-028-5-531. URL https://doi.org/10.3233/978-1-60750-028-5-531.

Letian Peng and Jingbo Shang. Quantifying and optimizing global faithfulness in persona-driven role-playing, 2024. URL https://arxiv.org/abs/2405.07726.

Chris Piech, Jonathan Spencer, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing, 2015. URL https://arxiv.org/abs/1506.05908.

Mark D. Reckase. The difficulty of test items that measure more than one ability. *Applied Psychological Measurement*, 9(4):401–412, December 1985. ISSN 1552-3497. doi: 10.1177/014662168500900409. URL http://dx.doi.org/10.1177/014662168500900409.

Vinay Samuel, Henry Peng Zou, Yue Zhou, Shreyas Chaudhari, Ashwin Kalyan, Tanmay Rajpurohit, Ameet Deshpande, Karthik Narasimhan, and Vishvak Murahari. Personagym: Evaluating persona agents and llms, 2025. URL https://arxiv.org/abs/2407.18416.

Alexander Scarlatos, Ryan S. Baker, and Andrew Lan. Exploring knowledge tracing in tutor-student dialogues using llms. In *Proceedings of the 15th International Learning Analytics and Knowledge Conference*, LAK 2025, pp. 249–259. ACM, March 2025a. doi: 10.1145/3706468.3706501. URL http://dx.doi.org/10.1145/3706468.3706501.

Alexander Scarlatos, Naiming Liu, Jaewook Lee, Richard Baraniuk, and Andrew Lan. Training llm-based tutors to improve student learning outcomes in dialogues, 2025b. URL https://arxiv.org/abs/2503.06424.

Alexander Scarlatos, Jaewook Lee, Simon Woodhead, and Andrew Lan. Simulated students in tutoring dialogues: Substance or illusion?, 2026. URL https://arxiv.org/abs/2601.04025.

Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR02, pp. 253–260. ACM, August 2002. doi: 10.1145/564376.564421. URL http://dx.doi.org/10.1145/564376.564421.

Yunfan Shao, Linyang Li, Junqi Dai, and Xipeng Qiu. Character-llm: A trainable agent for role-playing. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13153–13187. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.814. URL http://dx.doi.org/10.18653/v1/2023.emnlp-main.814.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Itamar Shatz. Refining and modifying the efcamdat: Lessons from creating a new corpus from an existing large-scale english learner language database. *International Journal of Learner Corpus Research*, 6(2):220–236, December 2020. ISSN 2215-1486. doi: 10.1075/ijlcr.20009.sha. URL http://dx.doi.org/10.1075/ijlcr.20009.sha.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297. Association for Computing Machinery, March 2025. doi: 10.1145/3689031.3696075. URL https://doi.org/10.1145/3689031.3696075.

Shashank Sonkar, Naiming Liu, Debshila Mallick, and Richard Baraniuk. Class: A design framework for building intelligent tutoring systems based on learning science principles. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 1941–1961. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.130. URL http://dx.doi.org/10.18653/v1/2023.findings-emnlp.130.

Katherine Stasaski, Kimberly Kao, and Marti A. Hearst. Cima: A large open access dialogue dataset for tutoring. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 52–64. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.bea-1.5. URL http://dx.doi.org/10.18653/v1/2020.bea-1.5.

Zhenwei Tang, Difan Jiao, Reid McIlroy-Young, Jon Kleinberg, Siddhartha Sen, and Ashton Anderson. Maia-2: A unified model for human-ai alignment in chess, 2024. URL https://arxiv.org/abs/2409.20553.

LearnLM Team, Abhinit Modi, Aditya Srikanth Veerubhotla, Aliya Rysbek, Andrea Huber, Brett Wiltshire, Brian Veprek, Daniel Gillick, Daniel Kasenberg, Derek Ahmed, Irina Jurenka, James Cohan, Jennifer She, Julia Wilkowski, Kaiz Alarakyia, Kevin R. McKee, Lisa Wang, Markus Kunesch, Mike Schaekermann, Miruna Pîslar, Nikhil Joshi, Parsa Mahmoudieh, Paul Jhun, Sara Wiltberger, Shakir Mohamed, Shashank Agarwal, Shubham Milind Phal, Sun Jae Lee, Theofilos Strinopoulos, Wei-Jen Ko, Amy Wang, Ankit Anand, Avishkar Bhoopchand, Dan Wild, Divya Pandya, Filip Bar, Garth Graham, Holger Winnemoeller, Mahvish Nagda, Prateek Kolhar, Renee Schneider, Shaojian Zhu, Stephanie Chan, Steve Yadlowsky, Viknesh Sounderajah, and Yannis Assael. Learnlm: Improving gemini for learning, 2024. URL https://arxiv.org/abs/2412.16429.

Silvia Terragni, Modestas Filipavicius, Nghia Khau, Bruna Guedes, André Manso, and Roland Mathis. In-context learning user simulators for task-oriented dialog systems, 2023. URL https://arxiv.org/abs/2306.00774.

The Stockfish developers. Stockfish. https://stockfishchess.org/, 2025. Source repository: https://github.com/official-stockfish/Stockfish. Recommended citation per project CITATION.cff at https://github.com/official-stockfish/Stockfish/blob/master/CITATION.cff.

Quan Tu, Shilong Fan, Zihang Tian, Tianhao Shen, Shuo Shang, Xin Gao, and Rui Yan. Charactereval: A chinese benchmark for role-playing conversational agent evaluation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11836–11850. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.638. URL http://dx.doi.org/10.18653/v1/2024.acl-long.638.

L. S. VYGOTSKY. *Mind in Society: Development of Higher Psychological Processes*. Harvard University Press, October 1980. ISBN 9780674576285. doi: 10.2307/j.ctvjf9vz4. URL http://dx.doi.org/10.2307/j.ctvjf9vz4.

Noah Wang, Z.y. Peng, Haoran Que, Jiaheng Liu, Wangchunshu Zhou, Yuhan Wu, Hongcheng Guo, Ruitong Gan, Zehao Ni, Jian Yang, Man Zhang, Zhaoxiang Zhang, Wanli Ouyang, Ke Xu, Wenhao Huang, Jie Fu, and Junran Peng. Rolellm: Benchmarking, eliciting, and enhancing role-playing abilities of large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pp. 14743–14777. Association for Computational Linguistics, 2024a. doi: 10.18653/v1/2024.findings-acl.878. URL http://dx.doi.org/10.18653/v1/2024.findings-acl.878.

Rose E. Wang, Ana T. Ribeiro, Carly D. Robinson, Susanna Loeb, and Dora Demszky. Tutor copilot: A human-ai approach for scaling real-time expertise, 2024b. URL https://arxiv.org/abs/2410.03017.

Rose E. Wang, Qingyang Zhang, Carly Robinson, Susanna Loeb, and Dorottya Demszky. Bridging the novice-expert gap via models of decision-making: A case study on remediating math mistakes, 2024c. URL https://arxiv.org/abs/2310.10648.

Xintao Wang, Yunze Xiao, Jen tse Huang, Siyu Yuan, Rui Xu, Haoran Guo, Quan Tu, Yaying Fei, Ziang Leng, Wei Wang, Jiangjie Chen, Cheng Li, and Yanghua Xiao. Incharacter: Evaluating personality fidelity in role-playing agents through psychological interviews, 2024d. URL https://arxiv.org/abs/2310.17976.

David Wood, Jerome S. Bruner, and Gail Ross. The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry*, 17(2):89–100, April 1976. ISSN 1469-7610. doi: 10.1111/j.1469-7610.1976.tb00381.x. URL http://dx.doi.org/10.1111/j.1469-7610.1976.tb00381.x.

Eamon Worden, Cristina Heffernan, Neil Heffernan, and Shashank Sonkar. Foundationalassist: An educational dataset for foundational knowledge tracing and pedagogical grounding of llms, 2026. URL https://arxiv.org/abs/2602.00070.

Tao Wu, Jingyuan Chen, Wang Lin, Mengze Li, Yumeng Zhu, Ang Li, Kun Kuang, and Fei Wu. Embracing imperfection: Simulating students with diverse cognitive levels using llm-based agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9887–9908. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.488. URL http://dx.doi.org/10.18653/v1/2025.acl-long.488.

Songlin Xu, Xinyu Zhang, and Lianhui Qin. Eduagent: Generative student agents in learning, 2024. URL https://arxiv.org/abs/2404.07963.

Songlin Xu, Hao-Ning Wen, Hongyi Pan, Dallas Dominguez, Dongyin Hu, and Xinyu Zhang. Classroom simulacra: Building contextual student generative agents in online education for learning behavioral simulation, 2025. URL https://arxiv.org/abs/2502.02780.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Ke Yang, Zixi Chen, Xuan He, Jize Jiang, Michel Galley, Chenglong Wang, Jianfeng Gao, Jiawei Han, and ChengXiang Zhai. Plugmem: A task-agnostic plugin memory module for llm agents, 2026. URL https://arxiv.org/abs/2603.03296.

Zhihao Yuan, Yunze Xiao, Ming Li, Weihao Xuan, Richard Tong, Mona Diab, and Tom Mitchell. Towards valid student simulation with large language models, 2026. URL https://arxiv.org/abs/2601.05473.

Jiani Zhang, Xingjian Shi, Irwin King, and Dit-Yan Yeung. Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pp. 765–774. International World Wide Web Conferences Steering Committee, April 2017. doi: 10.1145/3038912.3052580. URL http://dx.doi.org/10.1145/3038912.3052580.

Zheyuan Zhang, Daniel Zhang-Li, Jifan Yu, Linlu Gong, Jinchang Zhou, Zhanxin Hao, Jianxiao Jiang, Jie Cao, Huiqin Liu, Zhiyuan Liu, Lei Hou, and Juanzi Li. Simulating classroom education with llm-empowered agents, 2024. URL https://arxiv.org/abs/2406.19226.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. Swift: A scalable lightweight infrastructure for fine-tuning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(28):29733–29735, April 2025. ISSN 2159-5399. doi: 10.1609/aaai.v39i28.35383. URL http://dx.doi.org/10.1609/aaai.v39i28.35383.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

Doğuş Öksüz, Theodora Alexopoulou, Kateryna Derkach, and Ianthi Maria Tsimpli. The influence of l1 typology on the acquisition of the l2 english article: A large-scale corpus study. *Second Language Research*, 42(2):281–311, April 2026. ISSN 1477-0326. doi: 10.1177/02676583251395876. URL https://doi.org/10.1177/02676583251395876.

A Related Work

This appendix expands Section 2 with a per-paper survey along the same three lines: cognitive-state and behavior-prediction models (Sec. A.1), LLM-prompted student simulators (Sec. A.2), and tutor-side evaluation and optimization (Sec. A.3).

A.1 Cognitive-State and Behavior-Prediction Models

Knowledge tracing. Knowledge tracing infers a student's latent cognitive state from a sequence of observed responses and predicts future responses. Bayesian Knowledge Tracing maintains a per-skill Bernoulli mastery variable updated after each attempt with four parameters (prior, learn, slip, guess) (Corbett & Anderson, 1995); Deep Knowledge Tracing replaces hand-specified updates with a recurrent network over the interaction history (Piech et al., 2015); attentive variants condition on items and recent context for longer-range and item-similarity effects (Ghosh et al., 2020). Related state representations include key-value memory and transformer KT, auxiliary-task DKT, Item Response Theory and multidimensional IRT, and performance- and learning-factor analysis (Zhang et al., 2017; Pandey & Karypis, 2019; Choi et al., 2020; Liu et al., 2023; Lord, 2012; Reckase, 1985; Pavlik et al., 2009; Cen et al., 2006). Across the family the state is a compact latent updated from labeled response outcomes (correct/incorrect, sometimes latency, partial credit, or hints used), and the prediction target is the next item's outcome; the observation channel never includes free-form tutor guidance.

LLM-based knowledge tracing. From 2024 onward, question text and dialogue enter the KT stack. Integrating pretrained language-model representations of question text improves cold-start generalization over item-id baselines (Lee et al., 2024); extending KT to tutor-student dialogues with per-turn skill and correctness annotation beats classical KT especially under limited data (Scarlatos et al., 2025a); and uncertainty-aware variants add stochastic-embedding uncertainty (Cheng et al., 2025). The dialogue-KT entry is the closest published prior art for our $\mathcal{R}$ axis, being the only KT variant that ingests tutor language as input, yet all of these predict answer correctness from one shared model rather than generating a specific student's responses from a per-student one.

From correctness prediction to response generation. Option Tracing predicts which distractor a student selects rather than only whether they answer correctly (Ghosh et al., 2021); Open-Ended KT extends the idea to free-form responses in programming tasks, predicting the exact response rather than its correctness label, and is the canonical precursor of StudentSim's response-generation objective (Liu et al., 2022).

Human behavior cloning in chess. A parallel strand fits behavioral predictors directly to logs of human play. Maia trains separate move-prediction networks for rating bands and reproduces characteristic human errors at the target skill level (McIlroy-Young et al., 2020). Individual Maia specializes a pretrained Maia model to a particular player. In its data-scaling study, gains over non-personalized Maia emerge at 5,000 recorded games, while the 1,000-game condition does not improve over Maia (McIlroy-Young et al., 2022). Maia2 follows a complementary direction: it replaces rating-specific networks with one skill-conditioned model that improves population-level move prediction and coherence across skill levels (Tang et al., 2024). Individual Maia and Maia2 therefore model different granularities, per-player behavior and skill-conditioned population behavior, respectively; neither accepts free-form tutor guidance.

Where the gap shows up. This line ranges from population-wide models to independently specialized per-player models. All learn from behavioral logs, however, and none provides an architectural route by which a paragraph of explanation, a hint, or a worked example changes the predicted response. Under our metrics, Maia2 provides a strong behavioral reference for $\mathcal{F}$, but its $\mathcal{R}$ remains at a zero-instruction floor because guidance text has no input pathway (Section 5.3).

A.2 LLM-Prompted Student Simulators

User and student simulation. A second strand prompts a large language model to role-play a target user: build a textual description of the target (preferences, knowledge state, persona), prepend it to the conversation, and let the model continue conditioned on it. This recipe is used to

evaluate conversational and task-oriented dialogue agents by replaying simulated user turns (Owoicho et al., 2023; Terragni et al., 2023), and recurs as persona prompting, profile-conditioned simulation, synthetic-user generation for offline evaluation, and interactive goal-directed simulation, sometimes constrained with in-context exemplars or dual-LLM coordination (Luo et al., 2024). A parallel line measures whether such role-plays stay consistent with a described character, using the resulting scores as benchmarks, as training data, or as rewards for fine-tuning persona-following (Peng & Shang, 2024; Samuel et al., 2025; Wang et al., 2024d; Jiang et al., 2024; Wang et al., 2024a; Shao et al., 2023; Tu et al., 2024; Gusev, 2024).

Critical observation. The simulator is a frozen LLM with a prompt, pretrained to generate plausible language rather than to map a description of a learner onto the behavioral consequences of that cognitive state. Two failure modes follow: the simulator solves problems beyond the described student's level, or its surface mimicry overshoots and misses problems the student would solve. Both appear in our chess data, where GPT-5.4 reaches $\mathcal{F} = 0.23$ and GPT-4o $\mathcal{F} = 0.22$, below state-tracking baselines, even as they attain higher $\mathcal{R}$ than state-tracking models can (Section 5.2). We classify any system whose output is a student-style response (an answer, an essay, a confusion message) as a student simulator, and reserve "judge" or "LLM-as-judge" for systems whose output is a scalar score or rubric verdict (Zheng et al., 2023); the proxy-reward and LLM-as-judge line lives in Sec. A.3.

A.2.1 Student-Simulation Validity

The validity of LLM-based student simulation has emerged as an open question. The *competence paradox* holds that a broadly capable LLM asked to behave as a partially knowledgeable learner produces fluent but unreliable student text, because its pretrained reasoning and helpfulness habits override the competence description; this line reframes simulator validity as constrained generation under an explicit epistemic-state specification, with state evolution under instruction as a required axis rather than an emergent side effect of prompting (Yuan et al., 2026). Empirical work measures simulated-student quality directly and reports systematic mismatches with real students in solve rate and error type (Scarlatos et al., 2026), and other work proposes architectural changes such as cognitive prototype modules or controlled imperfect reasoning to elicit student-like errors instead of correcting them away (Wu et al., 2025). Teacher-facing studies add that instructors find LLM students overly attentive, emotionally flat, linguistically too sophisticated, and inconsistent across turns (Martynova et al., 2025). Our $\mathcal{F}$ axis targets exactly this per-individual fidelity gap: each simulator is trained against a real learner's behavioral data and scored per student on held-out records that learner produced. $\mathcal{R}$, by contrast, scores whether the simulator produces the canonical corrected response under tutor guidance (Appendix F.1), which is what a tutor RL loop rewards: guidance that moves the student to the right answer. How closely $\mathcal{R}$ tracks an individual student's literal post-guidance behavior is a complementary, counterfactual question that controlled studies with real students under matched guidance can address, a natural extension of the metric.

A.2.2 Educational Student Simulators

Educational simulators instantiate the prompting recipe in several ways: half-synthetic tutoring dialogues from teacher–LLM-student pairing or crowdworker role-play (Macina et al., 2023; Stasaski et al., 2020); multi-agent classrooms and generative learner agents with reflective memory, class-control mechanisms, or longitudinal real-student datasets (Zhang et al., 2024; Gao et al., 2025; Xu et al., 2025); profile- and personality-conditioned simulators tying behavior to cognitive-ability and personality traits (Liu et al., 2024b); and application-driven simulators for item-quality evaluation, teaching-assistant training, and learner-data generation (Lu & Wang, 2024; Markel et al., 2023; Xu et al., 2024). A per-individual sub-line specifies agents per target rather than per group: patient-simulation elicits per-target natural-language behavioral principles for a prompted LLM, the closest per-individual specification methodology even though it is prompt-based and non-educational (Louie et al., 2024), and social-science work constructs per-individual agents from a person's self-reports, builds generative agents with retrieval-based memory, or equips LLM agents with task-agnostic plugin memory modules, using prompting, retrieval, and external memory rather than weight specialization and without evaluating state evolution under instruction (Park et al., 2024; 2023; Yang et al., 2026). These approaches vary along granularity (persona, profile, group, or individual), conditioning interface (prompt, prompt-plus-retrieval, or weight-level fine-tuning), and evaluation target (stylistic

plausibility, a downstream task, or response-level fidelity to a specific learner); StudentSim is per-individual, weight-level, and response-level with the additional $\mathcal{R}$ guidance-response axis, a combination we are not aware of in prior educational work.

A.3 Tutor-Side Evaluation and Optimization

Tutor models, judges, and benchmarks. The third line studies the AI tutor side of the loop. Tutor-specialized models supervise on expert-curated or synthetic tutoring dialogues (Chevalier et al., 2024); proxy-reward and LLM-as-judge methods supply scalar feedback by querying a generic LLM against a quality rubric, with the target being rubric quality or human preference rather than a specific student's response (Zheng et al., 2023). Benchmark work moves from single-rubric judges toward multi-dimensional pedagogical evaluation and documents a tension between a tutor model's subject expertise and its pedagogical skill (Macina et al., 2025; Maurya et al., 2025). Classroom-scale studies test tutor-assist and tutor-as-replacement settings against real students and report learning gains, in some cases with particular benefit to under-served populations (Wang et al., 2024b), and LLM-tutor instruction-tuning reports parallel pedagogical-quality gains under expert preference evaluation (Team et al., 2024). A separate line conditions tutor output on a student descriptor or a structured pedagogical strategy (a profile, a recent error, an instructional style, or step-by-step scaffolding) (Sonkar et al., 2023; Nye et al., 2014; Liu et al., 2024a), extending the intelligent tutoring systems (ITS) lineage exemplified by Squirrel AI, ALEKS, and Carnegie Learning; StudentSim differs from ITS-era student models in handling open-ended natural-language guidance and training per individual rather than fitting a small skill graph with hand-coded production rules.

Simulator-grounded rewards. The closest cluster to our downstream setup uses a knowledge-tracing model or a prompted LLM student as the reward signal in a tutor optimization loop rather than a generic rubric. One pattern scores candidate tutor utterances by a knowledge-tracing model's predicted post-turn correctness, combines that with a rubric judge, and trains the tutor via direct preference optimization (Scarlatos et al., 2025b); another runs a full online RL loop in which the tutor and a prompted LLM student converse for multiple turns and the tutor is rewarded by the simulated student's post-dialogue solve rate plus a pedagogy judge, optimized via GRPO (Dinucu-Jianu et al., 2025). What recurs across this cluster is that the reward is grounded in a knowledge-tracing model that predicts only answer correctness or a prompted LLM student, rather than a simulator trained on real student behavioral data; where the student is a prompted LLM, the model that judges its simulated learner is typically the same generic LLM, so the reward is only as well-grounded as that LLM's persona-following, which the validity literature in Sec. A.2.1 documents to be unstable. Section 6 supplies the missing pathway: a frozen StudentSim, trained on real learner data and calibrated on $\mathcal{F}$ and $\mathcal{R}$, is plugged into a tutor RL loop as the reward source, with the reward depending on whether the simulated student's response improves under tutor guidance ($r = \text{score}(A_{\text{rev}}) - \text{score}(A_{\text{prev}})$). Because the signal is grounded in real-learner-derived behavior rather than a prompted persona or generic rubric, expert raters judge the resulting tutor as more accurate, better-guided, and more personalized than tutors rewarded by a prompted-simulator reward or no reinforcement learning (Sec. 6).

B Per-Benchmark Data and Tasks

This appendix documents the source corpus, raw record fields, training-instance formats, and corpus-level statistics for each of the three domains the StudentSim framework instantiates. Each subsection follows the same five-block structure: data source, raw record, single-turn record format $S_i = \{(x, m)\}$, multi-turn record format $T_i = \{(x, m, \tau, m^*)\}$, and a corpus-statistics tabular. Symbols S_i and T_i are defined in Section 3. Worked examples below show real fields with identifiers anonymized as Player A (chess) or Student A (second-language English writing and mathematics).

B.1 The StudentSimEval Evaluation Protocol

StudentSimEval fixes, once and independently of any method, both the population of evaluated students and the per-student held-out records that every method is scored on. It evaluates a frozen roster of 30 chess players, 15 L2 learners, and 15 math students (60 in total). For each student it fixes two held-out sets, disjoint from all training data: a single-turn set S_i that scores behavioral

fidelity $\mathcal{F}$, and a multi-turn set T_i that scores guidance responsiveness $\mathcal{R}$. The split is chronological: each student's earlier records form the training pool and their later records are held out for S_i and T_i, so every score reflects prediction of a student's later behavior from their earlier records. Every method is scored on the same frozen held-out sets. Methods that support per-student fitting or in-context conditioning receive information derived from the same per-student training records. Fixed pretrained domain models such as Maia2 are used without per-student adaptation because the available individual record volume does not support the thousands-of-games personalization regime evaluated by Individual Maia (McIlroy-Young et al., 2022). The held-out sets are materialized once as frozen split files and reused verbatim, so they are a property of the protocol, not of any method; a new method is evaluated by running it on the same files. The per-student held-out sizes are the "Held-out" columns of Table 1: $|S_i| = 5{,}000$ and $|T_i| = 4{,}000$ in chess; $|S_i| = 26$ and $|T_i| = 40$ in L2; $|S_i| = 66$ in math, with $|T_i|$ ranging over 21 to 99. The multi-turn set is split evenly across each domain's guidance modes: four in chess (error remediation, comparative, strategic, Socratic), two in L2 (point-based, rule-based), and three in math (error remediation, Socratic, conceptual).

The protocol equalizes the number of held-out records per student wherever the data allows, so that no student dominates a population mean. Where a domain has more records than the budget needs, a cap gives every student the same count; where equalizing would leave too little to estimate the metric, the protocol keeps every available record instead. Chess uses an absolute cap of $5{,}000$ single-turn positions and $1{,}000$ multi-turn records per guidance mode, subsampled so every player contributes the same count from a larger available pool. For the scarcer single-turn data in L2 and math, we equalize to the smallest student's held-out size, giving $|S_i| = 26$ (L2) and $|S_i| = 66$ (math) for every student, and L2 caps its multi-turn set at 20 records per correction mode. Math multi-turn records are the case where equalizing would leave too little to estimate $\mathcal{R}$, since a multi-turn record exists only for a problem the student first answered incorrectly, so the protocol keeps every eligible wrong-answer problem, fanned across the three guidance modes (879 records in total, 293 per mode). Training draws a separate capped subset of each student's records, so the held-out portion is reserved and never enters any training stage.

B.2 CHESS

Data source. The chess records come from the public Lichess game-database export at `https://database.lichess.org/` (Lichess, 2025), specifically the May 2025 monthly snapshot of standard-time-control human games. Lichess is an open-source online chess platform on which every published game is between two real human players; no game in the export is engine-vs-engine or otherwise synthetic. The platform also publishes Stockfish position evaluations for many games, and our pipeline uses these where available rather than running Stockfish from scratch. The export is released under CC0, which permits unrestricted research, redistribution, and modification.

Raw record. A Lichess game is a sequence of half-moves recorded in PGN, plus per-game metadata (the two players' Lichess handles, their ELO ratings at the time of the game, time control, opening ECO code, and final outcome). For each half-move we extract the position before the move in FEN notation, the player's actual move in UCI, and engine-derived quality features for that move: the engine top-k move list with their centipawn evaluations, the player's centipawn loss for the played move, a binary `is_best_move` flag from the engine ranking, the index of the played move within the engine top-k list, and the move-index within the game (used downstream to bucket positions by phase). The fields kept in each per-position record are therefore: `player_handle`, `player_elo`, `time_control`, `game_outcome`, `move_idx`, `fen_before`, `player_uci`, `engine_top_k`, `cp_loss`, `is_best_move`, and `player_move_rank`. We discard pre-move clock times, evaluation comments embedded in the PGN, and game-chat fields. Positions are bucketed by phase into very-early opening (`move_idx` < 10), opening (`move_idx` $\in [10, 19]$), middlegame ($[20, 39]$), and endgame (≥ 40).

Single-turn record format $S_i = \{(x, m)\}$. The single-turn input x contains a structured student state followed by the FEN of the current position and a request for the next move in UCI. In chess, the structured state combines two forms of evidence derived from prior human play. Player-level statistics summarize relatively stable tendencies, including recent win rate, best-move accuracy, average centipawn loss, playing style, piece usage, recent game history, and opening preferences. Maia2 complements these historical summaries with a position-level behavioral prior, represented by

its probability distribution over the five moves most characteristic of human play at the player's skill level. Together, these signals ground the student state that the language model is trained to realize for the individual player. The response m is the player's actual move on the held-out position, in UCI. A real held-out record from Player A illustrates the format:

Listing 1: A single-turn record for Player A

```
x:
  Player context: Recent win rate: 54.6%; Best-move accuracy: 32.84%;
  Avg centipawn loss: 215.74; Style: offensive;
  Piece usage: bishop 14.00%, knight 19.68%, queen 17.38%;
  Last 3 game results: loss, loss, loss; ...
  The current position in FEN notation is:
  2kr1b1r/pp2pppp/5nb1/8/2BN4/4N3/PPPB1PPP/R3K2R b KQ - 0 12
  You are playing as Black. Respond in UCI format.

m:
  d8d4
```

Multi-turn record format $T_i = \{(x, m, \tau, m^*)\}$. The multi-turn record reuses the same x and the same player move m, then appends two additional fields. Multi-turn records are drawn from positions outside the very-early opening (`move_idx` ≥ 10), because an opening move reflects a player's repertoire and preparation, where there is no single move to correct toward. Tutor guidance τ is a natural-language tutor message in one of four pedagogical styles (*error remediation*, *comparative*, *strategic*, *Socratic*), generated by an LLM tutor (GPT-4o or GPT-5.1, selected per record by whether the player's move fell outside the engine's top-k list) conditioned on the FEN, the player's actual (often non-optimal) move, the engine best move, and the engine continuations for both. Each style is realized by a fixed prompt template that constrains the tutor's framing: error-remediation pinpoints the mistake and states the correction, comparative contrasts the played move with the engine's top alternatives, strategic foregrounds long-term plan considerations, and Socratic prompts the player with one or two leading questions. The reference move m^* is the engine best move at fixed depth on the same FEN. A real strategic-style record from Player A:

Listing 2: A multi-turn record for Player A, strategic style

```
x:
  Player context (as above)
  + FEN 7k/5R2/2pr2p1/2p4p/P5p1/1B1P4/2P3PP/6K1 w - - 0 30
  You are playing as White.

m:
  h2h3

tau (strategic):
  "In this position, the strategic focus should be on maximizing piece
  activity and pawn structure. h2h3 does not address the immediate
  strategic opportunities; consider creating a passed pawn with a4a5,
  activating the bishop with b3c4, or improving the rook via f7f8 ..."

m*:
  a4a5
```

The four styles differ in how much of the correction they hand over. The record below shows all four on a single king-and-pawn endgame from Player B (Fig. 7), who pushed a flank pawn with `b5b4` and lost a winning position (engine evaluation $+425$ to -656 centipawns); the engine best move is the king activation `g6f7`.

Listing 3: The same position under all four guidance styles, Player B

```
x:
  Player context (as above)
```

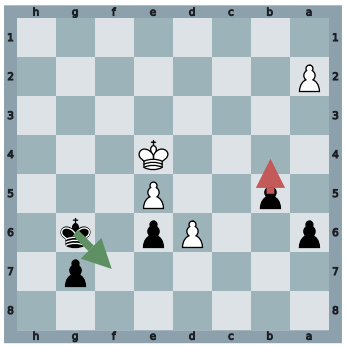


Figure 7: **The position behind the four-style example.** Black to move, FEN `8/6p1/p2Pp1k1/1p2P3/4K3/8/P7/8 b - - 0 41`. Red marks the move the player actually made (`b5b4`); green marks the engine best move m^* (`g6f7`).

```
  + FEN 8/6p1/p2Pp1k1/1p2P3/4K3/8/P7/8 b - - 0 41
  You are playing as Black.

================================================================================

m:
  b5b4

================================================================================

tau (error remediation):
  "Your move b4 initiates an unnecessary pawn push that loses critical
      tempo and
  worsens your position significantly. After b4, White plays d7,
      forcing your king to
  retreat to h7, and quickly promotes their pawn with d8=Q. This
      sequence leaves Black
  materially and positionally worse, as White's queen dominates the
      board and Black's
  pawns are left defenseless. Instead, the best move is Kf7. By
      activating your king
  with Kf7, you challenge White's king on e4 and prepare to hinder
      White's advancing
  pawns, starting with their central d-pawn. After Kf7, if White
      responds with Ke3, you
  can play b4 to activate your passed pawn while keeping your king
      near the action on
  d4-e4. This keeps White's king under pressure and delays their pawn
      promotion
  significantly. Effective king positioning is critical in endgames,
      and Kf7 sets up a
  much stronger resistance plan. To improve in positions like this,
      prioritize using
  your king actively in endgames to control key squares and block pawn
      advances. Avoid
  pushing pawns recklessly, as that can waste time and leave your king
      unable to
  respond to threats. Look for ways to hinder your opponent's plans
      and play precise
  moves that maximize your king's influence on the board."

================================================================================

tau (comparative):
  "Let's analyze three candidate moves: your move b4, Kf7, and Kh6.
      Your move, b4, aims
```

```
  to activate the pawn structure on the queenside and potentially
      disrupt White's pawn
  advancement. The continuation b4 d7 Kh7 d8=Q a5 follows. This move
      allows White to
  push the d7 pawn, creating a strong threat with the promotion to a
      queen on d8. The
  resulting position places White in a commanding position due to the
      powerful queen
  while Black's pawn play hasn't resulted in any significant counter-
      threats. Next,
  consider Kf7. This move shifts the Black king toward greater
      activity, attempting to
  control key squares and potentially disrupt White's advancement on
      the d-file. The
  continuation shows Kf7 Ke3 b4 Kd4 Ke8. Here, the Black king aims to
      support the
  defense of the back rank and control pivotal central squares,
      limiting White's
  capability to convert the d-pawn effectively. This active king
      positioning can lead
  to improved chances down the line. Lastly, Kh6 serves as a more
      defensive option. Its
  continuation is Kh6 d7 g5 d8=Q g4. This move attempts to prepare for
       an active pawn
  push on the kingside while avoiding immediate threats from the
      advancing d-pawn.
  However, promoting a queen on d8 remains a formidable option for
      White, which Kh6
  barely manages to counter. Comparing these approaches, piece
      activity and king
  mobility, seen in Kf7, prove more decisive in confronting White's
      pawn strength on
  the d-file than launching pawns themselves in b4 or the tentative
      king safety of
  Kh6."

================================================================================

tau (strategic):
  "In this position, the key strategic themes are the advancement of
      passed pawns and
  king activity. White has a dangerous passed pawn on d6, and Black's
      priority should
  be neutralizing this threat. The student's move b4 advances a flank
      pawn, but misses
  immediate central concerns. In endgames, especially pawn races, it's
       crucial to
  ensure that the opponent's most advanced pawns are blocked or
      captured before
  advancing your own. Consider moves that bring the king closer to
      White's advanced
  pawns, like Kf7. Such moves help restrain the opponent's pawn while
      activating your
  king. The strategic principle is that in pawn races, central pawn
      control and king
  activation will often dictate the balance of power. Moves aimed at
      mobilizing the
  king to confront the central passed pawn ensure that you stand ready
       to handle the
  advancing threat efficiently. In a similar situation, remember to
      focus on
  neutralizing central threats first before considering peripheral
      pawn advances."
```

```
================================================================================

tau (Socratic):
  "1. What happens to the balance of the position after b4? Consider
      the consequences
  for both sides' pawn structures. 2. How does White respond if Black
      tries b4? How
  effective is this in stopping the d-pawn? 3. What role does the
      Black king play in
  this position? Can it impact the pawn promotion directly? 4. Compare
       the move b4 with
  a potential move that increases king activity. How do these moves
      change the pressure
  on White's advance? 5. By considering moves other than b4, how can
      Black use king
  movement to create counterplay or directly block White's threats?"

================================================================================

m*:
  g6f7
```

Corpus statistics. The May 2025 standard-time-control export covers tens of millions of games drawn from a multi-million-player pool on Lichess. The Stage-1, Stage-2, and per-student held-out counts that draw from this pool are summarized in Table 1 of Section 5.1; the Stage-2 pool is nested inside the Stage-1 pool. Players are sampled to span a broad ELO band rather than a narrow rating slice, with a minimum game-count threshold so that each sampled player has enough recorded games to support both pooled training and held-out evaluation. The multi-turn evaluation set is restricted to non-best-move positions outside the very-early-opening band.

Table 5: Chess corpus statistics. Stage-1 / Stage-2 training counts and per-student held-out evaluation counts ($\mathcal{F}$ test, $\mathcal{R}$ test) are reported in Table 1.

Field	Value
Source corpus	Lichess May 2025 standard-time-control export, CC0
Engine analysis	Stockfish, top-4 candidate moves, depth adapted to game phase (8 in the very-early opening up to 15 in the endgame)
Per-record response space	legal moves at the position, in UCI
Guidance modes	error remediation, comparative, strategic, Socratic (4 total)

B.3 Second-Language English Writing

Data source. The L2 records come from EFCAMDAT, the EF–Cambridge Open Language Database (Geertzen et al., 2014), a large open-access corpus of English essays written by real human L2 English students (real students learning English as a second language) on the Education First online-school platform. We use the second release, which contains roughly 1.18M essays from roughly 174,000 students across the platform's 16 proficiency levels, which are aligned with the Common European Framework of Reference for Languages (CEFR), the six-level scale running from A1 for a beginner to C2 for a near-native writer. Two derived sub-corpora are relevant: the *cleaned subcorpus* (Shatz, 2020) restricts to students from the eleven most-represented nationalities and supplies normalized texts up to level 15, and the *cleaned error-coded subcorpus* (Öksüz et al., 2026) adds part-of-speech tagging and per-token error-category annotations on top of the cleaned subset. Access requires an academic affiliation and a signed user agreement administered by the Cambridge research lab. No essays in our pipeline are synthetic.

Raw record. Each EFCAMDAT essay has: a student identifier (the field is named `learner_id` in the source corpus), the student's nationality and current EFCAMDAT level, which the profile block

also renders as its approximate CEFR equivalent (levels 1–15 mapped onto A1 through C2), the lesson topic and unit, the required key words from the lesson, the raw essay text, the error-coded version of the essay (per-token tags by category and sub-category), the canonical corrected version of the essay, the platform's grade for the essay, and lightweight metadata (essay length, time spent). At training-instance construction time we keep `learner_id`, `nationality`, `efcamdat_level` (and derived CEFR), `topic_id`, `topic_title`, `level_unit`, `key_words`, `raw_text`, `corrected_text`, the per-token error-category annotations from the error-coded subcorpus, and the platform-assigned grade; we discard time-spent and other UI-side fields. EFCAMDAT marks each error in an essay with three pieces of information: the text the student wrote, the correction for it, and one of fifteen category labels (spelling, word choice, article, preposition, verb tense, plural, agreement, word order, and so on). A student who writes *I go to school yesterday* would have one error recorded, with original text `go`, correction `went`, and label `verb tense`. We keep all three fields. Counting the labels over a learner's earlier essays produces the error-frequency line of the student-profile block, which for one learner on our roster reads `Common error patterns: spelling(37), word choice(30), extra word (delete)(28), missing word(23), article(17)`, and a few of the original-to-correction pairs are listed under it as concrete examples. Each single marked error also becomes one multi-turn record, where the tutor turn points at that original text and the student's target reply is the correction. The seven LanguageTool issue-type buckets used by the fidelity metric (misspelling, grammar, typographical, style, uncategorized, whitespace, inconsistency) are not raw fields: they are computed downstream from `raw_text` via the LanguageTool open-source checker, and their formal role in the metric is defined in Appendix C.

Single-turn record format $S_i = \{(x, m)\}$. The single-turn input x is a student-profile-conditioned essay prompt: the student's nationality and CEFR level, an essay-history summary (essays written so far, average length, average platform grade), the top error categories the student makes most often with concrete examples of recent corrections, a short list of recent essays on similar topics with their grades, and the new lesson task (topic title, level-unit pair, required key words). The response m is the student's actual raw essay, with their characteristic errors preserved. EFCAMDAT is released to researchers on request and is not ours to redistribute, so we give the record schema instead of a verbatim record; the rendered prompt retains EFCAMDAT's source-side "Learner" field labels:

Listing 4: Single-turn record schema (L2). EFCAMDAT is released to researchers on request, so we show the schema in place of a verbatim record.

```
x:
  Learner profile: nationality; current EFCAMDAT level (with CEFR
      equivalent);
  essays written so far, average length, average platform grade;
  common error patterns with counts;
  recent essays (most recent last), each with its lesson topic and
      grade
  Now write a response to this lesson task: topic; level and unit; key
       words

m:
  the learner's own essay for that lesson task
```

Multi-turn record format $T_i = \{(x, m, \tau, m^*)\}$. The multi-turn record reuses the same x and the same student essay m, and then appends a tutor turn τ and a canonical corrected fragment m^*. The tutor message τ is generated in one of two surface-form templates: *point-based*, which states that an error of a given kind is present in the highlighted span and asks the student to fix it, and *rule-based*, which states the underlying linguistic rule first and then asks the student to apply it to the highlighted span. Only the surface phrasing of the two style templates is generated; the highlighted span and the target correction are drawn directly from EFCAMDAT's own error-coded annotations. The reference m^* is the correction EFCAMDAT records for that span. For errors of the delete type, their target is the literal token `delete`. The rule-based schema continues the same record:

Listing 5: Multi-turn record schema (L2), shown as a schema for the same reason.

```
tau (rule-based, verb tense):
  the grammatical rule that governs the flagged span, followed by an
      excerpt
  of the learner's essay with the offending fragment marked, and a
      request to
  apply the rule to that fragment

m*:
  the teacher's corrected fragment
```

Corpus statistics. We build on the error-coded subcorpus of EFCAMDAT (Öksüz et al., 2026), which is the release that carries the span-level teacher corrections our multi-turn records need. It holds 620,206 essays from 116,312 learners. Per-learner volume is low and right-skewed: a learner writes 5.3 essays on average and 3 at the median, 33.3% of learners have a single essay, and 71.4% have five or fewer. Learners are ranked by recorded volume and the most active are taken for the Stage-1 and Stage-2 pools, so that every selected learner supports both training and a frozen held-out split; the Stage-2 learners therefore sit above the corpus median, contributing on the order of 90 essays each. The Stage-1, Stage-2, and per-student held-out counts that draw from this corpus are summarized in Table 1 of Section 5.1; the Stage-2 pool is nested inside the Stage-1 pool. Students are sampled to span the level range and the eleven most-represented nationalities, with a minimum essay-count threshold to support held-out evaluation. The seven LanguageTool issue-type buckets used by the fidelity metric (misspelling, grammar, typographical, style, uncategorized, whitespace, inconsistency) are computed downstream and not directly retained as raw fields.

Although the per-student single-turn held-out budget on L2 ($S = 26$ in Table 1) is smaller in essay count than the per-student budgets for chess or mathematics, the per-student estimator of $\mathcal{F}$ is driven by error events rather than by essays. Each held-out essay typically contains on the order of 5–20 LanguageTool-flagged events, so the 26 essays per student aggregate to roughly 10^2–10^3 per-100-word error events across the seven LanguageTool buckets, and the per-student error-density vector that $\mathcal{F}$ scores is estimated from that event-level total rather than from the 26 essay-level outcomes. The per-bucket density is also a continuous quantity over the seven buckets, not a binary correctness outcome, which lowers the variance of the per-student estimator at this scale. The formal definition of $\mathcal{F}$ for L2 and the per-bucket aggregation are given in Appendix C.

Table 6: Second-language English writing corpus statistics. Stage-1 / Stage-2 training counts and per-student held-out evaluation counts ($\mathcal{F}$ test, $\mathcal{R}$ test) are reported in Table 1. Shaded rows are the subcorpus we use and the learners we train on; the two unshaded scale rows describe the full release for context.

Field	Value
Source corpus	EFCAMDAT, error-coded subcorpus (Öksüz et al., 2026)
Full second release, all essays	∼1,180,310
Full second release, all students	∼174,743
Error-coded subcorpus, essays	620,206
Error-coded subcorpus, learners	116,312
Essays per learner in the error-coded subcorpus	5.3 mean, 3 median, 141 max
Essays per learner, the 15 Stage-2 learners	92.4 mean, 87 median, 84–141 range
LanguageTool issue-type buckets (fidelity metric)	misspelling, grammar, typographical, style, uncategorized, whitespace, inconsistency (7 total)

B.4 MATHEMATICS

Data source. The mathematics records come from the FoundationalASSIST dataset (Worden et al., 2026), a release of student–problem interactions from the ASSISTments online-tutoring platform aligned to the Illustrative Mathematics curriculum and, through it, to the Common Core State Standards, the United States grade-by-grade specification of what mathematics a student should master. Each problem in the release carries the standard codes it exercises, which is where the skill labels used below come from. Every interaction is by a real human student; no responses are

synthetic. The dataset is hosted on Hugging Face under a gating mechanism that requires accepting a Responsible Use Agreement before download. FA-specific processing notes (Excel-date corruption fix, LLM answer-key audit, free-text vs. multiple-choice filter) are summarized below.

Raw record. Each FA interaction has a hashed student identifier, a problem identifier, the problem body (HTML with embedded MathML), the platform's claimed-correct answer key, the answer-type (Numeric, Algebraic Expression, Exact Match, Exact Fraction, Numeric Expression, Ordering, Drop Down, Multiple Choice, Check All That Apply), the student's free-text answer, a binary `discrete_score`, and timestamps; supplementary tables map problem to skill (224 unique skills) and skill to Common Core standard. The fields kept in each interaction record are: `interaction_id` (the unique row key, since $\sim 5\%$ of student–problem pairs occur on multiple days), `user_id`, `problem_id`, `skill`, `problem_body` (cleaned to plain text with paragraph breaks preserved and inline MathML rendered as `numerator/denominator`), `claimed_answer` (after the date-remap and audit steps below), `answer_type`, `student_free_text`, and `discrete_score`. Three FA-specific quirks are handled before downstream record construction: *i)* a subset of 149 problems whose `Fill-in Answers` field was Excel-coerced from fractions to dates (`5/6` stored as `6-May`, `3/10` as `10-Mar`, `8/15` as `15-Aug`) is repaired deterministically by a date-pattern remap; *ii)* every remaining `(problem_id, claimed_answer)` pair, the deterministically repaired ones excepted, is audited by GPT-4o, which solves the problem independently and returns `AGREE` or `DISAGREE` against the platform key, counting equivalent surface forms as agreement, and any pair whose verdict is not `AGREE` is dropped, which removes keys carrying arithmetic errors and keys for problems whose figures were lost in text extraction; and *iii)* we keep the problems the platform scores as free text (Numeric, Algebraic Expression, Exact Match, Exact Fraction, Numeric Expression, Ordering, Drop Down), since those are the problems that record what other students wrote, which is where our distractors come from.

Single-turn record format $S_i = \{(x, m)\}$. The single-turn input x is a student-profile prompt followed by a single problem in a four-way multiple-choice form. The student-profile block contains the pseudonym, overall and recent accuracy, per-skill performance summary (correct / attempted, with percentages), and the most recent attempts on the current problem's skill in compact form (correct flag, problem stem, student's free-text response). The multiple-choice form is constructed deterministically from the underlying free-text record: this student's own recorded answer is one option, and the other options are the answers other students most often gave on the same `problem_id`, after a normalization step that collapses equivalent surface forms and removes any that coincide with this student's answer. The options are shuffled by a fixed per-record seed and labelled A through D, and the prompt asks the simulator to pick a single letter, so m is the letter carrying this student's own answer. A problem on which other students gave too few distinct answers to fill the options is dropped. FoundationalASSIST is distributed under a Responsible Use Agreement, so we give the record schema in place of a verbatim record:

Listing 6: Single-turn record schema (math). FoundationalASSIST is distributed under a Responsible Use Agreement, so we show the schema in place of a verbatim record. Student names are pseudonyms.

```
x:
  Student profile: pseudonym; overall and recent accuracy;
  per-skill performance (correct / attempted); most recent attempts on
       the skill
  Now solve this problem: [skill] problem text
  Multiple-choice options: A) ...; B) ...; C) ...; D) ...
  Pick the letter (A-D). Respond with only the letter.

m:
  the letter carrying the student's own recorded answer
```

Multi-turn record format $T_i = \{(x, m, \tau, m^*)\}$. The multi-turn record extends the single-turn form with a tutor turn τ and a reference letter m^*. Tutor guidance τ is a natural-language tutor message in one of three styles (*error remediation*, *Socratic*, *conceptual*), generated by an LLM tutor

(default model GPT-5.4). The reference m^* is the letter carrying the correct answer. A Socratic record:

Listing 7: Multi-turn record schema (math), shown as a schema for the same reason.

```
x:
  student-profile block (as above) plus the four-way multiple-choice
      form

m:
  the letter carrying the student's own recorded answer

tau (Socratic):
  one or two guiding questions about the step the student got wrong

m*:
  the letter carrying the canonical correct answer
```

Corpus statistics. The FA release covers 5,000 students contributing 215–421 interactions each (350.7 on average, 349 at the median), for 1,753,384 interactions over 3,368 unique problems and 224 unique skills. Answers are first-attempt correct on 61.6% of interactions, so a student answers 129.7 distinct problems incorrectly on average. Students are ranked by recorded volume on the same rule as L2; the 15 Stage-2 students average 404.9 recorded interactions each, of which 122.7 problems are answered incorrectly and can therefore carry a correction turn. The Stage-1, Stage-2, and per-student held-out counts that draw from this corpus, after the date-remap, audit, and free-text filter described above, are summarized in Table 1 of Section 5.1; the Stage-2 pool is nested inside the Stage-1 pool. The multi-turn held-out test set is restricted to interactions on which the student answered incorrectly.

Table 7: Mathematics corpus statistics. Stage-1 / Stage-2 training counts and per-student held-out evaluation counts ($\mathcal{F}$ test, $\mathcal{R}$ test) are reported in Table 1. The shaded row is the population our simulators are trained on; the rows above it describe the release it is drawn from.

Field	Value
Source corpus	FoundationalASSIST (ASSISTments)
Curriculum	Illustrative Mathematics, Common Core aligned
Total students	5,000
Total interactions	1,753,384
Unique problems / unique skills	3,368 / 224
Interactions per student	350.7 mean, 349 median, 215–421 range
Interactions per student, the 15 Stage-2 students	404.9 mean, of which 122.7 problems answered incorrectly
Per-record response space	four-way MC over $\{A, B, C, D\}$

C Per-Benchmark Metric Definitions

Section 3 states $\mathcal{F}$ and $\mathcal{R}$ in symbols. This appendix gives the per-benchmark formula, the operational recipe used at evaluation, and the principled reason that operationalization is the right one for that benchmark's response space. The same symbols are used throughout: π_i (real student i), M_i (their simulator), $S_i = \{(x, m)\}$ (held-out single-turn records), $T_i = \{(x, m, \tau, m^*)\}$ (held-out multi-turn records). The population aggregation

$$\mathcal{F} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{F}_i, \qquad \mathcal{R} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{R}_i, \tag{1}$$

is the same in every benchmark; only the per-instance and per-student definitions change to track the response space.

The single principle running through the six choices below is that each $\mathcal{F}_i$ scores how closely the simulator's response on a held-out input matches what student π_i actually recorded, and each $\mathcal{R}_i$

scores how closely the simulator's response after reading guidance τ matches the canonical corrected response m^*. Where a per-instance score in $[0, 1]$ is well-defined from a single recorded observation, $\mathcal{F}_i$ is the mean of that score. Where the recorded supervision is categorical with an empirical candidate set, $\mathcal{F}_i$ is top-1 accuracy on that categorical set. Where the canonical corrected response is uniquely defined, $\mathcal{R}_i$ is exact match against it.

C.1 Chess Fidelity: Top-1 Move Accuracy

Definition.

$$\mathcal{F}_i \;=\; \frac{1}{|S_i|} \sum_{(x,m)\in S_i} \mathbb{1}\left[\arg\max_{m'\in\mathcal{L}(x)} P_{M_i}(m' \mid x) \;=\; m\right], \tag{2}$$

where $\mathcal{L}(x)$ is the set of legal moves at position x and m is the move student π_i actually played.

Computation at evaluation.

1. For each player i, S_i is the held-out set of single-turn records (board position, recorded move). The held-out games are disjoint from training games (Appendix B).
2. For each $(x, m) \in S_i$, decode the simulator M_i's top-1 next move on x under greedy decoding; call it $\hat{m}$.
3. Per-instance score is $\mathbb{1}[\hat{m} = m] \in \{0, 1\}$.
4. $\mathcal{F}_i$ is the mean per-instance score over S_i; $\mathcal{F}$ aggregates by Equation (1).

Why this choice. The chess move space at any position is finite (typically tens of legal moves), and the player's record contains exactly one chosen move per position; there is no recorded distribution to compare against. Top-1 accuracy uses only what is recorded and is therefore well-posed from one observation per position. Several plausible alternatives answer a different question or require supervision that the benchmark does not record: *i)* a move-distribution divergence (for example KL or JSD against the player's empirical distribution at x) requires repeated visits to the same position by the same player to estimate any target distribution, and Lichess records do not supply that; *ii)* top-k inclusion accuracy requires choosing k, and any $k > 1$ admits simulators that hedge across the legal-move set rather than commit to the player's move. Centipawn loss relative to the player's move measures how badly the simulator plays in absolute terms, not how closely it tracks the specific player; the latter is what $\mathcal{F}$ is for.

C.2 Chess Guidance Responsiveness: Top-1 Against the Engine-Best Move

Definition.

$$\mathcal{R}_i \;=\; \frac{1}{|T_i|} \sum_{(x,m,\tau,m^*)\in T_i} \mathbb{1}\left[\arg\max_{m'\in\mathcal{L}(x)} P_{M_i}(m' \mid x, m, \tau) \;=\; m^*\right], \tag{3}$$

where T_i contains held-out positions on which π_i played a non-optimal move m, τ is the tutor guidance addressing that mistake, and m^* is the engine-recommended move (Stockfish at fixed depth, treated as the canonical correction).

Computation at evaluation.

1. For each player i, T_i is the held-out set of multi-turn records on positions where π_i played a non-optimal move; the four guidance modes (error remediation, comparative, strategic, Socratic) are evenly represented.
2. For each record, the simulator receives the three-message conversation prefix (x, m, τ) and is queried for its corrected move; we read its top-1 prediction $\hat{m}^*$ under greedy decoding.
3. Per-instance score is $\mathbb{1}[\hat{m}^* = m^*] \in \{0, 1\}$.
4. $\mathcal{R}_i$ is the mean per-instance score over T_i; $\mathcal{R}$ aggregates by Equation (1).

Why this choice. After receiving guidance τ, the simulator's response is again a single legal move and the canonical correction m^* is the engine-recommended move, so top-1 against m^* is the direct comparison in the response space. The most plausible alternatives shift the measurement target away from guidance-following: *i)* centipawn-loss reduction credits any move that happens to be objectively strong, even when τ explicitly steers elsewhere; *ii)* improvement over m is automatic for any near-random change away from a known blunder; *iii)* top-k inclusion admits a hedging simulator on every record whose m^* is anywhere in its k-best list. Top-1 against m^* matches the strict question $\mathcal{R}$ is meant to answer, namely whether the simulator updates to the move that the guidance is steering toward.

C.3 L2 Fidelity: Error-Pattern Density Match

Definition. For a held-out essay record $(x, m) \in S_i$, let $\hat{m}$ be the simulator's generated essay on prompt x, let $\rho_e(t)$ denote the density of LanguageTool[1] issue type e on essay text t (errors of type e per 100 words), and let $\rho_{\text{tot}}(t) = \sum_{e \in \mathcal{C}} \rho_e(t)$ denote the total error density on t. Define the per-record fidelity score

$$\mathcal{F}_i^{(x,m)} \;=\; \tfrac{1}{2}\, d(\rho_{\text{tot}}(\hat{m}),\, \rho_{\text{tot}}(m)) \;+\; \tfrac{1}{2\,|\mathcal{C}|} \sum_{e \in \mathcal{C}} d(\rho_e(\hat{m}),\, \rho_e(m))\,, \qquad d(a,b) = 1 - \tfrac{|a-b|}{\max(a,b)}, \tag{4}$$

where $d(a, b) \in [0, 1]$ (with $d(0, 0) := 1$) is the density-similarity kernel and $\mathcal{C}$ is the set of seven LanguageTool issue types: misspelling, grammar, typographical, style, uncategorized, whitespace, inconsistency. Then $\mathcal{F}_i$ is the mean of $\mathcal{F}_i^{(x,m)}$ over S_i.

Computation at evaluation.

1. For each learner i, S_i is the held-out set of essay prompts paired with the learner's recorded essay m; $\hat{m}$ is the simulator's free-form essay generated on x.
2. Run LanguageTool on $\hat{m}$ and on m; bin issues by issue type to obtain per-category counts and a total count, divide by essay length to get densities ρ_e and ρ_{tot}.
3. Compute $\mathcal{F}_i^{(x,m)}$ via Equation (4).
4. $\mathcal{F}_i$ is the mean of $\mathcal{F}_i^{(x,m)}$ over S_i; $\mathcal{F}$ aggregates by Equation (1).

Why this choice. L2 essays are free-form text, so essay-level surface match is not a fidelity signal: no two essays the same learner writes are identical at the surface, and a strong baseline that produces a clean fluent essay would score well on lexical similarity even though the recorded learner essay is full of mistakes. The signal that the metric needs to reward is the learner's overall error rate and issue-type profile. Equation (4) operationalizes that signal with two terms guarding distinct failure modes: the total-density term rules out a "too clean" simulator that has the right error categories at near-zero rates, and the per-category mean rules out a simulator that hits the right total volume with a category profile that does not match the learner. The two-term split is symmetric and equally weighted; we do not tune the $0.5/0.5$ split. The seven LanguageTool issue types are the issue-type taxonomy LanguageTool itself emits; we do not collapse or relabel them. Surface-form alternatives (BLEU, ROUGE, chrF) reward fluency and topic-keyword overlap, both of which a strong general-purpose model satisfies independent of any learner profile; under those alternatives a simulator that ignores the learner's typical errors can score higher than one that reproduces them, which can invert the fidelity ranking on this corpus.

C.4 L2 Guidance Responsiveness: Canonical-Correction Exact Match

Definition. For each record $(x, m, \tau, m^*) \in T_i$, let $\hat{m}^*$ be the simulator's corrected fragment after reading (x, m, τ), and let $n(\cdot)$ denote the normalization that strips leading and trailing whitespace, lowercases, and collapses the parenthesized edit marker '(delete)' to the literal token 'delete'. Then

$$\mathcal{R}_i \;=\; \frac{1}{|T_i|} \sum_{(x,m,\tau,m^*) \in T_i} \mathbb{1}[\, n(\hat{m}^*) \;=\; n(m^*)\,]\,. \tag{5}$$

[1] https://languagetool.org

Computation at evaluation.

1. For each learner i, T_i is the held-out set of multi-turn correction records, spanning the two correction modes (point-based, rule-based).
2. For each record, the simulator receives the three-message conversation prefix (x, m, τ) and is queried for the corrected fragment; we read its free-form output $\hat{m}^*$.
3. Per-instance score is $\mathbb{1}[n(\hat{m}^*) = n(m^*)] \in \{0, 1\}$.
4. $\mathcal{R}_i$ is the mean per-instance score over T_i; $\mathcal{R}$ aggregates by Equation (1).

Why this choice. A tutor correction in the L2 corpus names a specific canonical corrected fragment m^*, and the question $\mathcal{R}$ is asking is whether the simulator produces that fragment after reading τ. Normalized exact match against m^* is the closest operationalization of that match under the corpus's annotation scheme; the normalization $n(\cdot)$ removes only whitespace, casing, and the syntactic edit marker, so it does not absorb genuine semantic differences. Softer alternatives are more permissive than the corpus's correction criterion: *i)* BLEU or chrF on the corrected fragment scores a simulator that produces the original fragment lightly edited as nearly equivalent to one that produces the canonical correction, which mixes "made the requested edit" with "made some edit lexically close to the original"; *ii)* sentence-level embedding similarity rewards staying on topic. Exact match also aligns with the corpus tutor-correction protocol, which records a single accepted corrected fragment per instance.

C.5 Math Fidelity: Top-1 Accuracy on a Four-Way Multiple-Choice Form

Definition. Each held-out problem x for student i is recast as a four-way multiple-choice instance whose option set $\mathcal{A}(x) = \{A, B, C, D\}$ contains *i)* the literal answer m that student π_i recorded on this problem (placed at the letter $\ell^* \in \{A, B, C, D\}$ by a deterministic shuffle keyed on x) and *ii)* three distractor letters, each holding an alternative answer drawn from the most common answers given by other students on the same problem (after case-insensitive string deduplication against the student's recorded answer, and excluding the 200 Stage-1 and 15 Stage-2 students themselves). The student's recorded answer is included whether it is canonically correct or wrong; on the 990-record math evaluation set 70.6% of records have a canonically correct recorded answer and 29.4% have a wrong one, and the simulator is asked to predict that specific letter in both cases. The per-record fidelity is top-1 accuracy of the simulator on this multiple-choice prompt,

$$\mathcal{F}_i^{(x,m)} \;=\; \mathbb{1}\left[\arg\max_{\ell \in \mathcal{A}(x)} P_{M_i}(\ell \mid x) \;=\; \ell^*\right], \tag{6}$$

and $\mathcal{F}_i$ is the mean of Equation (6) over S_i. When the empirical distractor pool for a problem holds fewer than three eligible answers after deduplication, $|\mathcal{A}(x)|$ falls to 3 for that record (we never pad with synthetic distractors); $K = 3$ occurs in 5 of the 990 records in the math evaluation set.

Computation at evaluation.

1. For each student i, S_i is the held-out set of math problems that the student recorded an answer to; each problem is converted to the four-way multiple-choice form once, with the per-record letter mapping ℓ^* fixed by a deterministic shuffle keyed on the problem id.
2. For each record, run the simulator on the four-way multiple-choice prompt and decode greedily for a single letter token.
3. Per-instance score is $\mathbb{1}[\hat{\ell} = \ell^*] \in \{0, 1\}$, where $\hat{\ell}$ is the simulator's decoded letter.
4. $\mathcal{F}_i$ is the mean per-instance score over S_i; $\mathcal{F}$ aggregates by Equation (1).

Why this choice. In math, the fidelity target is whether the simulator predicts the specific answer that this student would record on this problem, given a candidate set that contains the answers students actually give on the problem. Top-1 accuracy on a four-way multiple-choice form is the direct operationalization of that target on a categorical response space. Three operational details motivate the construction. First, scoring a free-text math answer directly as raw text would make per-record likelihoods or string comparisons incomparable across simulators because they would

depend on tokenization of the numerical string, on the model's prior over arbitrary number tokens, and on string-normalization choices; recasting the problem as a four-way form removes those confounds and pins the comparison to a controlled categorical set. Second, the candidate set is constructed from real student behavior on the same problem: the student's recorded answer plus the three most common other-student answers. This makes the comparison conditional on an empirical confusion set for the problem, rather than on random distractors or arbitrary string perturbations, and a correct letter prediction therefore reflects that the simulator distinguishes this student's specific answer from the answers other students actually give on the same problem. Third, top-1 accuracy is symmetric across simulators that expose logits and closed-source APIs that do not: greedy single-token decoding of a letter is supported uniformly, so every model is scored on the same observable quantity without depending on top-k logprob caps (GPT-5.4 exposes only top-5). We use $K = 4$ rather than larger K to keep the four letters distinct and to keep the empirical distractor pool well-populated on every problem; in the small fraction of records where fewer than three distinct other-student answers exist after deduplication, the record is kept with $K = 3$ rather than padded with synthetic options.

C.6 Math Guidance Responsiveness: Top-1 Accuracy on a Four-Way Post-Guidance Multiple-Choice Form

Definition. For each record $(x, m, \tau, m^*) \in T_i$, the simulator reads a three-message conversation prefix consisting of *i)* a problem prompt x recast in four-way multiple-choice form whose option set contains the student's recorded answer m and three other-student-answer distractors on the same problem, *ii)* the letter ℓ_m that maps to m under that record's deterministic shuffle, and *iii)* the tutor correction τ delivered as natural-language prose, and then decodes *iv)* the simulator's post-guidance letter prediction. The post-guidance candidate set $\mathcal{A}(x, \tau)$ reuses the same four-letter shuffle as $\mathcal{A}(x)$, with the canonical corrected answer m^* occupying its letter ℓ^{m^*} in that shuffle. The per-record score is top-1 accuracy of the simulator's letter at the post-guidance turn,

$$\mathcal{R}_i \;=\; \frac{1}{|T_i|} \sum_{(x,m,\tau,m^*)\in T_i} \mathbb{1}\left[\arg\max_{\ell \in \mathcal{A}(x,\tau)} P_{M_i}(\ell \mid x, m, \tau) \;=\; \ell^{m^*} \right]. \tag{7}$$

Computation at evaluation.

1. For each student i, T_i is the held-out set of multi-turn records on problems where π_i initially produced a wrong answer m and a tutor delivered a correction τ; the three guidance modes (error remediation, Socratic, conceptual) are evenly represented.
2. For each record, a single deterministic shuffle keyed on the problem id places the student's recorded answer m at letter ℓ_m and the canonical corrected answer m^* at letter ℓ^{m^*}; the same shuffle is used at both the student turn and the post-guidance turn.
3. The simulator receives the three-message prefix $(x_{\text{MC}}, \ell_m, \tau)$, where τ is the tutor correction delivered as natural-language prose, and decodes greedily for a single letter token $\hat{\ell}$.
4. Per-instance score is $\mathbb{1}[\hat{\ell} = \ell^{m^*}] \in \{0, 1\}$.
5. $\mathcal{R}_i$ is the mean per-instance score over T_i; $\mathcal{R}$ aggregates by Equation (1).

Why this choice. After reading a tutor correction that addresses the student's wrong answer, the question $\mathcal{R}$ asks is whether the simulator now arrives at the canonical corrected answer. We measure this on the same categorical four-way form used by $\mathcal{F}$ for two reasons. First, training and evaluation share the same form: the simulator is trained on multiple-choice records at both turns, and decoding a letter at evaluation matches the training distribution rather than crossing a free-text / categorical boundary that would penalize the simulator for an artifact of formatting. Second, the candidate set is constructed from the same empirical other-student answer pool used at the student turn, so the comparison is held against the answers students do actually give on this problem; the canonical corrected answer competes against three real alternative forms rather than against random distractors, and the metric rewards a simulator that resolves to the corrected form among realistic competitors. We considered free-text exact match against m^* and rejected it because string normalization absorbs or fails to absorb form differences (e.g., 0.5 versus $.5$ versus $1/2$) inconsistently across simulators, and that surface-form noise dominates the per-record signal under exact match; the four-way form with empirical distractors converts surface-form variation into the candidate set itself and lets the metric score whether the simulator selects the corrected canonical letter among the realistic alternatives.

Reproducibility note: bf16 nondeterminism on small samples. On the open-source per-student simulator (LoRA over a bf16 Qwen3-4B-Instruct base), greedy single-token decoding on borderline-logit records can flip across runs because of nondeterministic bf16 accumulation. The effect is per-record and averages out at scale: on a 315-record cap-7 subsample of the math evaluation set the same checkpoint scored $\mathcal{R} = 0.565$ on a first pass and $\mathcal{R} = 0.952$ on a re-run, while on the full 879-record no-cap evaluation set the population mean is stable across re-runs at the headline value reported in Table 3. The closed-source baselines (GPT-4o, GPT-5.4) are queried with API temperature 0 and are much less sensitive to within-checkpoint jitter. For faithful comparison the headline math number is reported on the full no-cap 879-record set; we recommend reproducing on the same set rather than on a downsample.

C.7 Per-Instruction-Mode $\mathcal{R}$ Breakdown

Each domain's multi-turn evaluation set covers multiple tutor guidance modes (Section 3). The tables below break the population-mean $\mathcal{R}$ from Table 3 down by mode within each domain. Per-mode aggregation is by mode within each per-student T_i and then averaged equally across students within the mode. Per-mode totals are reported only when the held-out evaluation records partition evenly across modes within each T_i; in math the per-student T_i ranges from 21 to 99 records jagged across students but the modes are evenly represented per student (counts of 293 per mode in the 879-record total).

Table 8: **Chess: $\mathcal{R}$ by guidance mode** across the four chess tutor modes. Population mean across the 30 chess players' held-out multi-turn records, equal records per mode within each player; subscripts are standard deviations across the three training seeds of Table 22. Maia2 reads only the board position and the player rating, so the four modes present it with the same input and its $\mathcal{R}$ does not decompose by mode.

mode	Maia2	GPT-4o	GPT-5.4	StudentSim
error remediation	N/A	0.9550	0.9652	$\mathbf{0.9959}_{\pm 0.0003}$
comparative	N/A	0.8778	0.7696	$\mathbf{0.9581}_{\pm 0.0017}$
strategic	N/A	0.7075	0.6223	$\mathbf{0.9076}_{\pm 0.0013}$
Socratic	N/A	0.5217	0.5174	$\mathbf{0.7649}_{\pm 0.0012}$

Table 9: **L2: $\mathcal{R}$ by guidance mode** across the two L2 tutor modes (point-based, rule-based). Population mean across the 15 L2 learners' held-out multi-turn records, 20 records per mode within each learner.

mode	base (Qwen3-4B)	GPT-4o	GPT-5.4	StudentSim
point-based	0.0233	0.3667	0.5767	$\mathbf{0.6533}_{\pm 0.0145}$
rule-based	0.0167	0.4100	0.6133	$\mathbf{0.6300}_{\pm 0.0557}$

Table 10: **Math: $\mathcal{R}$ by guidance mode** across the three math tutor modes (error remediation, Socratic, conceptual). Population mean across the 15 math students' held-out multi-turn records, 293 records per mode in the 879-record total (modes evenly represented within each per-student T_i).

mode	base (Qwen3-4B)	GPT-4o	GPT-5.4	StudentSim
error remediation	0.819	0.877	0.809	$\mathbf{0.9909}_{\pm 0.0020}$
Socratic	0.481	0.560	0.655	$\mathbf{0.8783}_{\pm 0.0232}$
conceptual	0.539	0.645	0.666	$\mathbf{0.8851}_{\pm 0.0104}$

Across all three domains, StudentSim outperforms the strongest baseline on every guidance mode, including the hardest, most indirect ones, for example chess Socratic and the math Socratic and conceptual settings. In these modes the tutor steers the student with leading questions or conceptual hints without stating which move or answer is correct, so the simulator has to infer the correction from the tutor's reasoning. That makes these modes the strictest test of guidance responsiveness.

D TRAINING PIPELINE ABLATIONS AND PARAMETER SWEEPS

This appendix reports the chess-domain sweeps that set the default training recipe for the reference simulators, together with a robustness check on the pooled two-turn fraction. We run these sweeps only in chess. Chess is the most challenging of the three domains to retrain end-to-end (since LLMs are less familiar with playing chess than learning a second-language or doing mathematics), and it uses the same two training stages and the same guidance-conditioning mechanism as the other two domains. We then transfer the resulting recipe to second-language English writing and to mathematics without per-knob retuning. This appendix therefore establishes a default recipe, not a claim that each knob has the same optimum in every domain. We first ablate the main pipeline choice of cross-student pooling in Stage 1, then examine three recipe settings: *i)* input modality (text vs. vision), *ii)* the fraction of guidance data in pooled-training, and *iii)* the composition of the guidance-mode mix.

We report two summaries. $\mathcal{F}$ on chess is the population mean of the per-student behavioral-fidelity score, measured on held-out positions on randomly selected 3 students. $\mathcal{R}$ on chess is the Turn-2 move accuracy on a held-out evaluation set of guided interactions. The board positions in this evaluation set are held out from both training stages, and the set covers all four chess guidance modes (error remediation, Socratic, strategic, comparative) with an equal number of positions per mode. The equal-count construction ensures that $\mathcal{R}$ reflects overall guidance responsiveness rather than capability on whichever mode happens to be most represented in evaluation, and it also makes per-mode breakdowns directly comparable. Both metrics are defined formally in Section 5.2 and Section 5.3. Each sweep is evaluated as a matched comparison within its own run configuration and student subset, so rows inside a table are directly comparable, while rows from different tables come from separate ablation runs.

D.1 WHAT STAGE-1 POOLING CONTRIBUTES

This ablation asks what the pooled Stage-1 corpus contributes in the main pipeline, holding the optimization budget fixed. In the reference condition, Stage 1 trains on 100,000 chess records pooled from 100 players, giving 391 optimizer steps at effective batch 256. In the matched ablation, Stage 1 instead trains on one player's own 1,000 records repeated 100 times, which produces the same 391 optimizer steps under the same effective batch, LoRA rank and scaling, learning rates, warmup, and seed. Stage 2 is identical in both conditions: the model is specialized on that player's own 1,000 records for 12 optimizer steps. This construction holds the update budget fixed and changes only the number of distinct students represented in Stage 1. We evaluate on three randomly sampled players.

Table 11: Ablation of cross-student pooling in chess Stage 1. The reference arm uses the default pooled Stage-1 corpus of 100,000 records from 100 players. The matched ablation replaces this with one player's own 1,000 records repeated 100 times, which preserves the Stage-1 optimizer budget (391 steps) and all training hyperparameters. Both arms then run the same Stage-2 specialization on that player's own 1,000 records (12 steps). Results are averaged over three ramdomly sampled players.

Stage-1 corpus	$\mathcal{F}$	$\mathcal{R}$
pooled across 100 players *(default)*	0.5131	0.9003
one player's own records, repeated	0.4602	0.8276

Cross-student pooling is what the pooled stage contributes. In Table 11, the reference arm reaches $\mathcal{F} = 0.5131$ and $\mathcal{R} = 0.9003$, while the repeated-single-student arm reaches $\mathcal{F} = 0.4602$ and $\mathcal{R} = 0.8276$. The same pattern holds for each player individually: fidelity drops from 0.4558 to 0.3992, from 0.4660 to 0.4186, and from 0.6174 to 0.5628, while responsiveness drops from 0.9093 to 0.8363, from 0.8928 to 0.8238, and from 0.8990 to 0.8228.

With Stage-1 and Stage-2 step counts, schedules, and data volume matched, the gain in the reference arm comes from the Stage-1 corpus spanning many students. Repeating one player's limited records for the same number of updates does not reproduce the effect of pooling across students.

D.2 Input Modality: Text vs. Vision

Chess is one of the three domains where a non-textual encoding of the student's working state is available: the board position can be rendered as an image and consumed by a vision-language base. We compared the text-only recipe (Qwen3-4B-Instruct, FEN plus features rendered as a string) against a matched vision-language recipe (Qwen3-VL-4B-Instruct, 400×400 board image plus the same text features), keeping every other hyperparameter fixed at the chess defaults.

Table 12: Modality sweep on chess. The board image adds 0.001 to $\mathcal{F}$ and requires a vision tower in the training loop; the two arms are matched on effective batch size.

Modality	$\mathcal{F}$	**training cost (relative)**
text-only (Qwen3-4B-Instruct)	0.536	$1.0\times$
vision-language (Qwen3-VL-4B-Instruct)	0.537	$\approx 2.0\times$

Table 12 shows that the visual board does not improve $\mathcal{F}$ in any meaningful sense. The textual board encoding (FEN plus the candidate-move and history features described in Section 4) already specifies the full game state relevant to move selection, so the rendered image adds little new information beyond what the text encoding already carries. We therefore use the text-only recipe in every domain. The other two domains are already text-native, so we do not repeat the modality sweep there.

D.3 Robustness to the Pooled Two-Turn Fraction

The pooled-training mix contains both single-turn samples (board $\rightarrow$ move) and two-turn samples (board $\rightarrow$ move; AI tutor guidance $\rightarrow$ revised move). The mixing ratio ρ controls the fraction of two-turn samples in the pooled stage. We fix $\rho = 0.2$ in all production runs, and the sweep below checks whether the paper's conclusions depend on that setting.

Table 13: Multi-turn mixing-ratio sweep on chess (sequential sampler, fixed Turn-2 suffix). Adding any nonzero amount of guidance data lifts $\mathcal{R}$ from 0.17 to above 0.80. From there $\mathcal{R}$ stays on a plateau, and $\mathcal{F}$ is unaffected by ρ within ± 0.004. The production setting $\rho = 0.20$ sits inside the plateau.

multi-turn ratio ρ	$\mathcal{F}$	$\mathcal{R}$	$\Delta\mathcal{R}$ **vs.** $\rho = 0$
0.000 (single-turn only)	0.490	0.167	n/a
0.025	0.489	0.826	$+0.659$
0.05	0.488	0.849	$+0.682$
0.10	0.489	0.816	$+0.649$
0.20 *(default)*	**0.487**	**0.859**	$\mathbf{+0.692}$
0.40	0.491	0.878	$+0.711$
0.80	0.487	0.857	$+0.690$

Two facts stand out. First, $\mathcal{F}$ is essentially invariant in ρ across the full range, so fidelity is stable under the Stage-1 multi-turn mixing ratio and no fidelity result hinges on this setting. Second, $\mathcal{R}$ reaches its high-value regime as soon as any guidance data is present and stays there, so the guidance results at $\rho = 0.2$ sit well inside a plateau. That plateau is not automatic: without a fixed Turn-2 suffix that re-asks for the updated move, some runs at intermediate ρ fail on error remediation by stopping after the reasoning trace instead of emitting a move. Pairing sequential sampling with the Turn-2 suffix removes that failure mode and produces the wide plateau in Table 13.

D.4 Guidance-Mode Composition

Chess uses four guidance modes (Section 5.3): error remediation (the tutor diagnoses the weakness in the student's move, then motivates the correct answer with concrete consequences or alternative candidates), strategic (the tutor states a plan such as "activate the king-side rook"), comparative (the tutor contrasts other stronger candidate moves), and Socratic (the tutor asks a question that prompts the student to derive the move themselves, e.g. "what is Black threatening on the next move?"). We include error remediation because direct correction is one end of the assistance dilemma (Koedinger & Aleven, 2007): when a student is stuck or repeatedly failing, the appropriate tutor action is to

supply the move and the reason for it, and a faithful simulator population has to cover that end of the spectrum as well. We include the Socratic mode to represent guided discovery at the indirect end of the tutoring spectrum, where the tutor prompts the student to resolve the correction. Because the tutor does not state which move to play, this mode is demanding for the simulator, which must combine the student's characteristic style with the tutor's reasoning prompt. Mixing four modes during pooled-training is meant to prevent the simulator from collapsing onto whichever mode is easiest.

We swept the per-mode composition at fixed $\rho = 0.2$ in three ways: *i) leave-one-out*, dropping each mode in turn, *ii) single-mode training*, keeping only one mode, and *iii) Socratic weight scaling*, multiplying the Socratic mode's weight while keeping the others at 1. Table 14, Table 15, and Table 16 report $\mathcal{R}$ broken down by evaluation mode.

Table 14: Leave-one-out sweep on chess. Each row drops one guidance mode from the pooled-training mix and evaluates on all four modes. Three of the four removals *improve* overall $\mathcal{R}$, including the per-mode score for the dropped mode itself, indicating that the modes are largely complementary at training time.

training mix (weights err:strat:comp:soc)	$\mathcal{R}$	**error rem.**	**strategic**	**comparative**	**Socratic**
1:1:1:1 *(uniform default)*	0.836	0.962	0.880	0.890	0.611
0:1:1:1 (drop error remediation)	0.859	0.969	0.864	0.919	0.683
1:1:0:1 (drop comparative)	0.869	0.984	0.878	0.934	0.682
1:1:1:0 (drop Socratic)	0.849	0.986	0.884	0.903	0.621
1:0:1:1 (drop strategic)	0.820	0.926	0.854	0.846	0.654

Table 15: Single-mode training on chess. Each row trains the pooled-training mix on one guidance mode only and evaluates on all four. Even one-mode training reaches $\mathcal{R} \in [0.72, 0.80]$, against 0.836 for the four-mode default. Comparative-only transfers best to the other three modes, and error-remediation-only transfers worst.

training mode (single mode at full weight)	$\mathcal{R}$	**error rem.**	**strategic**	**comparative**	**Socratic**
all four *(uniform default, ref)*	0.836	0.962	0.880	0.890	0.611
comparative only	0.798	0.926	0.850	0.865	0.552
strategic only	0.790	0.859	0.857	0.883	0.562
Socratic only	0.769	0.864	0.816	0.878	0.519
error remediation only	0.720	0.898	0.751	0.781	0.449

Table 16: Weight-scaling sweep on chess: scaling the Socratic mode while keeping the other three at 1. Multiplying the Socratic weight raises the Socratic per-mode score from 0.611 to 0.670 ($+5.9$ pp at $4\times$). The other three modes shift by a few points at the strongest setting, and overall $\mathcal{R}$ is close to flat.

training mix (weights err:strat:comp:soc)	$\mathcal{R}$	**error rem.**	**strategic**	**comparative**	**Socratic**
1:1:1:1 *(uniform default)*	0.836	0.962	0.880	0.890	0.611
1:1:1:2 (boost Socratic $2\times$)	0.842	0.972	0.875	0.884	0.635
1:1:1:4 (boost Socratic $4\times$)	0.841	0.924	0.864	0.904	0.670

Three findings come out of this chess composition sweep.

Modes partly substitute for one another during pooled-training. In Table 14, removing error remediation, comparative, or Socratic does not lower that mode's evaluation score, and in two of those three cases the score on the dropped mode actually rises. The most extreme case is comparative: removing it from training raises the comparative evaluation score from 0.890 to 0.934. The four guidance modes share enough underlying structure (extracting a chess move that is consistent with both the student's prior and the AI tutor's pedagogical signal) that training on three of them transfers to the fourth. Strategic is the one exception: its removal lowers overall $\mathcal{R}$, which suggests that strategic carries pedagogical signal the other three modes do not fully cover.

A model trained on one mode still answers the other three. Table 15 pushes the substitutability claim further: training the pooled-training mix on a single guidance mode and evaluating on all four still

gives $\mathcal{R} \in [0.72, 0.80]$, compared with 0.836 for the four-mode default. The transfer is asymmetric. Comparative-only training transfers best (overall $\mathcal{R} = 0.798$), consistent with comparative being the most redundant mode in the leave-one-out sweep. Error-remediation-only training transfers worst (overall $\mathcal{R} = 0.720$). This mode delivers a full diagnosis of the mistake together with the correction and its rationale, and naming the move comes with that direct-correction role; a model trained on it alone can settle for copying the named move, which does not carry over to the other three modes, where the tutor does not state which move to play. The Socratic per-mode score also stays low (≤ 0.56) in every single-mode row, so Socratic capability is the hardest piece to acquire by transfer alone and benefits most from being explicitly represented in the training mix.

Per-mode capability is controllable through training-data weights. In Table 16, scaling the Socratic weight scales the Socratic per-mode score in a graded way, with the other three modes shifting by a few points at the strongest setting. The per-mode weighting handle is useful when a downstream experiment needs a different per-mode profile: a simulator that is especially responsive to Socratic guidance, or that under-responds to it to model a student who ignores reasoning prompts, can be produced by adjusting the pooled-training mix rather than by adding new training procedures or new objectives. The reference simulators reported in the main paper use the uniform 1:1:1:1 default; the weight-scaling result is recorded here only to document that the per-mode profile can be adjusted predictably.

E Training and Experimental Details

This appendix records the hyperparameters and infrastructure that produced every checkpoint and metric reported in Section 5.2, Section 5.3, and Section 6. Data sources, training-instance formats, and corpus statistics are in Appendix B; metric definitions are in Appendix C; sweep results that motivated the recipe are in Appendix D. A reader who reproduces the source corpora and follows the values in the tables below should be able to retrain the full reference family and rerun every evaluation reported in the main paper.

The base model in all three domains is Qwen3-4B-Instruct, specifically the `Qwen3-4B-Instruct-2507` release (Yang et al., 2025). Training is in `bf16` on a single 8× A100 (40 GB) node using ms-swift (Zhao et al., 2025). Both stages train LoRA adapters (Hu et al., 2021); Stage 1 trains a fresh LoRA on the pooled-training corpus, and Stage 2 loads the Stage-1 adapter and continues training that adapter on one student's records with a smaller learning rate. Stage 2 therefore inherits the Stage-1 LoRA architecture (rank, α, dropout, target modules) in all three domains. The single random seed is the ms-swift trainer default (`42`) for optimizer state and dataloader shuffling, with a separate seed (`65`) for the data sampler that constructs the per-domain training-instance pool.

Two-stage training and shared design. The two-stage design separates a Stage-1 *pooled training* phase that pools records across a large student set from a Stage-2 *specialization* phase that adapts the resulting LoRA adapter to one student's records. The two stages share the base model (Qwen3-4B-Instruct), LoRA rank, α, dropout, target modules, optimizer (AdamW with $\beta = (0.9, 0.95)$), bf16 precision with gradient checkpointing, weight decay, gradient-norm clip, max sequence length, loss masking, trainer seed, and data-sampler seed; they differ only in learning rate, schedule, warmup length, and the per-domain values of corpus size, effective batch, and optimization length. Within each stage we keep a single shared configuration across chess, L2, and math; the per-domain values that differ (Table 18 and Table 20) are dictated by per-domain dataset scale, not tuned for any single benchmark. Every deviation from the shared configuration is enumerated in Section E.2.

E.1 Stage 1 Hyperparameters

Stage 1 trains one domain-specific LoRA on the pooled-training corpus described in Appendix B, mixing single-turn and multi-turn records at a fixed multi-turn ratio of $\rho = 0.20$ in all three domains; Appendix D.3 shows that the results do not depend on this setting. Table 17 lists the settings shared across chess, second-language English writing (L2), and mathematics (math); Table 18 lists the per-domain corpus size and resulting optimization length.

Table 17: Shared pooled-training-stage hyperparameters across chess, L2, and math.

Setting	Value
Base model	Qwen3-4B-Instruct-2507 (Yang et al., 2025)
Training framework	ms-swift (Zhao et al., 2025)
Precision	bf16 with gradient checkpointing
Hardware	1 node, 8× A100 (40 GB)
Adapter type	LoRA (Hu et al., 2021)
LoRA rank r	128
LoRA α	256
LoRA dropout	0.05
Target modules	all linear projections (q, k, v, o, gate, up, down)
Optimizer	AdamW
Learning rate	1×10^{-4}
Schedule	cosine, 100 warmup steps
Weight decay	0.1
Gradient-norm clip	1.0
Per-device batch	4
Gradient accumulation	8
Effective batch	256
Max sequence length	4096
Epochs	1.0 (L2 uses 3.0)
Multi-turn ratio ρ	0.20
Loss masking	ignore tokens inside empty `<think></think>`
Corpus-construction sampler seed	65

Table 18: Per-domain pooled-training-stage corpus size and realized optimization length. Corpus sizes match Table 1 in Section 5.1; the multi-turn fraction within each corpus is the shared $\rho = 0.20$ from Table 17. Wall-clock times are rough estimates.

Domain	training instances	total grad steps	wall-clock time
chess	100,000	391	$\sim$75 min
L2	7,800	93	$\sim$25 min
math	23,400	91	$\sim$30 min

In all runs, LoRA is applied to all linear projections (`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`). Loss masking ignores tokens inside empty `<think></think>` spans. The chess and math Stage-1 grad-step counts in Table 18 are induced directly by the shared single-epoch setting; L2 Stage 1 is the one deviation: its pooled corpus is small enough that a single epoch gives too few optimizer steps, so it trains for 3 epochs, which is why its row differs.

E.2 Stage 2 Hyperparameters

In the per-student specialization stage, each run loads the Stage-1 adapter via the ms-swift `--adapters` interface and continues training the same LoRA on one student's records. The LoRA architecture, target modules, precision, max sequence length, loss masking, weight decay, gradient-norm clip, and seeds are inherited from Stage 1. Table 19 lists the shared Stage-2 hyperparameters; Table 20 lists the per-domain values that remain specific to each domain.

Table 19: Shared per-student-specialization-stage hyperparameters.

Setting	Value
Hardware	1 node, 8× A100 (40 GB)
Initialization	load Stage-1 adapter via ms-swift `--adapters` (no re-init)
Optimizer	AdamW (`adamw_torch_fused`), fresh state
Learning rate	5×10^{-5}
Schedule	cosine, 30 warmup steps

Table 20: Per-domain per-student-specialization-stage settings. "Per-student instances" is the training-set size used for one student's specialization run; for a given domain this size is constant across the per-student adapters in our reference family. "Total steps" is the number of optimizer steps in one student's specialization run. The chess and math effective batch of 256 is realized as per-device batch $4\times$ gradient accumulation 8×8 GPUs; the L2 effective batch of 8 is per-device batch $1\times$ no accumulation $\times 8$ GPUs.

Domain	students	per-student instances	Stage-2 ρ	epochs	effective batch	total steps
chess	30	1,000	0.20	3.0	256	12
L2	15	73	0.20	3.0	8	~ 30
math	15	153	0.20	3.0	256	3

The Stage-2 learning rate is half of Stage 1 and the warmup is shorter (30 vs. 100) because Stage 2 starts from the Stage-1 adapter rather than from the base model, and per-student datasets are about two orders of magnitude smaller than the pooled corpus in all three domains. Chess and math keep the Stage-1 effective batch (256); L2 uses an effective batch of 8 (per-device batch 1, no gradient accumulation, 8 GPUs), which at 73 per-student records and 3 epochs yields approximately 30 optimizer steps, sufficient to specialize the warm-started Stage-1 LoRA to each learner's error profile. L2 Stage 2 mixes single-turn essay records and multi-turn correction records at $\rho = 0.2$ (15 multi-turn plus 58 single-turn per student), so the per-student adapter continues to receive multi-turn signal beyond what Stage 1 supplied; the L2 guidance-responsiveness numbers in Section 5.3 reflect both stages' multi-turn training.

The saved Stage-2 adapter for every per-student simulator carries four LoRA fields ($r = 128$, $\alpha = 256$, dropout $= 0.05$, target modules $=$ all linear projections) matching Table 17, confirming inheritance from Stage 1 across all three domains.

Per-domain deviations and rationale. Chess Stage 1 and Stage 2 follow the shared defaults (1 epoch Stage 1, 3 epochs Stage 2, effective batch 256, decoding budget 32 tokens, repetition penalty 1.0). L2 and math each deviate in a small number of places summarized below; each deviation is forced by the per-domain dataset scale.

- **L2 Stage 1 epochs: 3 (default 1).** At the L2 pooled-corpus size of 7,800 instances and effective batch 256, one epoch yields only 31 optimizer steps. Three epochs bring the total to 93 steps, comparable in magnitude to chess (391) and math (91) at a single epoch.
- **L2 Stage 2 effective batch: 8 (default 256),** realized as per-device batch $1\times$ no gradient accumulation $\times 8$ GPUs. With only 73 records per learner, batch 256 does not produce even one full optimizer step per epoch; batch 8 gives approximately 30 steps over the 3-epoch run.
- **L2 decoding budget: `max_new_tokens` 256 (default 32), `repetition_penalty` 1.1 (default 1.0).** L2 outputs are full essays of roughly 50 to 150 words; the larger token budget accommodates the longer outputs, and a small repetition penalty prevents greedy-decoding loops past the natural essay end.
- **Math Stage 2 optimization length: 3 optimizer steps total** (chess 12, L2 $\approx$ 30). At 153 per-student instances (Table 20), the shared effective batch 256 over 3 epochs yields exactly 3 steps, the maximum the data supports without departing from the shared batch size. Because each student contributes few records, the Stage-2 optimization horizon is short in all three domains, and within this regime the trainer seed has little effect on run-to-run variance. For example, in math the observed std across runs (Table 22) is dominated by the data-sampler seed.

E.3 Evaluation Protocol

Decoding for both behavioral fidelity (single-turn, on S_i) and guidance responsiveness (multi-turn, on T_i) is greedy ($T = 0$) under the per-domain budgets in Table 21. The chat-template flag `enable_thinking` is pinned to `False` at decode time. Without that flag, Qwen3's chat template defaults to opening a `<think>` block that is only closed if the model emits the matched closing tag within `max_new_tokens`; under the short-answer budgets used for chess and math the model otherwise stays inside the reasoning block and never produces the answer token. The flag is set identically at training time (Stage 1, Stage 2, and the tutor RL rollouts in Appendix E.5) so

train and inference agree. The same decoding budgets are applied to the trained simulators and to the prompted closed-source baselines unless a metric requires token log-probabilities rather than free-form decoding.

Table 21: Per-domain decoding settings at evaluation. All decodes are greedy ($T = 0$), and the token budget applies to fidelity and guidance evaluation alike.

Domain	`max_new_tokens`	`repetition_penalty`
chess	32	1.0
L2	256	1.1
math	32	1.0

The L2 `max_new_tokens` budget is larger because L2 outputs are full essays of roughly 50 to 150 words; chess outputs are a single move and math outputs a short numerical answer. The L2 repetition penalty of 1.1 applies when decoding an essay, where pure-greedy decoding occasionally entered repeat loops past the natural essay end; the penalty removes that decoding artifact without changing the per-token error profile that L2 fidelity scores.

Held-out sets. Each per-student evaluation uses the frozen STUDENTSIMEVAL held-out sets (Appendix B.1), which follow the (S_i, T_i) schema from Section 3. The single-turn set S_i is sampled from the same data stream as training but on disjoint records (different game positions in chess, different essays in L2, different problems in math; per-domain split protocols in Appendix B). The multi-turn set T_i is similarly disjoint. For the headline numbers in Section 5.2 and Section 5.3, $|S_i|$ is 5,000 positions per chess player, 26 essays per L2 student, and 66 problems per math student (uniform); $|T_i|$ is 4,000 multi-turn records per chess player (1,000 per guidance mode), 40 multi-turn records per L2 student (20 per guidance mode), and jagged across math students with a per-student range of 21 to 99 (mean 59, evenly split across the three math guidance modes within each student). Per-domain $|S_i|$ and $|T_i|$ counts are tabulated in Appendix B.

Closed-source baseline scoring. GPT-4o and GPT-5.4 are queried through the Azure OpenAI API with greedy decoding ($T = 0$) and the same per-token budgets as Table 21. For math fidelity and math guidance responsiveness, which are top-1 letter accuracy on a four-way multiple-choice form (Appendix C.5, Appendix C.6), the baselines decode a single letter token and are scored by exact letter match.

Random seeds and number of runs. Headline numbers in Section 5.2 and Section 5.3 are the cross-seed mean over $S = 3$ independent training runs per domain that vary both the trainer seed and the data-sampler seed; each run retrains the full pipeline (Stage 1 + Stage 2) from scratch and re-evaluates on the same held-out set. The run-to-run standard deviation across these seeds is reported in Table 22 (Appendix E.4). Decoding is greedy with $T = 0$ throughout.

E.4 RUN-TO-RUN STANDARD DEVIATION

Table 2 and Table 3 report population means only. Table 22 reports the run-to-run standard deviation of the population means across multiple training runs with different random seeds (trainer seed and data-sampler seed both varied). This measures method-level variance: how much the headline number moves if we retrain from scratch with a different seed. The closed-source baselines and the naive Qwen3-4B-Instruct baseline are not trained, so there are no training seeds to vary and this retrain-from-scratch variance does not apply to them; the run-to-run column is therefore omitted for them, and only STUDENTSIM carries a run-to-run std.

E.5 TUTOR RL: SFT BASE, GRPO SETUP, AND EVALUATION

Section 6 reports a chess proof of concept that compares three tutor-RL rewards: the supervised-finetuned tutor with no RL, a frontier LLM (GPT-5.4) prompted as the student simulator, and our trained STUDENTSIM with extensible heads. The two RL conditions start from the same SFT checkpoint and continue with GRPO (Shao et al., 2024). The implementation is verl (Sheng et al., 2025) on a single 8× A100 (40 GB) node, over a chess RL playground built from held-out

Table 22: **Run-to-run standard deviations for StudentSim.** Each cell is the StudentSim population mean for that domain and metric, with the across-seed standard deviation appended as a subscript. Computed from S independent training runs with different trainer seeds and data-sampler seeds; each run retrains the full pipeline (Stage 1 + Stage 2) from scratch and re-evaluates on the same held-out set. The naive Qwen3-4B-Instruct baseline and the closed-source baselines (GPT-4o, GPT-5.4) are not trained, so there are no training seeds to vary and this table's retrain-from-scratch std does not apply to them; they are omitted.

Domain	S (# seeds)	StudentSim $\mathcal{F}$	StudentSim $\mathcal{R}$
chess (30 players)	3	$\mathbf{0.5150}_{\pm 0.0009}$	$\mathbf{0.9067}_{\pm 0.0009}$
L2 (15 learners)	3	$\mathbf{0.5624}_{\pm 0.0046}$	$\mathbf{0.6417}_{\pm 0.0233}$
math (15 students)	3	$\mathbf{0.6384}_{\pm 0.0044}$	$\mathbf{0.9181}_{\pm 0.0099}$

positions for the 30 chess players (Appendix B). The conditions share the SFT base and every GRPO hyperparameter and differ only in the reward model queried per rollout. The two RL conditions are compared at their final checkpoint, after the 20 GRPO steps in Table 24. This appendix documents the SFT base (Table 23), the GRPO settings (Table 24), the reward definitions (Table 25), and the expert human evaluation (Table 26).

SFT base (the no-RL condition, and the initialization for the two RL conditions). The tutor is Qwen3-VL-8B-Instruct supervised-finetuned with LoRA via ms-swift on rendered chess positions paired with reference tutor guidance; hyperparameters are in Table 23. The SFT data recipe combines four ingredients that together cut factual errors on this multimodal task: *i)* the board image is rendered with the side-to-move at the bottom, *ii)* the position is grounded piece-by-square in the prompt text, *iii)* reference-answer filters drop instances whose guidance contains an illegal (hallucinated) move, makes a piece-on-square claim that contradicts the board, or quotes engine metadata such as a numeric evaluation, and *iv)* the prompt carries a color hint and a generic system message. We select the step-704 checkpoint, merge the LoRA adapter into the base weights, and use the merged checkpoint directly as the no-RL condition and as the policy initialization for the two RL conditions.

Table 23: SFT hyperparameters for the tutor base (the no-RL condition, and the initialization for the two RL conditions).

Setting	Value
Base model	Qwen3-VL-8B-Instruct (Bai et al., 2025)
Framework	ms-swift, LoRA
LoRA rank / α	32 / 64 (all linear projections)
Optimizer / LR	AdamW / 1×10^{-4}
Epochs	1
Effective batch	256
Max sequence length	2048
Max image pixels	448×448
Selected checkpoint	step 704, LoRA merged into base

GRPO settings. The two RL conditions use the identical GRPO configuration in Table 24 and differ only in the reward model.

Reward definitions. The three conditions differ only in how a rollout is scored (Table 25). Both RL rewards build on a shared move-quality term: given the student's wrong move and the simulator's revised move after reading the tutor's guidance, the centipawn difference is capped at ± 1500 to keep mate-changing moves from saturating the signal, then squashed through a $\tanh$ with a 500-centipawn shoulder so that typical move improvements live in the linear regime; an illegal or missing move scores -1.0. Centipawn values come from a precomputed Stockfish depth-15 lookup over the playground positions. The conditions differ in who produces the revised move and in what shapes it: our StudentSim reward uses our trained StudentSim as the simulator and multiplies the move-quality base by a style gate and a perception gate, both read off the frozen simulator backbone, whereas the GPT-5.4-simulator reward uses GPT-5.4 prompted as the student.

Table 24: Shared GRPO hyperparameters for the chess tutor proof of concept; the two RL conditions use these values and differ only in the reward model (Table 25).

Setting	Value
RL framework	verl (Sheng et al., 2025)
Advantage estimator	GRPO (critic-free; no value model)
Group size (rollouts per prompt)	4
KL	`use_kl_loss` True, `kl_loss_type` low-variance, coefficient 0.01; `use_kl_in_reward` False
Entropy coefficient	0
Actor opt / LR	AdamW / 1×10^{-6}
Train batch / mini-batch	112 / 112
Sharding	FSDP2 with parameter and optimizer offload
Rollout engine	vLLM, max model length 2048
Training length	20 steps

Table 25: Reward definitions for the three conditions. The move-quality term (cp = Stockfish centipawns) is shared by the two RL rewards. Our STUDENTSIM reward multiplies the move-quality base by a style gate and a perception gate; the 2.0 and 3.0 are the gates' α decay rates (defined below the table).

Condition	Reward
No RL	none (SFT base, no reinforcement learning)
GPT-5.4 simulator	with GPT-5.4 prompted as the student to produce the revised move; one frontier-model API call per rollout
Ours	move-quality (our STUDENTSIM simulator) $\times$ style gate $\times$ perception gate, one scalar per rollout
move-quality term	$\tanh\big(\mathrm{clip}(q_{\text{post}} - q_{\text{wrong}}, \pm 1500)/500\big)$, units in cp
illegal / missing move	-1.0
quality lookup	precomputed Stockfish table at depth 15 on playground positions
STUDENTSIM simulator	chess Stage-1 LoRA on Qwen3-4B-Instruct-2507 (Table 17), served locally
style gate	$\exp\big(-2.0\,(1 - p_{\text{pref}})\big)$; p_{pref} is the style head's probability of the preferred style
perception gate	$\exp\big(-3.0 \sum_i w_i p_i\big)$; p_i are the six board-error probabilities, w_i their per-class F1 scores normalized to sum to one

Reward heads. The style and perception heads are two linear probes on the same frozen chess Stage-1 STUDENTSIM simulator that produces the student's revised move: only these two heads are trained, while the simulator's weights and its move prediction are inherited unchanged from Stage 1, so the entire reward is computed from one on-node model. Both heads read the pooled hidden states over the tutor's explanation. The style head classifies the explanation into one of four teaching styles (error remediation, Socratic, strategic, comparative), distilled from a TF-IDF logistic-regression style classifier; its gate rewards adherence to the run's preferred style, set to Socratic in the reported runs to match the Socratic-leaning student used in the human-study personalization phase. The perception head produces six independent probabilities for board-grounding errors in the explanation (wrong square, wrong piece, wrong color, hallucinated piece, wrong capture, illegal move), trained against a `python-chess` rule verifier and GPT-5.4 labels; its gate penalizes explanations that misdescribe the position, weighting the six error types by their per-class F1. Both gates lie in $(0, 1]$ and multiply the move-quality base (Table 25), so an on-style, board-accurate explanation leaves the base intact while an off-style or board-misdescribing one discounts it.

GPU layout and serving. GRPO uses 7 trainer GPUs (FSDP2 with parameter and optimizer offload) and 1 GPU that hosts the reward model. For our STUDENTSIM reward, that GPU serves the local STUDENTSIM simulator together with the style and perception heads, so the entire reward is computed on-node with no external dependency. For the GPT-5.4-simulator reward, the reward instead issues a GPT-5.4 API call on every rollout to obtain the simulated student's revised move; this frontier-model dependency, absent from our reward, is the practical cost of the GPT-5.4-simulator reward at training time. The no-RL baseline runs no RL and needs no reward GPU.

Expert Human Study. The evaluators are competitive chess players screened by an internal rating assessment; their ratings run up to 2200, and the three rated 2000 or above form an expert subset that we report separately (Table 26). They rate each tutor's response, blind to condition and presentation order, in three phases. Phase 1 (accuracy): annotators mark each sentence of the tutor's guidance for factual correctness, and accuracy is the fraction of responses with no sentence flagged as a severe, actively misleading error. Phase 2 (teaching quality) is a 1–5 rating of guidance quality. Phase 3 (personalization) is a 1–5 rating in the context of a Socratic-leaning student. The study collected 74 finalized annotations from these evaluators; Table 26 reports all annotations, a cleaner triply-annotated (k=3) subset, and the 2000+ expert subset.

Table 26: **Human evaluation of the AI chess tutors.** *Accuracy* is the percentage of responses with no severe (actively misleading) factual error; *Guidance* and *Personalization* are 1–5 ratings. "Global" is all annotations; "k=3" is the triply-annotated subset; "Experts $\geq$ 2000" restricts to the three evaluators rated 2000 or above. Each annotation rates every condition, so n is the number of annotations behind each row of a view: 74 annotations over 30 positions from 8 evaluators (Global), 48 over the 16 positions that three evaluators each annotated (k=3), and 30 from the three highest-rated evaluators (Experts $\geq$ 2000). Our STUDENTSIM reward ranks first on all three axes in all three views.

View	Condition	Accuracy (%)	Guidance (1–5, ↑)	Personalization (1–5, ↑)
Global (n=74)	No RL	75.7	2.99	2.80
	GPT-5.4 reward	71.6	3.08	2.42
	Ours	**90.5**	**3.31**	**3.93**
k=3 (n=48)	No RL	75.0	3.10	2.65
	GPT-5.4 reward	70.8	3.21	2.33
	Ours	**91.7**	**3.44**	**4.00**
Experts $\geq$ 2000 (n=30)	No RL	70.0	3.03	2.83
	GPT-5.4 reward	56.7	2.97	2.37
	Ours	**93.3**	**3.17**	**3.70**

E.6 COMPUTE RESOURCES

All training was performed on a single 8× A100-40GB node. The runs reported in the main paper consumed approximately 390 A100-40GB GPU-hours in total: roughly 150 for Stage-1 pooled training across the three domains, 20 for Stage-2 per-student specialization, 170 for the chess tutor SFT and the GRPO runs behind the proof of concept, and 50 for evaluation and decoding (including prompted-baseline inference for GPT-4o and GPT-5.4). The full project, including ablations on subsets of students within a single domain (the modality VL sweep and the reward-variant comparisons) and preliminary or failed runs, used approximately 1,000 A100-40GB GPU-hours. All GPU-hour figures here are rough estimates from the run wall-clocks rather than exact accounting.

F SCOPE AND DESIGN RATIONALE

This appendix records the design rationale behind several scope and operationalization choices in the framework: what behavioral fidelity and guidance responsiveness measure, how the multi-turn corpora are constructed, why the per-domain $\mathcal{F}$ instantiations differ, how per-student data scale interacts with the two-stage pipeline, which baselines we evaluate against in each domain, and what the tutor RL proof of concept claims. Each subsection states the design choice as a thesis, gives the constraint or principle that motivates it, and points to the section in the main paper where the choice is exercised.

F.1 WHAT GUIDANCE RESPONSIVENESS MEASURES

Educational measurement has long treated assisted performance as informative about the learner in its own right. Vygotsky's zone of proximal development locates achievement under guidance alongside independent achievement (VYGOTSKY, 1980), and dynamic assessment operationalizes that idea by evaluating how a learner responds after mediation (Grigorenko & Sternberg, 1998). Our

multi-turn record (x, m, τ, m^*) has exactly this structure: a problem x, the learner's initial response m, a tutor's mediating turn τ, and the criterion response m^* that the guidance is steering toward. In this framework, behavioral fidelity $\mathcal{F}$ captures the learner's recorded independent starting point, while guidance responsiveness $\mathcal{R}$ captures whether the simulator reaches the criterion response under guidance. Prior student simulators reproduce isolated responses; carrying an instructional exchange forward through answer, guidance, and revision is the capability an interactive tutor loop depends on.

This grounding also fixes the target of $\mathcal{R}$. Dynamic assessment scores post-mediation performance against a criterion response (Grigorenko & Sternberg, 1998), because the construct of interest is whether the learner can move in the direction opened by the help. We therefore score whether the simulator emits the canonical resolved response m^* on each record (x, m, τ, m^*) after reading the tutor guidance τ. The target m^* is well-defined per benchmark: the engine best move in chess, the correct option in mathematics, and the canonical written correction in second-language English writing. Section 3 gives the formal definition.

This criterion-based target is also consistent with theories of contingent tutoring and scaffolding, where effective support is adapted to the learner and then withdrawn as the learner takes over (Wood et al., 1976), and with the assistance dilemma, which holds that the appropriate form of help depends on how much the learner can productively do with support (Koedinger & Aleven, 2007). In our criterion-based setting, responsiveness concerns whether the simulated response advances to the target of the intervention, and a per-student continuation target would entangle $\mathcal{R}$ with the student-specific response patterns already measured by $\mathcal{F}$. Collecting such continuations for each intervention at benchmark scale is also a data-collection cost few settings can absorb. Substituting real per-student continuations in future datasets remains compatible with the $\mathcal{F}$/$\mathcal{R}$ decomposition.

This is also exactly the signal the tutor optimization loop needs. In STUDENTSIM, the simulator serves as the reward model in tutor RL (Section 6), so the tutor must receive feedback about whether its guidance moved the simulated response toward the response that guidance was intended to elicit. $\mathcal{R}$ measures the rate at which the simulator supplies that feedback correctly. A simulator with high $\mathcal{R}$ produces a reward whose sign and magnitude track the direction of tutor intent, enabling policy optimization; a simulator with low $\mathcal{R}$ breaks that link, so optimization cannot recover the missing instructional signal.

Together, $\mathcal{F}$ and $\mathcal{R}$ supply the two properties a tutor RL loop needs from a simulator: the interaction begins from the actual error this student would make ($\mathcal{F}$) and then tests whether the simulator moves to the canonical response targeted by guidance ($\mathcal{R}$). The fuller longitudinal dynamics a learner exhibits across many interactions extend beyond these two axes and remain the horizon this decomposition opens toward (Section 7). Section 6 shows that simulators strong on both axes provide a stronger tutor-RL reward than a frontier-LLM simulator or no RL on held-out chess positions, which is the operational test of this measurement choice.

F.2 MULTI-TURN CORPUS CONSTRUCTION AND THE SIMULATOR-TUTOR DECOUPLING

Multi-turn (x, m, τ, m^*) tuples do not exist at scale in two of our three domains, so constructing them with a parameterized LLM tutor is the principled way to instantiate the (S_i, T_i) schema of Section 3 and to control the guidance-style distribution at training time. The Lichess export (Lichess, 2025) pairs each board position with the player's move and an engine evaluation, but carries no tutor commentary attached to the player's error. FoundationalASSIST (Worden et al., 2026) pairs each problem with a student answer and a correctness flag, but carries no per-error tutor explanation. A corpus that exposes problem, error, and canonical correction does not, by itself, expose the natural-language guidance turn τ that the schema needs.

L2 uses real-teacher annotations directly. EFCAMDAT (Geertzen et al., 2014; Shatz, 2020; Öksüz et al., 2026) contains span-level corrections written by real teachers, and we use those annotations as the canonical m^* on every L2 multi-turn record. The natural-language tutor turn τ is rendered from each annotation through one of two style templates, point-based and rule-based, so the surface phrasing is consistent while the underlying span and correction stay human-authored. The framework instantiates against whatever guidance source each domain exposes: real-teacher annotations on L2, controllable LLM authorship on chess and math.

For chess and math, τ is generated by an LLM tutor conditioned on the position or problem, the student error, and the canonical correction, under a fixed style template per record. This authorship is a control mechanism. The templates parameterize four guidance styles in chess (error remediation, comparative, strategic, Socratic) and three in math (error remediation, Socratic, conceptual explanation), mixed uniformly at training time. The style set spans the range of instructional support a tutor chooses among, from stating the correction outright to prompting the student to derive it, the range over which effective tutoring adjusts help to the learner (Wood et al., 1976) and along which the assistance dilemma is posed (Koedinger & Aleven, 2007). That control is what enables the per-mode $\mathcal{R}$ breakdown in Section 5.3 and the held-out Socratic case study, in which the simulator solves a position whose Socratic prompt it never saw at training time. A fixed corpus of opportunistic tutor traces would not afford a balanced per-style population.

Guidance Quality. For every record (x, m, τ, m^*), the target m^* is fixed by verified supervision before any tutor turn is written, and τ is generated from the analysis that establishes that target. In chess, m^* is the Stockfish depth-15 best move, and the generator is given the engine's full position analysis: the ranked candidate moves with their centipawn evaluations, and the principal continuation of each candidate. The continuations are what let the guidance explain how the game unfolds after each choice. In math, m^* is the platform answer key after an audit (Appendix B), and the generator receives that audited key together with the student's recorded answer. In L2 writing, m^* is the human teacher's span correction, and τ is templated directly from that annotation with no LLM generation step. Across all three domains, the LLM determines only the wording of the guidance; the endpoint it steers toward is set in advance by an engine or a human annotation and is the same supervision used to compute $\mathcal{R}$.

The generator, the simulator, and the prompted closed-source baseline are three separate systems. The LLM that authors τ during corpus construction is GPT-4o or GPT-5.1 in chess, selected per record by whether the student's move fell outside the engine's top-k list (contrasting challenging positions and easy ones), and GPT-5.4 in math. The simulator is Qwen3-4B-Instruct (Yang et al., 2025) with LoRA adapters; it never produces τ and only consumes τ to emit m^*. At evaluation the closed-source baseline is GPT-5.4, prompted under the same τ that the simulator reads, so on math the baseline shares a model family with the generator, an arrangement that works in the baseline's favor. Our simulator never authors τ at any point; it only consumes it.

F.3 Per-Domain Operationalization of Behavioral Fidelity

$\mathcal{F}$ is a generic concept: how closely the simulator's response on a held-out problem matches student π_i's recorded response. The operational form is per-domain because the response space is per-domain. Chess $\mathcal{F}_i$ is top-1 accuracy against the recorded move. L2 $\mathcal{F}_i$ is per-record error-density and issue-type-profile match across seven LanguageTool categories between the simulator's generated essay and the recorded essay. Math $\mathcal{F}_i$ is top-1 accuracy on a four-way multiple-choice form: the target letter holds the student's recorded answer (correct or wrong) and the three distractor letters hold the most common other-student answers on the same problem. Appendix C gives the full formulas.

Each operationalization is the well-posed match metric on its response space. Chess legal-move sets are finite (typically about thirty moves) and the recorded supervision is one chosen move per position; top-1 against that move is the only fidelity signal one observation can support. L2 essays are free-form text, and no two essays the same learner writes are identical at the surface; what survives as a per-learner signal is the rate and type-profile of mistakes the learner still makes, which is what per-record error-profile match captures. Math student answers are open-ended free-text strings, and raw-text likelihoods on them are incomparable across simulators because they depend on tokenization of the numerical answer; casting each problem into a four-way form with empirical other-student distractors turns the response space into a categorical one, on which top-1 accuracy is the natural fidelity score.

Cross-domain absolute comparison of $\mathcal{F}$ values is not the framework's unit of analysis. What we report per domain is the relative improvement of StudentSim over the strongest available baseline (Table 2, Section 5.2), which is comparable across domains because every baseline in a domain is scored on the same per-domain $\mathcal{F}$. This is the convention that supports BLEU on machine translation alongside ROUGE on summarization: both rank systems within a benchmark, and neither gains from being coerced into the other's units.

A unified $\mathcal{F}$ is not the right design. Coercing all responses into a shared embedding space loses the domain-specific signal: chess top-1 uses a discrete legal-move list, L2 profiles a fixed seven-category taxonomy, math letter accuracy a four-way categorical decision. Reporting only relative improvement over a population prior discards the absolute scale that makes per-student claims interpretable within a domain. Reporting per-domain $\mathcal{F}$ and per-domain relative improvement together keeps both an absolute interpretation within each domain and a clean comparison across baselines.

F.4 Per-Student Data Scale and the Population Prior

Per-student data is bounded by the number of interactions any one learner actually generates, and obtaining more of them takes as long as that learner takes to produce them, which is why per-user sparsity recurs in every system that learns from interactions with real users (Schein et al., 2002; Lee et al., 2019; Bhattacharjee & Wayllace, 2025; Li et al., 2017). Training a simulator end to end on one student's records would overfit the handful that exist and would learn the domain's shared regularities over again for every student (Dodge et al., 2020; Finn et al., 2017). The framework instead uses the data that exists at two levels of granularity: pooled domain records for what students share, and individual records to adapt that base to a particular learner.

The two-stage pipeline in Section 4 turns that constraint into a training recipe. Stage 1 pools records across students within a domain, producing a population prior over the response space, the mistakes students commonly make, and the ways students revise after guidance. Stage 2 continues from that base on one student's own records, so each additional personal instance gives the simulator more evidence about that learner's characteristic behavior. The design therefore scales with whatever evidence a platform holds: the pooled stage supplies broad competence from the full domain, as supported by the matched chess ablation in Appendix D.1, and the specialization stage sharpens the per-student fit as personal data accrues. A deployment needs only those two ingredients, and for online learning platforms that want to train student simulators to help each student grow can supply both, since a large population generates the pooled corpus while each learner contributes a handful of records.

A learner with no recorded interactions occupies the pooled-prior end of this range. The simulator is then the Stage-1 domain model. Once interactions are observed, Stage 2 specializes that prior toward the individual learner, so the framework needs no separate cold-start mechanism.

F.5 Domain-Specific Baselines and the Strongest Available Reference

For each domain, we evaluate references that isolate the capabilities integrated by StudentSim. Chess has both skill-conditioned and per-player human move-prediction research; L2 and math do not have published domain models that generate the same response forms as the target students.

Individual Maia specializes a pretrained Maia model to a particular player and shows that player-specific fine-tuning can improve move prediction when sufficient history is available (McIlroy-Young et al., 2022). The reported gains begin at 5,000 complete games per player. Collecting that volume for every learner is unrealistic in the educational setting we target, especially for a broad population that includes beginners. Individual Maia also predicts moves from chess records without an input pathway for natural-language tutor guidance. Our available per-player data therefore does not support its demonstrated training regime. We therefore use Maia2 (Tang et al., 2024), the latest population-level human move-prediction model available at the time of our experiments, as the chess baseline.

For L2 and math, no published model plays the analogous role of Maia2. Knowledge tracing exists for math problem traces in principle, and earlier work has applied BKT (Corbett & Anderson, 1995), DKT (Piech et al., 2015), and AKT (Ghosh et al., 2020) to predict per-step correctness; however, those models predict binary correctness on the next problem rather than which specific (often wrong) free-text answer a particular student would emit, which is the fidelity target our framework operationalizes. They are therefore not a drop-in baseline for our $\mathcal{F}$. For L2 essay generation conditioned on per-learner history, no published artifact exists at all. In both domains, the closed-source LLM with the student profile in context is the strongest available reference, and that is what we evaluate against in Section 5.2 and Section 5.3.

Component references and integrated simulation. The baseline set is designed to locate the two capabilities that STUDENTSIM integrates. Maia2 represents structured behavioral grounding in chess: it maps a board position and player skill level to a human move distribution, but has no interface for natural language tutor guidance. GPT-4o and GPT-5.4 represent language based interaction: they can condition on a student profile, problem context, and tutor guidance, but do not contain a separately trained model of the student's domain behavior. STUDENTSIM combines these functions by training a language model to realize a structured student state and to update the resulting behavior after instruction. In chess, Maia2 supplies one source of evidence within that state; in second language writing and mathematics, the state is constructed from the corresponding learner history and error or skill statistics. The three systems therefore form a component-to-system evaluation: behavioral grounding, language based interaction, and their trained integration.

For the prompted language-model references, the input contains the historical student profile, the current problem, and, where applicable, the tutor guidance turn. The chess prompt includes the player statistics, FEN representation, and color to move; Maia2 is evaluated separately as the domain behavioral reference. A short system instruction and move-request suffix require a bare UCI move so that the output is parseable under the same top-1 scoring protocol.

The framework itself is benchmark-agnostic. The (S_i, T_i) schema and the $\mathcal{F}$ and $\mathcal{R}$ metrics do not depend on which baselines are present in the comparison set. Any future per-domain student-behavior model published in the L2 or math literature can be added to Table 2 and Table 3 without modifying the metrics or the held-out evaluation protocol. The framework is therefore designed to absorb stronger baselines as they become available.

F.6 VALIDATION SCOPE OF THE TUTOR RL PROOF OF CONCEPT

The Section 6 comparison is constructed so that the reported tutor quality is evaluated independently of the GRPO reward used during training. We assess the trained tutors through an expert human study in which competitive chess players rate blind, order-shuffled tutor responses on accuracy, guidance quality, and personalization. Evaluators do not see which system produced a response, and they never see the reward definition and the student simulator used during tutor training. This addresses the system-level question of interest: whether an integrated reward system built around a trained, extensible STUDENTSIM backbone produces a better tutor than the practically available frontier-LLM-as-student-simulator reward and than no reinforcement learning.

The study is reported in three views: all annotations, the triply-annotated subset, and the subset of evaluators rated 2000 or above. These views test whether the ordering holds under stricter annotation agreement and stronger evaluator expertise, and all of them rest on judgments that are decoupled from the optimization signal that trained the tutors.

The proof of concept is scoped to simulator-informed AI tutor optimization, where the question is whether feedback from a trained student simulator is informative enough to improve tutor responses under a controlled comparison. Section 6 answers that by training tutors with different reward sources and evaluating their outputs with independent expert judgments. Real-student learning outcomes are the target of a deployment study, whose endpoint is student improvement over time under live use, and the simulator-feedback result established here is what motivates that next stage.

Why chess is the right scope for the proof of concept. Chess supplies a precise per-position notion of move quality through Stockfish centipawn evaluation at fixed depth, a signal that is independent of the simulator and of the tutor's wording and is tied directly to the game state. That makes chess a clean setting in which to run tutor RL against a well-specified reward while keeping a separate human evaluation of the final tutor outputs. Instantiating the same proof of concept in L2 writing or open-ended mathematics would first require per-domain reward functions over free-form student responses, such as reliable essay-quality scoring or robust verification of mathematical reasoning, each an open research program in its own right. Whether simulator feedback can improve a tutor is a question about the framework, and chess is the clearest domain in which to answer it under an evaluation protocol that stays independent of the training reward.

G BROADER IMPACTS

Student simulators can replace much of the slow, expensive real-student recruitment otherwise needed for tutor evaluation and improvement, providing proxy feedback at machine timescales and broadening the profile spectrum that tutor research can practically cover. A first risk is profiling misuse: per-student behavioral models could be repurposed for predictive screening, ranking, or differential treatment of students if used outside the evaluation setting they are trained for. A second risk is distributional inheritance: simulators reflect the populations present in their source corpora, namely EFCAMDAT's eleven most-represented nationalities for L2, the ASSISTments grades 6 through 8 American mathematics curriculum for math, and the Lichess online player population for chess. Tutors optimized only against this pool may underserve students outside it. To limit these risks, the benchmark is defined using paper-local pseudonymous identifiers, and the accompanying code is scoped to the published evaluation tasks rather than downstream student-facing deployment. The framework itself is corpus-agnostic; explicitly documenting the represented populations, as in Appendix B, allows targeted extension to underrepresented groups as new corpora become available.