WHO SCORES THE TUTOR? SAMPLING NOISE AND SELF-PREFERENCE IN SIMULATED-LEARNER EVALUATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Tutoring agents are compared by the learning gain they produce in a simulated learner. This learner is a knowledge-tracing model fitted to real interaction data, and the same model also scores the teaching. On one corpus, both DKT scorers rank a policy that plans against a DKT model first out of eight, but an AKT scorer ranks it seventh on the same learners and sequences. This paper isolates the effect of the scorer. Five planning policies (one per knowledge-tracing architecture) and three scripted controls choose exercises for the same learners. The chosen sequences are then frozen, and 10 scorers (five architectures with two seeds each) rank the policies on each of three corpora. Disagreement between scorers is measured against a resampling floor, the agreement of one scorer with itself on disjoint halves of the learners. This floor is already low (Kendall $\tau = 0.524$), so with *sampling noise* alone a benchmark of this size barely determines its own ranking. Changing the architecture of the scorer (*scorer identity*) lowers τ by a further 0.189 (bootstrap 95% interval $[+0.048, +0.237]$, 2,062 simulated learners from 999 students). On the two ASSISTments corpora, changing only its seed adds no disagreement beyond resampling. Scorers also show a directional bias. A scorer ranks the policy planned against its own architecture 1.57 positions higher than scorers of other architectures do (exact assignment tests, $p \leq 0.0083$). The bias is positive on all three corpora and depends on the planning. A greedy control that uses the same model without planning against it receives no detectable preference. On ASSIST2017, raising the search budget from 12 to 192 candidates nearly triples the gain that scorers of the same architecture as the planner see in excess of the other scorers. Averaging a larger panel of scorers does not produce a usable ranking on average at any panel size tested. Excluding scorers of the same architecture as the planner is free and reorders every corpus leaderboard. It changes the reported winner on NIPS34 in both runs and on ASSIST2017 in the second run. The protocol, exclusion estimator and instrument will be released.

1 INTRODUCTION

Evaluating a tutoring agent requires a learner to be taught. Recruiting one per condition does not scale to the number of comparisons a paper needs, so the field substitutes a simulator. The simulator of choice is a knowledge-tracing model fitted to recorded interaction data (Chen et al., 2023b; Wang et al., 2026; Lee et al., 2026). The arrangement is economical, but it carries an unexamined consequence. The fitted model that simulates the learner also scores the teaching.

A policy that selects exercises by consulting a model of the learner optimises against one particular parameterisation of learner behaviour. A scorer of the same architecture scores the policy according to the same assumptions. The resulting advantage remains unmeasured. Whether it is large enough to matter is an empirical question about the protocol, not about any method.

This paper measures this advantage with a purpose-built experiment. Five planning policies (one per knowledge-tracing architecture) and three scripted controls are run once per corpus. The chosen sequences are frozen and scored by 10 scorers spanning five architectures at two seeds each. Only the scorer and its sampled responses vary across leaderboards (Figure 1).

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.

With a few hundred simulated learners, resampling alone substantially reorders the leaderboard. More learners reduce this instability. Beyond that resampling floor, changing the scoring architecture reduces Kendall τ by a further 0.189.

The paper makes four contributions. The first is **a frozen-sequence protocol with a resampling floor**. Freezing the taught sequences before scoring separates the effect of the scorer from that of the teaching. The protocol measures scorer disagreement against the agreement of one scorer with itself, rather than against a random ranking. The floor is low, so reporting it is a prerequisite for interpreting every subsequent number. The second is **the measurement of scorer identity** across 2,062 simulated learners, three corpora and 10 scorers. Cross-architecture agreement falls 0.189 below the resampling floor (95% interval $[+0.048, +0.237]$). On both ASSISTments corpora, two seeds of one architecture agree as well as resampling allows, which locates the disagreement in the architecture. The third is **self-preference and its dose response**. Scorers also favour policies planned against their own architecture. In rank positions, this bias is undetected for a scripted control that consults the same model without planning against it. A larger search budget enlarges the gain that own-architecture scorers see in excess of the others. The fourth is **a free exclusion estimator**, evaluated against the intuitive alternative. Enlarging the scorer panel does not work on average at any size tested. Excluding scorers that share an architecture with the policy under test costs nothing, reorders the leaderboard of every corpus and changes two reported winners.

Two features separate this setting from self-preference in language-model evaluation. First, the scorer generates none of the teaching it scores. It simulates a learner, and the outcome is a probe measurement, so the effect cannot come from a judge recognising its own prose. Second, the quantity that drives the effect is under experimental control. How strongly a policy optimises against a model is set by the protocol, not inferred from text. This control is what makes Section 4.4 count as evidence at all.

A simulated-learner tutoring leaderboard is weakly determined by its own data and prefers policies planned against the scoring architecture. Excluding the own-architecture scorers is free and changes the answer.

2 Related work

Simulated users as evaluation instruments. Agentic benchmarks increasingly pair an agent with a simulated counterpart, and recent work questions whether that counterpart is a faithful proxy. Simulated users driven by prompted language models are unreliable stand-ins for people (Seshadri et al., 2026), display cooperativeness absent from real users (Chopra et al., 2026) and show a measurable simulation-to-reality gap (Zhou et al., 2026). In education, the same substitution is now routine. Generative agents (Gao et al., 2025; Xu et al., 2025) and profiles built from interaction history (Duan et al., 2026) simulate student behaviour, and purpose-built frameworks evaluate the simulations (Moreau-Pernet, 2026). Simulated students match real ones poorly in behaviour (Scarlatos et al., 2026) and ability (Srivatsa et al., 2025). Yet the same simulated student often both trains a language-model tutor and scores it (Dinucu-Jianu et al., 2025; Scarlatos et al., 2025). The mechanism studied here differs in three respects. The simulator is a model fitted to interaction data rather than a prompted persona. The outcome is a probe measurement rather than a judgement issued by a language model. The measured disagreement is larger than noise, and a directional bias accompanies it. The present work also differs in its comparison. A difference between rankings from two simulators is uninformative without the agreement of one simulator with itself. This paper establishes that agreement first.

How well a leaderboard is determined. Uncertainty in model rankings is increasingly treated as a quantity to report rather than a caveat to mention. Benchmark scores vary enough across training seeds that model comparisons must account for this uncertainty (Madaan et al., 2024). This uncertainty argues for evaluation error bars (Miller, 2024) and for selecting benchmarks by their ratio of signal to noise (Heineman et al., 2025). Neuhof & Benjamini (2026) construct rank intervals with statistical guarantees, Patel et al. (2026) rank configurations by how well an in-sample ranking predicts an out-of-sample one, and Ndzomga (2026) measures how much of a benchmark can be dropped before rank fidelity fails. A leaderboard can also be systematically biased, for example when selective disclosure fits submissions to one evaluator (Singh et al., 2025). The resampling

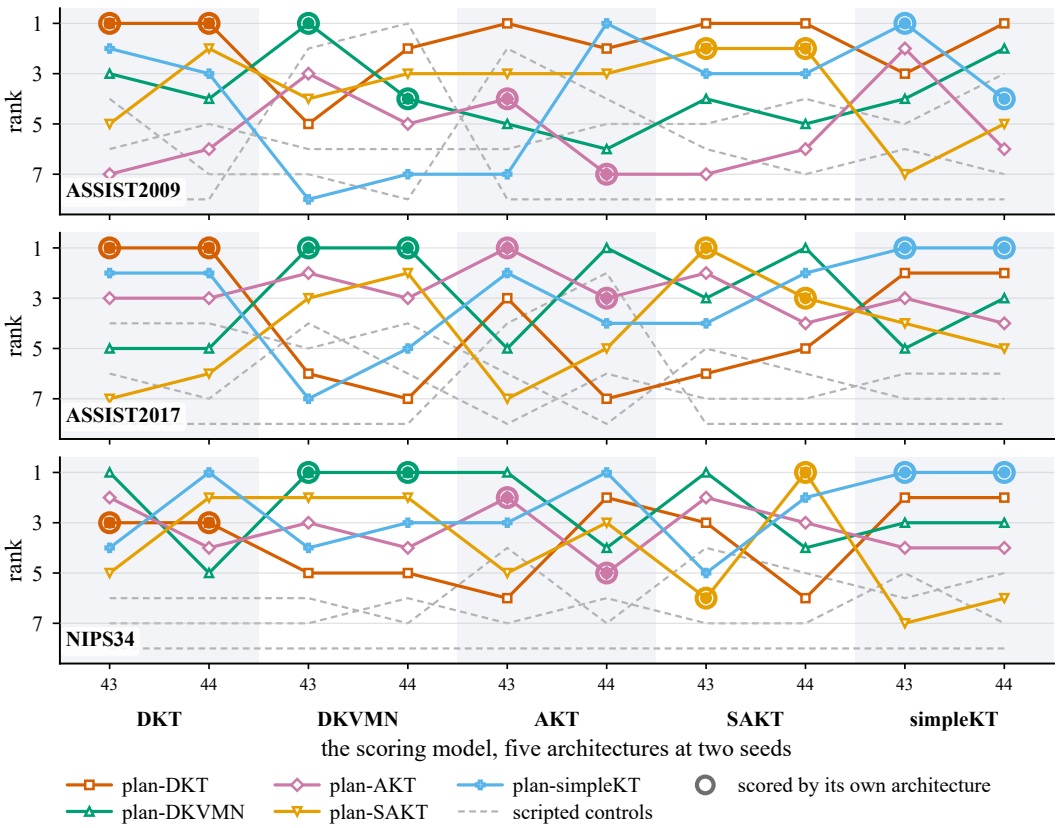

Figure 1: Every leaderboard the protocol produces, on all three corpora. Each policy has one line, each scorer has one column and the vertical axis shows rank. Rings mark the two scorers of the architecture a policy plans against. Crossing lines indicate scorer disagreement. Rings near the top of each corpus show the self-preference. On ASSIST2017 PLAN-DKT ranks first under both DKT scorers and seventh under an AKT scorer.

floor used here belongs to that line of work and addresses the same question for a simulated-learner leaderboard. The answer is that the ranking is weakly determined at the sample sizes in use.

Tutoring benchmarks grounded in knowledge tracing. Knowledge tracing predicts the next response from the history of a learner (Abdelrahman et al., 2023). Recurrent (DKT; Piech et al., 2015), memory-augmented (DKVMN; Zhang et al., 2017) and attentive (SAKT, AKT and simpleKT; Pandey & Karypis, 2019; Ghosh et al., 2020; Liu et al., 2023) architectures are benchmarked under a common protocol (Liu et al., 2022). Recent models use pretrained language-model representations (Norris et al., 2025), make item difficulty explicit (Huang et al., 2026) or train for the multi-step rollouts a simulator needs (Fu et al., 2025). A fitted model of this kind is the standard environment for learning-path and exercise recommendation. A policy is trained against the fitted model, and the learning gain of the policy is measured in a simulator (Chen et al., 2023b;a; Cheng et al., 2026; Wang et al., 2026). A benchmark for tutoring agents adopts the same arrangement (Lee et al., 2026). In several recommendation studies, one fitted model supplies both the training feedback and the reported learning gain (Chen et al., 2023b;a; Wang et al., 2026).

Self-preference in automatic evaluation. Language models used as evaluators show self-preference. Early work reports possible self-preference but cannot confirm it (Zheng et al., 2023). Later studies find that evaluators favour their own outputs (Panickssery et al., 2024; Wataoka et al., 2024) and models similar to themselves (Goel et al., 2025). A judge also favours students trained on data from a related generator (Li et al., 2026). The effect survives careful measurement (Chen

et al., 2025; Spiliopoulou et al., 2025), is only partly removed by mitigation (Yang et al., 2026) and runs in both directions when the source is labelled (Chae et al., 2026). Closest to the present setting, Pombal et al. (2026) find that the effect persists even when the evaluation criteria are programmatically verifiable. The effect reported here is the analogue for simulated learners, but the mechanism differs in kind. The scorer generates none of the teaching it scores, and the preference operates over sequences of exercises selected by a planner rather than over text. The difference allows the contrast of Section 4.4 to serve as evidence. Here the protocol sets whether and how strongly a policy optimises against the scoring architecture, whereas in the text setting neither is under control.

Optimising against a learned proxy. A planner that searches against a fitted learner model optimises a learned proxy. The reward-model literature describes the consequences. Stronger optimisation against a reward model raises its score whereas the gold objective stalls and then falls (Gao et al., 2023; Karwowski et al., 2024). Best-of-n selection is the inference-time counterpart of the random shooting used here and follows the same pattern (Khalaf et al., 2025). A policy aligned to one reward model gains little under another fitted to the same data, and ensembles reduce this effect without removing it (Eisenstein et al., 2024). Here the proxy and the scorer are models of the same kind fitted to the same data, which allows their divergence to be measured directly and without a gold objective. The protocol sets the search budget directly, so the excess gain under own-architecture scorers can be traced as optimisation pressure grows (Appendix C).

3 PROTOCOL

3.1 POLICIES

Figure 5 in Appendix D presents the full design, and Figure 1 shows the result. A policy observes the first $t_0 = 50$ interactions of a simulated learner and selects $K = 10$ exercises from a pool. Five of the eight policies plan. They sample candidate sequences, roll them forward under one knowledge-tracing model and keep the sequence that maximises imagined probe correctness. The five differ only in which knowledge-tracing model they use, and each is named after its architecture, as in PLAN-DKT.

Access to model internals gives a planner an implementation advantage and confounds the comparison. All five therefore use the single interface that every backend already exposes, namely the predicted correctness of the next response given a history. No planner sees anything the others cannot.

Three controls complete the leaderboard. RANDOM samples from the pool and ignores the learner model. GREEDY selects the items DKT rates lowest, consulting the model without planning with it. MATCHED reproduces the empirical difficulty profile of the sequence planned against DKT but chooses each item by difficulty alone. Difficulty is the fraction of correct responses to an item in the observed histories, so this control consults no model when choosing. Neither GREEDY nor MATCHED plans against a model. Both are also rebuilt for each of the five models and scored alongside the leaderboard but outside it, so every planner can be compared with a GREEDY control built on its model and a MATCHED control built on its sequence.

The difficulty-matched control is not optional. Against GREEDY, every planner appears to succeed. On ASSIST2009 the GREEDY controls built on the five models lose 0.0270 to 0.0375 of gain averaged over the scorers, whereas MATCHED loses 0.0063. Teaching the lowest-rated items means teaching the hardest ones, which most scorers penalise regardless of the model that picks them. Without MATCHED, a comparison measures difficulty selection, not teaching.

3.2 SCORING

The chosen sequences are frozen. A scorer then acts as the learner over the selected exercises. Learning gain is the change in mean predicted correctness over a held-out probe, measured before and after the taught block. Probe items are withheld from the schedulable pool, so no policy can teach them. Each probe item is appended separately, so no measurement conditions the next.

The 10 scorers cover five architectures at two seeds each. Each policy is scored by all 10 scorers, including the two scorers of the architecture the policy plans against. Any difference between two

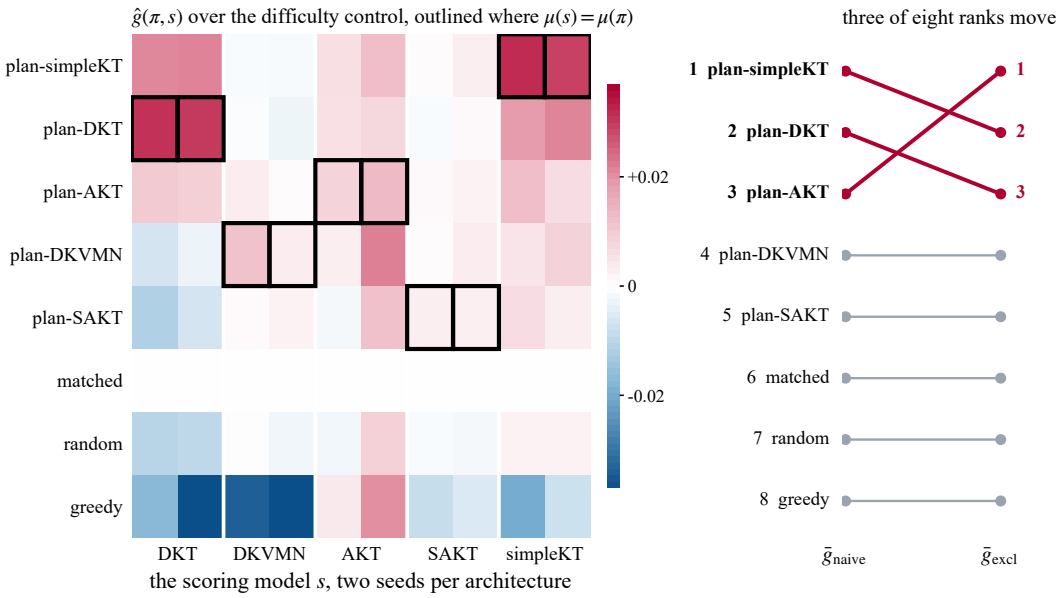

Figure 2: Equation 2 illustrated on ASSIST2017. The left panel shows $\hat{g}(\pi, s) - \hat{g}(\text{MATCHED}, s)$ for every policy and scorer, which makes the columns comparable. The estimators average the raw $\hat{g}(\pi, s)$. Cells where $\mu(s) = \mu(\pi)$ are outlined. These are the cells that $\bar{g}_{\text{excl}}$ drops, and one of them is the warmest cell in three of the five planning rows. MATCHED is white by construction. GREEDY is the bottom band, which is blue under most scorers and shows the difficulty artefact that MATCHED removes. The right panel shows the ranking under each estimator. Three of eight positions move here, and the reported winner changes.

leaderboards is therefore attributable to the scorer rather than to what is taught. The sequences are chosen once for each learner and shared, so all comparisons are paired on the same learners.

3.3 Notation and the two leaderboards

Let $\mathcal{U}$ be the simulated learners, $|\mathcal{U}| = n$, and let $\mathcal{S}$ be the scorers, $|\mathcal{S}| = 10$. Each scorer s is one fitted model, and $\mu(s)$ denotes its architecture. A planning policy π_a plans against a fitted model of architecture a, so $\mu(\pi_a) = a$. A control plans against nothing, so $\mu(\pi) = \varnothing$.

For learner u with history x_u of length t_0, a policy returns a block of K items $\pi(x_u)$, chosen once and then frozen. With $p_s(q \mid \cdot)$ the correctness probability predicted by scorer s on probe item $q \in Q$, the gain is

$$g(\pi, s, u) \;=\; \frac{1}{|Q|} \sum_{q \in Q} \Big[p_s\big(q \mid x_u \oplus \pi(x_u)\big) - p_s\big(q \mid x_u\big) \Big], \qquad \hat{g}(\pi, s) \;=\; \frac{1}{n} \sum_{u \in \mathcal{U}} g(\pi, s, u). \quad (1)$$

Every probe item is appended separately to x_u and to $x_u \oplus \pi(x_u)$, so no probe measurement conditions another. The probe set Q is disjoint from the pool that π draws from.

A benchmark reports a ranking of the eight policies Π by the average over its scorers. Two averages are compared throughout:

$$\bar{g}_{\text{naive}}(\pi) = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \hat{g}(\pi, s), \quad \bar{g}_{\text{excl}}(\pi) = \frac{1}{|\mathcal{S}_\pi|} \sum_{s \in \mathcal{S}_\pi} \hat{g}(\pi, s), \quad \mathcal{S}_\pi = \{s : \mu(s) \neq \mu(\pi)\}. \quad (2)$$

The second is the exclusion estimator, shown in Figure 2. It is the whole of the correction evaluated in Section 4.6. Under this estimator, a policy is never scored by the architecture it optimises against. Controls have $\mathcal{S}_\pi = \mathcal{S}$ and are unaffected. The rule excludes two of 10 scorers per planning policy and needs no additional computation.

Algorithm 1 The measurement. Sequences are chosen once and frozen before scoring.

draw the probe Q and the pool P once per corpus, $Q \cap P = \varnothing$
for each learner $u \in \mathcal{U}$ **do**
  for each policy $\pi \in \Pi$ **do**
    $B^u_\pi \leftarrow \pi(x_u)$ ▷ K items from P, frozen from here on
for each scorer $s \in \mathcal{S}$ **do**
  for each policy π, each learner u **do**
    roll s forward over B^u_π and record $g(\pi, s, u)$ by Eq. 1
  $L_s \leftarrow$ the ranking of Π by $\hat{g}(\cdot, s)$ ▷ one leaderboard per scorer
$\tau_{\text{floor}}, \tau_{\text{cross}} \leftarrow$ Eq. 3 ▷ report the floor beside the ranking
return the ranking by $\bar{g}_{\text{excl}}$ of Eq. 2 ▷ never the naive one

3.4 The null that matters

The relevant null is not a random ranking. A leaderboard moves under resampling even when nothing else changes. This movement sets the floor against which scorer disagreement is compared. Let $\hat{g}_V$ denote the mean of Equation 1 restricted to a subset $V \subseteq \mathcal{U}$, and let τ denote Kendall τ-b over the vector of policy gains. The resampling floor and the quantity compared with it are

$$\tau_{\text{floor}} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \tau\big(\hat{g}_{\mathcal{U}_1}(\cdot, s),\ \hat{g}_{\mathcal{U}_2}(\cdot, s)\big), \qquad \tau_{\text{cross}} = \operatorname*{mean}_{\mu(s) \neq \mu(s')} \tau\big(\hat{g}(\cdot, s),\ \hat{g}(\cdot, s')\big), \tag{3}$$

for a random split $\mathcal{U} = \mathcal{U}_1 \sqcup \mathcal{U}_2$ into halves. The tested claim is $\tau_{\text{cross}} < \tau_{\text{floor}}$. Algorithm 1 states the measurement end to end, with the freeze as an explicit step between choosing and scoring.

The floor is deliberately conservative. Each half contains $n/2$ learners, whereas cross-scorer agreement uses all n learners. The floor therefore uses a smaller sample than the quantity compared with it and understates within-scorer agreement. In expectation, any gap below it is a lower bound.

3.5 Scale

Three public corpora, ASSIST2009 (Feng et al., 2009), ASSIST2017 (Patikorn et al., 2020) and NIPS34 (Wang et al., 2020), give $287/894/881$ simulated learners, 2,062 in total from 999 students. All 10 scorers score all eight policies on every learner. The floor, the same- and cross-architecture agreements, and self-preference are primary endpoints, fixed in writing before the first run.

4 Results

4.1 The leaderboard is weakly determined under a fixed scorer

Holding the scorer fixed and resampling learners gives $\tau = 0.524 \pm 0.036$ (standard error) between the two halves. A different draw of a few hundred learners and their sampled responses (Appendix F) inverts roughly a quarter of the pairwise orderings of eight policies. This instability does not mean that the corpora fail to separate the policies. The mean within-scorer standard deviation of policy gain is 2.81, 2.97 and 4.87 times the mean standard error on the three corpora.

The obvious remedy is more learners. The floor rises from $\tau = 0.215$ at 25 learners per half to 0.548 at 400 (the two largest sizes are measured on the two larger corpora). The rise is close to linear in the logarithm of sample size ($R^2 = 0.997$). Across these sizes, τ increases by 0.082 per doubling (Appendix D). This paper takes $\tau = 0.8$ as the agreement of a ranking one can act on. Extrapolated that far, the line calls for roughly 7,300 simulated learners, some eight times the largest corpus here. That number is outside the measured range and is an order of magnitude, not a target. Reaching such a ranking of eight similar policies is no small change to current practice.

4.2 Changing the scorer costs more than resampling the learners

Figure 3(a) summarises the agreement of the leaderboards in Figure 1. Two scorers with different architectures agree at $\tau = 0.335$, which is 0.189 below the resampling floor (bootstrap 95% interval

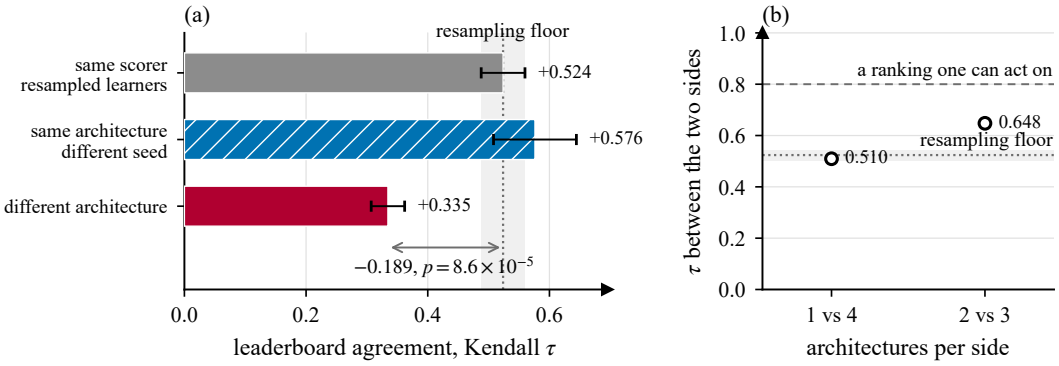

Figure 3: (a) Agreement between leaderboards. The dotted line and band mark the resampling floor, the agreement of one scorer with itself on disjoint halves of the learners. Two seeds of one architecture agree just above the floor. Two different architectures agree a further 0.189 below it. Bars show means over 30 scorer-corpus cells, 15 same-architecture pairs and 120 cross-architecture pairs. Whiskers show standard errors. (b) Mean agreement between the two averaged sides over every partition of the five architectures, one against four and two against three. On average, neither size reaches a ranking one can act on.

$[+0.048, +0.237]$, Appendix E). Even the three least noisy architectures agree at only $\tau = 0.357$ on ASSIST2017, against a floor of 0.586 (Appendix F), so scorer choice is a first-order concern.

Two scorers of the same architecture at different seeds agree at $\tau = 0.576$, just above the floor and 0.242 above the cross-architecture agreement (Welch $p = 0.0039$). These refits cost no more agreement than resampling. The disagreement is architectural on both ASSISTments corpora, where it persists among stronger scorers (Appendix D). On NIPS34, two seeds of one architecture disagree as much as two architectures do (Appendix E).

4.3 Scorers also prefer their own architecture

The scores also carry a directional bias. A scorer ranks a policy planned against its own architecture 2.17 on average, compared with 3.73 for scorers of other architectures. This advantage of 1.57 rank positions ($p = 4.4 \times 10^{-5}$, one-sided) is present in 13 of 15 policy-corpus cells. The scorer and the planner use models fitted at different seeds, so the preference is for the architecture rather than for shared parameters. An exact assignment test confirms it (Appendix E).

The advantage is positive on every corpus. It is +2.52 rank positions on ASSIST2017 (two-sided $p = 0.0047$), +1.22 on ASSIST2009 ($p = 0.039$) and +0.95 on NIPS34 ($p = 0.075$). It is positive for all five planning policies on ASSIST2017 and for four of five on the other two corpora, where the policy planned against AKT is the exception. Table 2 in Appendix A gives the ranks by corpus.

4.4 Self-preference follows the optimisation

Two controls in the design test whether self-preference follows the optimisation. The first separates planning from model use. A GREEDY control built on the model of each planner reads that model but does not plan against it. Ranked among themselves under each scorer, the five planners gain 1.33 positions from the scorers of their own architecture ($p = 1.4 \times 10^{-5}$). The five GREEDY controls gain −0.04 ($p = 0.89$). This difference holds in a paired comparison over the 15 cells ($p = 0.0008$). Self-preference follows planning against an architecture, not the greedy use of its model.

The second control is difficulty-matched to the sequence of each planner and expresses the preference in units of gain. The own-architecture scorers are the two scorers of the same architecture as the planner. In 13 of 15 cells, they see more gain than the other eight, by 0.0088 on average ($p = 0.0032$, one-sided). On ASSIST2017, this excess grows from +0.0061 to +0.0171 as the search budget increases 16-fold, from 12 through 48 to 192 candidates. The arms share their learners, probe and pool, so only the search budget and the random draws differ. Every step is sig-

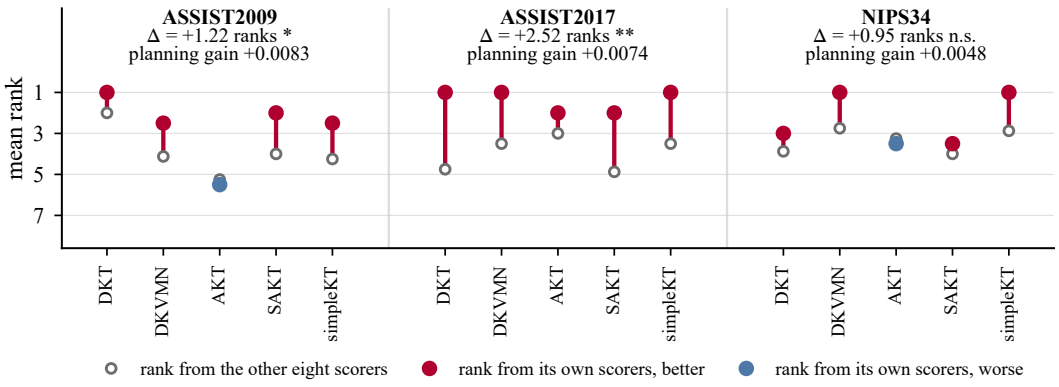

Figure 4: Self-preference in 15 policy-corpus cells. Each segment compares the mean rank a policy receives from its two own-architecture scorers with the rank from the other eight, and points up when those scorers favour it. Panels give Δ (other minus own rank) with its test result from Table 2 (* $p < 0.05$, ** $p < 0.01$, n.s. $p \geq 0.05$) and the planning gain over MATCHED. Thirteen of the 15 segments point up, and both exceptions are the policy planned against AKT.

Table 1: The top of the leaderboard under the naive average and under the exclusion estimator in Equation 2. All entries are planning policies, named after the architecture they plan against.

Corpus	Naive top three	Exclusion top three	Ranks moved
ASSIST2009	DKT > simpleKT > DKVMN	DKT > simpleKT > SAKT	3 of 8
ASSIST2017	simpleKT > DKT > AKT	**AKT** > simpleKT > DKT	3 of 8
NIPS34	DKVMN > simpleKT > AKT	**simpleKT** > DKT > AKT	5 of 8

nificant, and the excess grows over that range for every planner. The gain the other architectures see moves from $+0.0036$ to $+0.0054$ (Appendix C). Further search yields mostly gain that only the own-architecture scorers see. Since no scorer has privileged validity, gain that only one architecture sees cannot be credited to the teaching. Ranks hide it because a scorer already places its own policy near the top at the smallest budget.

Figure 4 shows self-preference cell by cell and the planning gain per corpus. The planning policies beat the shared MATCHED on every corpus, by $+0.0083$ on ASSIST2009, $+0.0074$ on ASSIST2017 and $+0.0048$ on NIPS34. On every corpus a scorer also gives its own policy an advantage. Across corpora it does not track the planning gain. Corpora differ in more than the optimisation, whereas the search-budget arms of Appendix C change no other setting.

4.5 A LARGER SCORER PANEL DOES NOT REPAIR THE RANKING

The intuitive remedy for scorer disagreement is to average more scorers. Over all 15 partitions of the five architectures, the averaged leaderboards of the two sides agree at $\tau = 0.510$ for one architecture against four and at 0.648 for two against three (Figure 3(b)). On average, neither size reaches a ranking one can act on in any corpus, and only four of the 45 partitions over the three corpora do.

4.6 EXCLUDING OWN-ARCHITECTURE SCORERS REORDERS EVERY CORPUS LEADERBOARD

Replacing $\bar{g}_{\text{naive}}$ with $\bar{g}_{\text{excl}}$ of Equation 2 is free and requires only that each policy declare the architecture it plans against. Table 1 reports the effect. On each corpus, three to five of the eight positions move, and the reported winner changes on ASSIST2017 and on NIPS34. On gains measured against MATCHED under each scorer, the exclusion moves four positions on ASSIST2009 and none on ASSIST2017, and it changes the NIPS34 winner on both scales (Table 7). Leaderboards of different corpora agree at $\tau = 0.357$, 0.429 and 0.786 after the correction and at 0.643, 0.429 and 0.643 before it, so the correction does not make the corpora agree.

4.7 Self-preference and disagreement are separate errors

Self-preference and scorer disagreement are separate phenomena, and the correction addresses only the first. Recomputing cross-architecture agreement over the six policies that plan against neither architecture leaves τ nearly unchanged, at 0.340 against 0.335 ($p = 0.77$). Dropping two planning policies at random instead shortens the ranking by the same amount and gives 0.393 ($p = 0.012$ against the preceding version). Excluding the policies that carry the bias does not raise the agreement. The exclusion estimator removes the self-preference, not the disagreement.

5 What follows for benchmark practice

Three changes follow. Report the resampling floor beside the ranking. Reporting it costs one more split of the learners already collected, and orderings that reverse between halves are not established. Do not score a policy with the architecture it plans against. This rule removes the self-preference at no cost wherever policies optimise against a learner model, even when they barely search. At 12 candidate sequences, the excess gain is already $+0.0061$ (Appendix C). Do not treat a larger scorer panel as a fix. It is the obvious remedy but fails on average at every size tested. None of the three needs more than the per-learner gains a benchmark already records. Where a corpus is exhausted, scoring each learner several times is expected to raise the floor (Appendix F).

6 What holds across runs

The design is run twice. The second run has 2.6 times as many learners, including those of the first run. Its new sampling seed changes the probe, the item pool and the random draws of every rollout. The correction changes the winner on NIPS34 in both runs and on ASSIST2017 only in the second, so that flip depends on the run. In the first run, the gap between same- and cross-architecture agreement passes only the randomisation test of Appendix E and is absent on ASSIST2009. The excess gain of that run appears in nine of 15 cells rather than 13 (Appendix B). The gap to the floor is 0.151 then 0.189, and its bootstrap interval excludes zero in the second run only. Self-preference is $+1.54$ then $+1.57$ ranks, and the GREEDY control receives none in either run.

7 Limitations

The learners are simulated throughout, so nothing here establishes what a real student learns. An internally consistent leaderboard can still fail to predict human learning. The intervention is a single block of 10 exercises, which bounds achievable gain and may lower the floor. Scoring each learner once is also estimated to lower it (Appendix F). Whether scorer identity and self-preference survive a longer intervention remains untested. Five architectures and three mathematics corpora do not exhaust either space, and the policies are automated rather than hand-tuned tutors. The dose response is measured on one corpus. Removing the self-preference does not reduce the disagreement between scorers. This design locates that disagreement in the architectures on the two ASSISTments corpora but does not diagnose it.

8 Conclusion

A tutoring leaderboard scored by a fitted model is weakly determined by its own data. Its reported ranking also depends on the architecture chosen to score it. Sampling noise is large but shrinks with more learners. Scorer identity lowers τ by a further 0.189. Scorers also prefer policies optimised against their own architecture. A greedy control that uses the same model without planning against it receives no detectable preference, and in gain the preference grows with the search budget on ASSIST2017. Excluding own-architecture scorers removes this preference but leaves the disagreement intact. No panel size tested yields, on average, a ranking one can act on. Reporting the resampling floor and excluding own-architecture scorers are cheap, but neither is current practice.

AI USE STATEMENT

Generative AI tools assist in preparing this work. They help design and code experiments and analyses, which the authors test and verify. They help find related work, and the authors check it against the published records. They help draft and copyedit manuscript text, and the authors review and edit it line by line. Generative AI is not used to generate, alter or select any experimental data. Every reported number is recomputed from the recorded outputs by the analysis scripts. The authors review all AI-assisted work and take full responsibility for this publication.

ETHICS STATEMENT

The experiments use public knowledge-tracing corpora released for research use and containing no personally identifying information. No new data are collected from human subjects, and no human subject interacts with any system described here. The finding concerns the validity of an evaluation protocol. Its intended consequence is that benchmarks built on fitted simulators report more conservative comparisons.

REPRODUCIBILITY STATEMENT

The code and recorded outputs will be released. A script recomputes every result figure and table from the recorded outputs. The corpora are public, all five architectures are published, and the runner reads fitted models through a documented one-function interface. The release includes both runs of the design, at 789 and 2,062 learners, together with the search-budget arms and per-model controls. It records which second-run findings the first run supports. Recomputation requires neither a GPU nor a fitted model. A further script recomputes 89 headline numbers from the recorded outputs and checks each against the text.

REFERENCES

Ghodai Abdelrahman, Qing Wang, and Bernardo Pereira Nunes. Knowledge tracing: A survey. *ACM Computing Surveys*, 55(11):1–37, 2023.

Songeun Chae, Min Kim, Donghoon Jung, Seojin Choi, and Seohyon Jung. Self- and other-labels induce bidirectional bias in LLM judges. *arXiv preprint arXiv:2608.18091*, 2026.

Jyun-Yi Chen, Saeed Saeedvand, and I-Wei Lai. Adaptive learning path navigation based on knowledge tracing and reinforcement learning. *arXiv preprint arXiv:2305.04475*, 2023a.

Xianyu Chen, Jian Shen, Wei Xia, Jiarui Jin, Yakun Song, Weinan Zhang, Weiwen Liu, Menghui Zhu, Ruiming Tang, Kai Dong, Dingyin Xia, and Yong Yu. Set-to-sequence ranking-based concept-aware learning path recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 5027–5035, 2023b.

Zhi-Yuan Chen, Hao Wang, Xinyu Zhang, Enrui Hu, and Yankai Lin. Beyond the surface: Measuring self-preference in LLM judgments. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1653–1672, 2025.

Xinghe Cheng, Zihan Zhang, Jiapu Wang, Liangda Fang, Chaobo He, Quanlong Guan, Shirui Pan, and Weiqi Luo. GraphRAG-induced dual knowledge structure graphs for personalized learning path recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 14610–14620, 2026.

Harshita Chopra, Kshitish Ghate, Aylin Caliskan, Tadayoshi Kohno, Chirag Shah, and Natasha Jaques. Beyond cooperative simulators: Generating realistic user personas for robust evaluation of LLM agents. *arXiv preprint arXiv:2605.12894*, 2026.

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. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 272–292, 2025.

Zhangqi Duan, Shuyan Huang, Alexander Scarlatos, Jaewook Lee, Simon Woodhead, and Andrew Lan. Who am I? history-aware profiles for student simulation in tutoring dialogues. *arXiv preprint arXiv:2605.30051*, 2026.

Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alex D'Amour, DJ Dvijotham, Adam Fisch, Katherine Heller, Stephen Pfohl, Deepak Ramachandran, Peter Shaw, and Jonathan Berant. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking. In *Conference on Language Modeling*, 2024. arXiv:2312.09244.

Mingyu Feng, Neil T. Heffernan, and Kenneth R. Koedinger. Addressing the assessment challenge with an online system that tutors as it assesses. *User Modeling and User-Adapted Interaction*, 19 (3):243–266, 2009.

Lingyue Fu, Ting Long, Jianghao Lin, Wei Xia, Xinyi Dai, Ruiming Tang, Yasheng Wang, Weinan Zhang, and Yong Yu. AdvKT: An adversarial multi-step training framework for knowledge tracing. In *Machine Learning and Knowledge Discovery in Databases, Research Track*, Lecture Notes in Computer Science, pp. 183–200, 2025. arXiv:2504.04706.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10835–10866, 2023.

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. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 23923–23932, 2025. arXiv:2501.10332.

Aritra Ghosh, Neil Heffernan, and Andrew S. Lan. Context-aware attentive knowledge tracing. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020.

Shashwat Goel, Joschka Strüber, Ilze Amanda Auzina, Karuna K. Chandra, Ponnurangam Kumaraguru, Douwe Kiela, Ameya Prabhu, Matthias Bethge, and Jonas Geiping. Great models think alike and this undermines AI oversight. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 19621–19678, 2025.

David Heineman, Valentin Hofmann, Ian Magnusson, Yuling Gu, Noah A. Smith, Hannaneh Hajishirzi, Kyle Lo, and Jesse Dodge. Signal and noise: A framework for reducing uncertainty in language model evaluation. In *Advances in Neural Information Processing Systems*, volume 38, 2025. arXiv:2508.13144.

Shuyan Huang, Alexander Scarlatos, Jaewook Lee, and Andrew Lan. Interpretable difficulty-aware knowledge tracing in tutor-student dialogues. *arXiv preprint arXiv:2605.01097*, 2026.

Jacek Karwowski, Oliver Hayman, Xingjian Bai, Klaus Kiendlhofer, Charlie Griffin, and Joar Skalse. Goodhart's law in reinforcement learning. In *International Conference on Learning Representations*, 2024. arXiv:2310.09144.

Hadi Khalaf, Claudio Mayrink Verdun, Alex Oesterling, Himabindu Lakkaraju, and Flavio du Pin Calmon. Inference-time reward hacking in large language models. In *Advances in Neural Information Processing Systems*, volume 38, 2025. arXiv:2506.19248.

Unggi Lee, Sookbun Lee, Yeil Jeong, Eunjoo Lee, Minchul Shin, and Hoilym Kwon. EduClaw-Bench: A long-horizon benchmark for pedagogical LLM agents with simulated learners. *arXiv preprint arXiv:2608.03206*, 2026.

Dawei Li, Renliang Sun, Yue Huang, Ming Zhong, Bohan Jiang, Jiawei Han, Xiangliang Zhang, Wei Wang, and Huan Liu. Preference leakage: A contamination problem in LLM-as-a-judge. In *International Conference on Learning Representations*, 2026. arXiv:2502.01534.

Zitao Liu, Qiongqiong Liu, Jiahao Chen, Shuyan Huang, Jiliang Tang, and Weiqi Luo. pyKT: A python library to benchmark deep learning based knowledge tracing models. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 35, 2022.

Zitao Liu, Qiongqiong Liu, Jiahao Chen, Shuyan Huang, and Weiqi Luo. simpleKT: A simple but tough-to-beat baseline for knowledge tracing. In *International Conference on Learning Representations*, 2023. arXiv:2302.06881.

Lovish Madaan, Aaditya K. Singh, Rylan Schaeffer, Andrew Poulton, Sanmi Koyejo, Pontus Stenetorp, Sharan Narang, and Dieuwke Hupkes. Quantifying variance in evaluation benchmarks. *arXiv preprint arXiv:2406.10229*, 2024.

Evan Miller. Adding error bars to evals: A statistical approach to language model evaluations. *arXiv preprint arXiv:2411.00640*, 2024.

Baptiste Moreau-Pernet. EvalConvoLearn: An open-source framework for evaluating grounded learner simulations in tutoring conversations. In *Proceedings of the Impactful and Responsible AI Systems for Education Workshop*, volume 339 of *Proceedings of Machine Learning Research*, pp. 180–186, 2026. arXiv:2608.07497.

Franck Ndzomga. Efficient benchmarking of AI agents. *arXiv preprint arXiv:2603.23749*, 2026.

Bitya Neuhof and Yuval Benjamini. Rank intervals for leaderboards: A hierarchical framework for model evaluation. *arXiv preprint arXiv:2606.08679*, 2026.

Max Norris, Kobi Gal, and Sahan Bulathwela. Next token knowledge tracing: Exploiting pretrained LLM representations to decode student behaviour. *arXiv preprint arXiv:2511.02599*, 2025.

Shalini Pandey and George Karypis. A self-attentive model for knowledge tracing. In *Proceedings of the 12th International Conference on Educational Data Mining*, 2019. arXiv:1907.06837.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. In *Advances in Neural Information Processing Systems*, volume 37, 2024. arXiv:2404.13076.

Dhaval C. Patel, Kaoutar El Maghraoui, Shuxin Lin, Yusheng Li, Tianjun Feng, Chun-Yi Tsai, et al. Beyond static leaderboards: Predictive validity for the evaluation of LLM agents. *arXiv preprint arXiv:2606.19704*, 2026.

Thanaporn Patikorn, Ryan S. Baker, and Neil T. Heffernan. ASSISTments longitudinal data mining competition special issue: A preface. *Journal of Educational Data Mining*, 12(2):i–xi, 2020.

Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. In *Advances in Neural Information Processing Systems*, 2015.

José Pombal, Ricardo Rei, and André F. T. Martins. Self-preference bias in rubric-based evaluation of large language models. *arXiv preprint arXiv:2604.06996*, 2026.

Alexander Scarlatos, Naiming Liu, Jaewook Lee, Richard Baraniuk, and Andrew Lan. Training LLM-based tutors to improve student learning outcomes in dialogues. In *Artificial Intelligence in Education*, volume 15877 of *Lecture Notes in Computer Science*, pp. 251–266, 2025. arXiv:2503.06424.

Alexander Scarlatos, Jaewook Lee, Simon Woodhead, and Andrew Lan. Simulated students in tutoring dialogues: Substance or illusion? In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 42349–42385, 2026. arXiv:2601.04025.

Preethi Seshadri, Samuel Cahyawijaya, Ayomide Odumakinde, Sameer Singh, and Seraphina Goldfarb-Tarrant. Lost in simulation: LLM-simulated users are unreliable proxies for human users in agentic evaluations. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 47423–47439, 2026. arXiv:2601.17087.

Shivalika Singh, Yiyang Nan, Alex Wang, Daniel D'Souza, Sayash Kapoor, Ahmet Üstün, Sanmi Koyejo, Yuntian Deng, Shayne Longpre, Noah A. Smith, Beyza Ermis, Marzieh Fadaee, and Sara Hooker. The leaderboard illusion. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 38, 2025.

Evangelia Spiliopoulou, Riccardo Fogliato, Hanna Burnsky, Tamer Soliman, Jie Ma, Graham Horwood, and Miguel Ballesteros. Play favorites: A statistical method to measure self-bias in LLM-as-a-judge. *arXiv preprint arXiv:2508.06709*, 2025.

KV Aditya Srivatsa, Kaushal Kumar Maurya, and Ekaterina Kochmar. Can LLMs reliably simulate real students' abilities in mathematics and reading comprehension? In *Proceedings of the 20th Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 988–1001, 2025.

Shuai Wang, Yaoming Yang, Bingdong Li, Hao Hao, and Aimin Zhou. IB-GRPO: Aligning LLM-based learning path recommendation with educational objectives via indicator-based group relative policy optimization. *arXiv preprint arXiv:2601.14686*, 2026.

Zichao Wang, Angus Lamb, Evgeny Saveliev, Pashmina Cameron, Yordan Zaykov, José Miguel Hernández-Lobato, Richard E. Turner, Richard G. Baraniuk, Craig Barton, Simon Peyton Jones, Simon Woodhead, and Cheng Zhang. Diagnostic questions: The NeurIPS 2020 education challenge. *arXiv preprint arXiv:2007.12061*, 2020.

Koki Wataoka, Tsubasa Takahashi, and Ryokan Ri. Self-preference bias in LLM-as-a-judge. In *NeurIPS Workshop on Safe Generative AI*, 2024. arXiv:2410.21819.

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. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pp. 1–26, 2025. arXiv:2502.02780.

Jinming Yang, Zheng Hu, Chuxian Qiu, Zhenyu Deng, Xinshan Jiao, and Tao Zhou. Quantifying and mitigating self-preference bias of LLM judges. *arXiv preprint arXiv:2604.22891*, 2026.

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*, 2017. arXiv:1611.08108.

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. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 36, 2023.

Xuhui Zhou, Weiwei Sun, Qianou Ma, Yiqing Xie, Jiarui Liu, Weihua Du, Sean Welleck, Yiming Yang, Graham Neubig, Sherry Tongshuang Wu, and Maarten Sap. Mind the Sim2Real gap in user simulation for agentic tasks. In *Conference on Language Modeling*, 2026. arXiv:2603.11245.

A PER-CORPUS LEADERBOARDS

Table 2 reports mean self-preference by corpus. Table 3 compares this self-preference with the gains of the planning policies over MATCHED on the same corpus.

Table 2: Self-preference per corpus. Own is the mean rank from the two scorers of the architecture a policy plans against. Other is the mean rank from the remaining eight scorers. Lower rank is better, so a positive Δ (other minus own, before rounding) indicates self-preference. p is from a paired t test over policies, two-sided for each corpus and one-sided for the 15 pooled policy-corpus cells.

Corpus	Own rank	Other rank	Δ	p	Cells positive
ASSIST2009	2.70	3.92	$+1.22$	0.039	4 of 5
ASSIST2017	1.40	3.92	$+2.52$	0.0047	5 of 5
NIPS34	2.40	3.35	$+0.95$	0.075	4 of 5
Pooled	2.17	3.73	$+1.57$	4.4×10^{-5}	13 of 15

Tables 4 to 6 give the full eight-policy leaderboard for each corpus. They report the naive average over all 10 scorers and an average that excludes the two scorers of the architecture each policy plans

Table 3: Self-preference per corpus against the mean gain of the planning policies over MATCHED. Spread is the mean standard deviation of the eight policy gains in mean standard errors.

Corpus	Planning gain over MATCHED	Self-preference Δ	Spread (s.e.)
ASSIST2009	$+0.0083$	$+1.22$	2.81
ASSIST2017	$+0.0074$	$+2.52$	2.97
NIPS34	$+0.0048$	$+0.95$	4.87

against. The analysis script generates the tables from the recorded per-learner outputs. The exclusion estimator scores each planning policy only with the scorers of architectures it does not plan against, much as held-out data evaluate a fitted model. It estimates the gain that a policy shows to scorers it does not optimise against and so removes the preference that Section 4.3 measures. The scorer sets of any two planning policies differ in one architecture each. Table 7 repeats Table 1 with every gain measured against MATCHED under the same scorer. Each policy is averaged over a different set of scorers, so a baseline that differs between scorers does not cancel, and the exclusion reorders the two scales differently.

The tables show two features. GREEDY ranks last on all three corpora, by a wide margin. On ASSIST2009, it scores -0.0354, compared with -0.0063 for MATCHED. Items that a model rates lowest are the hardest available items, which most scorers penalise. This artefact makes a difficulty-matched control necessary. Measured against GREEDY rather than against MATCHED, every planning policy appears to have a gain that consists mostly of the gap between GREEDY and MATCHED. The other two controls do not always remain at the bottom. RANDOM ranks fifth on ASSIST2009 and fourth after the correction. On NIPS34, all five planning policies rank above the three controls under the naive average (Table 6).

The exclusion leaves the scores of the controls unchanged by construction, since no control plans against a model. A control changes rank only when a planning policy moves past it.

Table 4: ASSIST2009 leaderboard, $n = 287$ simulated learners. The naive average uses all 10 scorers, and the exclusion estimator drops the two scorers of the architecture each policy plans against. A moved rank is marked.

Policy	Naive gain	Rank	Exclusion gain	Rank	Moved
PLAN-DKT	$+0.0094$	1	$+0.0076$	1	
PLAN-simpleKT	$+0.0028$	2	$+0.0010$	2	
PLAN-DKVMN	$+0.0013$	3	-0.0021	5	•
PLAN-SAKT	0.0000	4	-0.0010	3	•
RANDOM	-0.0020	5	-0.0020	4	•
PLAN-AKT	-0.0036	6	-0.0040	6	
MATCHED	-0.0063	7	-0.0063	7	
GREEDY	-0.0354	8	-0.0354	8	

Table 5: ASSIST2017 leaderboard, $n = 894$ simulated learners. The naive average uses all 10 scorers, and the exclusion estimator drops the two scorers of the architecture each policy plans against. A moved rank is marked.

Policy	Naive gain	Rank	Exclusion gain	Rank	Moved
PLAN-simpleKT	$+0.0256$	1	$+0.0200$	2	•
PLAN-DKT	$+0.0244$	2	$+0.0180$	3	•
PLAN-AKT	$+0.0200$	3	$+0.0208$	1	•
PLAN-DKVMN	$+0.0181$	4	$+0.0178$	4	
PLAN-SAKT	$+0.0144$	5	$+0.0143$	5	
MATCHED	$+0.0131$	6	$+0.0131$	6	
RANDOM	$+0.0120$	7	$+0.0120$	7	
GREEDY	-0.0008	8	-0.0008	8	

Table 6: NIPS34 leaderboard, $n = 881$ simulated learners. The naive average uses all 10 scorers, and the exclusion estimator drops the two scorers of the architecture each policy plans against. A moved rank is marked.

Policy	Naive gain	Rank	Exclusion gain	Rank	Moved
PLAN-DKVMN	-0.0132	1	-0.0173	4	•
PLAN-simpleKT	-0.0135	2	-0.0122	1	•
PLAN-AKT	-0.0154	3	-0.0149	3	
PLAN-SAKT	-0.0158	4	-0.0205	6	•
PLAN-DKT	-0.0169	5	-0.0132	2	•
MATCHED	-0.0198	6	-0.0198	5	•
RANDOM	-0.0232	7	-0.0232	7	
GREEDY	-0.0553	8	-0.0553	8	

Table 7: The top of the leaderboard as in Table 1, with every gain measured against MATCHED under the same scorer before averaging. Planning policies are named after the architecture they plan against.

Corpus	Naive top three	Exclusion top three	Ranks moved
ASSIST2009	DKT > simpleKT > DKVMN	DKT > DKVMN > SAKT	4 of 8
ASSIST2017	simpleKT > DKT > AKT	simpleKT > DKT > AKT	0 of 8
NIPS34	DKVMN > simpleKT > AKT	**simpleKT** > DKVMN > SAKT	4 of 8

B BOTH RUNS OF THE DESIGN

The design is run twice. The first run uses sampling seed 42 and the first seven batches of each corpus, for 789 learners. The run reported throughout uses sampling seed 7 and up to 20 batches (2,062 learners), so it contains the learners of the first run. Changing the sampling seed changes the probe, the item pool and the random draws of every rollout. Both runs use the same fitted models for planning and for scoring. Table 8 compares the two runs, and both will be released.

Table 8: All endpoints from both runs. Agreement gaps are tested with Welch tests over scorer pairs. Pooled self-preference is tested with a one-sided paired test over the 15 policy-corpus cells. Per-corpus entries are ordered ASSIST2009 / ASSIST2017 / NIPS34. Differences are taken before rounding.

	First run	Second run
Resampling floor τ	0.467	0.524
Different architectures τ	0.315	0.335
gap to the floor	0.151 ($p = 0.0053$)	0.189 ($p = 8.6 \times 10^{-5}$)
Same architecture τ	0.438	0.576
gap to different architectures	0.123 ($p = 0.29$)	0.242 ($p = 0.0039$)
Self-preference, pooled	$+1.54$ ($p = 2.8 \times 10^{-5}$)	$+1.57$ ($p = 4.4 \times 10^{-5}$)
cells positive	14 of 15	13 of 15
Ranked among themselves, planners / GREEDY	$+1.21$ / -0.12	$+1.33$ / -0.04
Cells where own architecture sees more gain	9 of 15	13 of 15
Learners	171 / 307 / 311	287 / 894 / 881
Planning gain over MATCHED	$+.0155$ / $+.0054$ / $+.0102$	$+.0083$ / $+.0074$ / $+.0048$
Self-preference	$+1.27$ / $+1.75$ / $+1.60$	$+1.22$ / $+2.52$ / $+0.95$
Winner changed by the correction	NIPS34	ASSIST2017, NIPS34

What holds. Agreement between architectures is below the resampling floor in both runs. In point estimate, the margin is 0.151 in the first run and 0.189 in the second. Self-preference is present in 14 of 15 cells in the first run and 13 of 15 in the second. In both runs, self-preference and the planning gain over MATCHED are positive on every corpus. A GREEDY control built on the same model receives no such preference in either run (-0.12 and -0.04 ranks), whereas the planners

receive $+1.21$ and $+1.33$. The paired difference has $p = 0.0017$ in the first run and $p = 0.0008$ in the second.

What does not. The correction changes the winner on NIPS34 in both runs but on ASSIST2017 only in the second. A reported flip can therefore depend on the run. The architecture gap, the difference between same- and cross-architecture agreement, has the same direction in the first run but passes only the randomisation test there. Per corpus it is -0.029 on ASSIST2009 in the first run and -0.011 on NIPS34 in the second, so it is positive in both runs only on ASSIST2017 (Appendix E). The gap to the floor passes the Welch test in the first run, but its bootstrap interval includes zero (Appendix E). Self-preference in ranks does not track planning gain across corpora in either run. In the first run, ASSIST2009 has the largest gain ($+0.0155$) but the smallest self-preference ($+1.27$). The dose response of Appendix C concerns gain, not ranks. The excess gain of the own-architecture scorers appears in 13 of 15 cells of the second run and in nine of the first.

C SEARCH BUDGET AND THE DOSE RESPONSE

The planner evaluates a fixed number of candidate sequences against its own model. This number is a parameter of the planner, so the amount of optimisation can vary while the policies, learners, scorers, probe and item pool remain fixed. Studies of self-preference in language-model judges admit no such test, because the prose they score is not optimised against the judge by a set amount. The run reported throughout uses 48 candidates. Two further arms use 12 and 192 candidates on all 894 ASSIST2017 learners, with every other setting identical.

Table 9: Self-preference across search budgets, in ranks and in gain. Gains are measured against the difficulty-matched control built on the sequence of each planner, under the two own-architecture scorers (own) and the other eight (other). The excess gain is the difference between the unrounded own and other gains. Rank p values are one-sided.

	In ranks		In gain over own control		
Candidates	Own / other	Self-preference	Own	Other	Excess
12	2.10 / 4.20	$+2.10$ ($p = 0.055$)	$+0.0097$	$+0.0036$	$+0.0061$
48	1.40 / 3.92	$+2.52$ ($p = 0.0024$)	$+0.0164$	$+0.0048$	$+0.0117$
192	1.60 / 4.28	$+2.68$ ($p = 0.0024$)	$+0.0225$	$+0.0054$	$+0.0171$

Self-preference measured in gain increases with the dose (Table 9). Increasing the search budget 16-fold from 12 to 192 candidates raises the gain seen by own-architecture scorers from $+0.0097$ to $+0.0225$, whereas the gain seen by scorers of other architectures rises only from $+0.0036$ to $+0.0054$ ($p = 0.07$). The excess nearly triples, from $+0.0061$ to $+0.0171$. The excess increases significantly at each step in a paired test over learners ($p = 0.0059$ and 0.0078), and both steps remain significant after a Holm correction for the two steps ($p = 0.012$). It is larger at 192 than at 12 candidates for all five planners. Further search yields mostly gain that only the own-architecture scorers can see, which is the signature of optimisation against a learned proxy (Gao et al., 2023; Khalaf et al., 2025).

The dose response is nearly invisible in ranks. At the smallest budget, the own-architecture scorers already place their planner close to the top at a mean rank of 2.10 out of eight, so the rank difference has little room to grow. This difference moves from $+2.10$ to $+2.68$. A paired bootstrap over learners gives the increase of $+0.58$ ranks a 95% interval of $[-0.45, +1.30]$. A rank statistic is the natural way to report a leaderboard and the wrong way to look for a dose response.

Self-preference also persists with less search. At 12 candidates, a quarter of the search the paper reports, the excess gain of $+0.0061$ already differs from zero ($p = 5.0 \times 10^{-5}$). It is a third of that at 192 candidates. In ranks, the own-architecture scorers still place their policy 2.10 positions higher than the other scorers do, although at this budget the difference is not significant ($p = 0.055$). The correction of Equation 2 removes this excess even for such a cheap planner.

The arms share learners and recorded probe and pool items, but use different random draws during scoring. The planner consumes a budget-dependent number of samples, so the generator state entering the rollouts differs between arms. The comparison is therefore unbiased but noisier than a fully

paired one. GREEDY is the one policy that no budget affects, so it isolates this noise. Its sequences are identical across the arms, and its scores differ only because of sampling in scoring ($p \geq 0.88$ between any two arms).

D INSTRUMENT DETAILS

Figure 5 lays out the design, and the paragraphs below give the settings behind it.

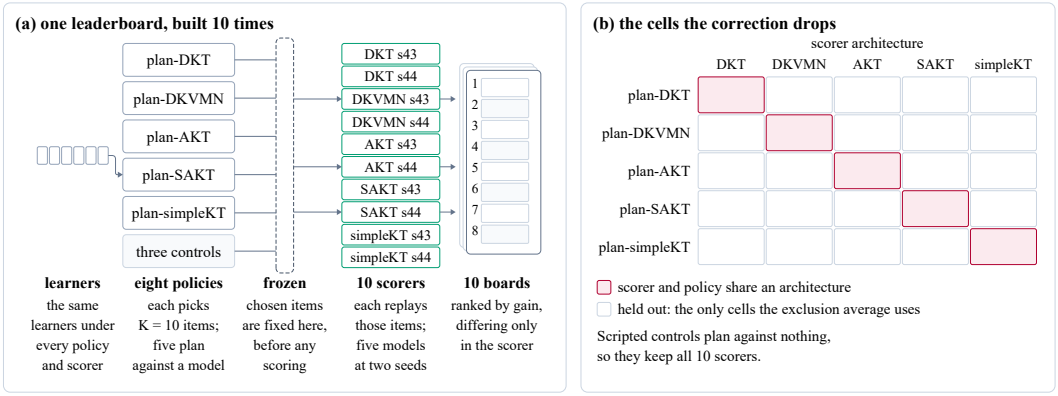

Figure 5: The protocol in one picture. (a) Eight policies select exercises for a learner, five by planning against a fitted knowledge-tracing model and three as scripted controls (GREEDY, MATCHED, RANDOM). The chosen items are frozen before any of the 10 scorers replays the learner over them, so two leaderboards differ only in the scorer and its sampled responses. (b) A policy that plans against an architecture is scored by the two scorers of that architecture, which are fitted at other seeds. Shaded cells mark these scores, and the correction evaluated in Section 4.6 consists entirely of dropping them.

Policies. Each planning policy samples 48 candidate sequences of $K = 10$ items from a pool of 100, drawing each item uniformly and with replacement. It rolls out each candidate once under its own fitted model and keeps the sequence that maximises mean predicted correctness over the probe. During the autoregressive rollout, each sampled response is appended to the history before the next step is predicted. Candidates are evaluated in chunks of 16 to bound memory. All five planning policies share this code path and differ only in the fitted model. Every appended item carries all the concepts it has in the corpus, as in fitting.

Controls. RANDOM draws K items uniformly and with replacement from the pool. GREEDY ranks pool items by the correctness DKT predicts for them and takes the K lowest. MATCHED reproduces the empirical difficulty profile of the sequence planned against DKT. Item difficulty is the fraction of correct responses to an item in the first t_0 interactions of the test split. For each position, MATCHED draws uniformly among the eight pool items closest in difficulty. The per-model versions of GREEDY and MATCHED consist of one pair for each of the five models and are built from the same inputs on a separate random stream that is restored afterwards. Every scorer scores them, but they never enter a leaderboard, and every leaderboard number is identical with or without them.

Probe and pool. Both sets are drawn once per run from the items that learners of the first batch attempt in their first $t_0 = 50$ interactions. They are held fixed across every policy, scorer and learner. The probe of 16 items is disjoint from the pool of 100, so no policy can teach a probe item. Probe items are appended one at a time and independently, so no measurement is conditioned on another. Together the probe and pool hold 116 of the 15,911, 3,162 and 948 items of ASSIST2009, ASSIST2017 and NIPS34. On the two ASSISTments corpora they carry 46 of 123 and 47 of 102 concepts, which account for 70% and 82% of the responses. NIPS34 tags an item with a path in a subject tree, and at its most specific level the probe and pool carry 34 of 53 concepts, which account

for 92% of the responses. The two sets share no item, but 15, 12 and 15 of the 16 probe items share a concept with the pool.

Scorers. The 10 scorers are models of five architectures, each fitted at seeds 43 and 44. The policies plan against the model of each architecture fitted at seed 42. No policy is therefore scored by the parameters it plans against, and self-preference cannot be an artefact of shared parameters. Learning gain is the change in mean predicted probe correctness across the taught block, averaged over learners.

Fitting. The 15 models of a corpus share one split of its students into 70% for training, 10% for validation and 20% for testing. Student sequences are cut into windows of 200 interactions. An item carries up to four concepts (eight on NIPS34). All models use 64-dimensional embeddings and dropout 0.2. DKT has a recurrent state of 200 units and DKVMN has 50 memory slots. The attentive models use four heads, with two layers in simpleKT and an L_2 penalty of 10^{-5} on the item difficulties of AKT. Training uses Adam with a learning rate of 10^{-3}, batches of 64 windows and gradients clipped at norm 5. It stops after 10 epochs without a gain in the area under the receiver operating characteristic curve (AUC) on the validation split or at 100 epochs, and it keeps the best checkpoint.

Table 10: Next-response prediction on the test split, averaged over the two scorers of each architecture. ECE is the expected calibration error over 10 equal-width bins. The test splits hold 50,587, 183,653 and 282,159 responses.

	ASSIST2009		ASSIST2017		NIPS34	
Architecture	AUC	ECE	AUC	ECE	AUC	ECE
DKT	0.756	0.026	0.782	0.017	0.797	0.010
DKVMN	0.753	0.037	0.768	0.026	0.796	0.007
AKT	0.749	0.026	0.767	0.024	0.795	0.010
SAKT	0.742	0.028	0.727	0.024	0.790	0.005
simpleKT	0.746	0.039	0.734	0.015	0.790	0.012

Predictive quality. Table 10 reports next-response prediction on the test split. The mean AUC of the five architectures spans 0.014 on ASSIST2009, 0.055 on ASSIST2017 and 0.007 on NIPS34. Every mean calibration error is below 0.04. The three models of an architecture differ by less than 0.007 AUC, so each policy plans against a model that predicts about as well as the scorers of its architecture. On NIPS34, where the architectures predict almost equally well, two scorers of different architectures agree at $\tau = 0.525$, close to the floor of 0.554 averaged over 200 splits. On the two ASSISTments corpora, leaving out the two architectures that predict worst lowers the agreement among the other three, from $\tau = 0.239$ to 0.048 on ASSIST2009 and from 0.239 to 0.036 on ASSIST2017. Disagreement there persists among the scorers that predict best.

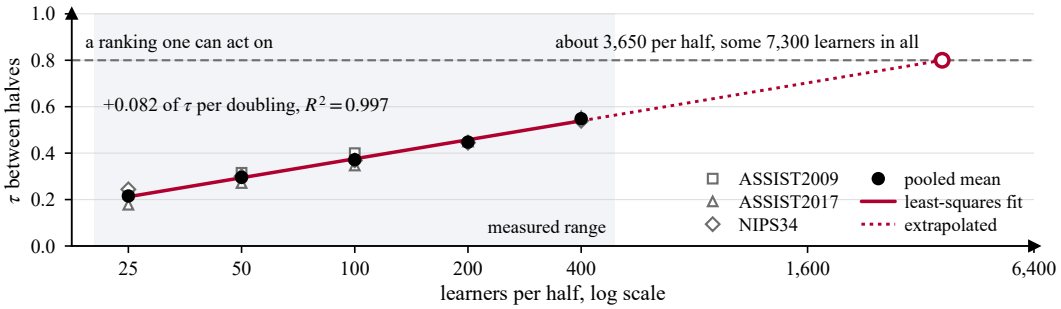

Figure 6: The resampling floor against learners per half, with per-corpus means, the pooled mean and its least-squares fit in $\log_2 m$, where m is the number of learners per half. The dotted part extends the fit to $\tau = 0.8$.

Scale and seeds. A simulated learner starts from a window of up to 200 interactions of one test-split student, and a student with a long history yields several windows. The runs read up to 20 batches of 48 windows and keep those whose first 50 interactions are complete. On ASSIST2009 this keeps 287 of all 876 test windows, from 202 of 769 students. ASSIST2017 keeps 894 of 960 windows, from 304 students. NIPS34 keeps 881 of 960 windows, from 493 students. The 2,062 simulated learners therefore come from 999 students. Drawing the halves of the floor by student instead of by window changes the floor of each corpus by at most 0.002. The run reported throughout uses sampling seed 7, and the first run of Appendix B uses sampling seed 42 and the first seven batches. Learners are shared across all policies and scorers, so every comparison is paired.

Statistics. Rank comparisons use a paired t test over the 15 policy-corpus cells when pooled and over the five planning policies within a corpus. Agreement comparisons use the Welch t test over scorer pairs, and gain comparisons use paired t tests over learners, or over the 15 cells when pooled. Appendix E repeats the first two without assuming independent units. The resampling floor comes from one random split of the learners per corpus, with the policies ranked under each scorer. Over 200 random splits per corpus the floor averages 0.533, slightly above the 0.524 of the single split. In the first run it averages 0.410 against 0.467, so there the single split widens the gap to the floor. Kendall τ is the τ-b statistic over the eight policy gains. Sampling responses in scoring adds noise that Appendix F measures.

Students as units. A student can contribute several windows, so the gain tests of Appendix C and the bootstrap of the gap to the floor are repeated with students as units. With the windows of each student averaged, the excess gain differs from zero at 12 candidates ($p = 0.0076$). It increases at each step ($p = 0.0067$ and 0.0045, or 0.009 for both after the Holm correction). The gain seen by the other architectures still does not rise significantly ($p = 0.20$). With whole students split into halves and resampled within each half, the gap to the floor has a 95% interval of $[+0.045, +0.229]$, against $[+0.048, +0.237]$ with windows as units. In the first run it is $[-0.051, +0.142]$ and still includes zero. Resampling whole students in each corpus gives the pooled self-preference of $+1.57$ ranks a 95% interval of $[+1.11, +1.93]$, and $[+0.78, +1.75]$ in the first run.

Floor against sample size. At each size and for each corpus, the scaling in Section 4.1 draws two disjoint halves of m learners 60 times and averages τ over the 10 scorers (Figure 6). ASSIST2009 is too small for $m = 200$ and 400, so only the other two corpora contribute at these sizes. The fitted floor reaches 0.8 far outside the measured range.

E Inference without the independence assumption

Every rank and agreement p value of Section 4 comes from a t test over dependent units. The 120 cross-architecture scorer pairs are formed from the 10 scorers of each corpus, and the 15 policy-corpus cells share five architectures. Table 11 therefore repeats the headline tests, holding the recorded matrix fixed and randomising the label whose effect the paper claims, or, for the gap to the floor, resampling the learners. Every conclusion stands. Randomisation lowers the p values of the architecture gap and the planners below those of the t tests. It raises those of the self-preference and of the difference between planners and GREEDY, to 0.00008 and 0.0032.

The architecture gap admits an exact test. For each corpus, the 10 scorers have 945 partitions into five unlabelled pairs, one of which is the true architecture partition. The p value is the share of partitions with a gap at least as large as the observed one, and its minimum is 0.0011. With only five same-architecture pairs, a single corpus gives the test little power. Even so, ASSIST2017 attains the minimum and ASSIST2009 gives 0.020. On NIPS34 the gap is -0.011 ($p = 0.62$), so the architecture gap rests on the two ASSISTments corpora. When the three corpora are relabelled independently, only six of 20,000 draws reach the pooled gap. The same five architectures recur in every corpus, so a second null shares one partition across the three corpora. It gives the pooled gap $p = 0.0042$ over 945 partitions.

The floor and cross-architecture agreement are estimated from one learner sample rather than obtained as two labellings of one matrix, so their comparison uses a bootstrap over learners instead of a randomisation. Learners are split into disjoint halves that are resampled separately to keep them dis-

Table 11: The headline rank and agreement tests, repeated by label randomisation or a learner bootstrap instead of a test over units. The first row randomises which scorers count as one architecture. The second row bootstraps learners instead, so its entry is a 95% interval for the gap rather than a p value. The four rank rows randomise which architecture each policy is taken to plan against. The rank t tests are one-sided except for GREEDY, and the agreement tests are two-sided. Differences are taken before rounding.

Quantity	Value	t test	Resampling
Same minus different architecture, τ	$+0.242$	0.0039	0.0003
Floor minus different architecture, τ	$+0.189$	8.6×10^{-5}	$[+0.048, +0.237]$
Self-preference, ranks	$+1.57$	4.4×10^{-5}	0.00008
Planners ranked among themselves	$+1.33$	1.4×10^{-5}	0.00001
GREEDY controls ranked among themselves	-0.04	0.89	0.57
Planners minus GREEDY, ranks	$+1.38$	0.0008	0.0032

joint. Repeating a learner within a half depresses agreement in that half, so the interval understates the floor and is conservative for the gap. Two of 1,000 replicates are at or below zero.

The self-preference test randomises which architecture each planning policy is taken to plan against, over 120 assignments per corpus and 1,728,000 jointly. The pooled null is the convolution of the three per-corpus nulls, so its p value is exact under independent assignments per corpus. With one assignment shared by the three corpora, the observed one gives the largest self-preference of all 120 ($p = 0.0083$, the smallest attainable), whereas GREEDY gets $p = 0.61$. Under the same null, the difference between planners and GREEDY has $p = 0.025$. The tests confirm the self-preference and its difference from the GREEDY control built on the same model, which receives no detectable preference.

The same tests on the first run of Appendix B agree with what it reports. Self-preference there is $+1.54$ ranks at an exact p of 0.00003, against 2.8×10^{-5} from the t test. The planners gain $+1.21$ positions at an exact p of 0.00002 when ranked among themselves. The GREEDY controls gain -0.12. The difference has an exact p of 0.0032, against 0.0017 from the t test. In the bootstrap, the gap to the floor has a mean of $+0.048$, below its point estimate of 0.151, and a 95% interval of $[-0.052, +0.139]$. Of 1,000 replicates, 169 are at or below zero. Per corpus the same-minus-different gaps there are -0.029, $+0.282$ and $+0.114$, so the pooled value of $+0.123$ rests on the two larger corpora. With shared relabelling the first run gives $p = 0.0083$ for self-preference and for its difference from GREEDY, and 0.030 for the architecture gap.

In the first run the two readings disagree on two quantities. The architecture gap of $+0.123$ has $p = 0.017$ under randomisation and 0.29 under the Welch test, so the first run detects it once the dependence is respected. The gap to the floor passes the Welch test ($p = 0.0053$), but its bootstrap interval includes zero, so the first run does not establish it. The abstract, the introduction and Section 4.2 report the bootstrap interval of the gap to the floor, and the abstract gives the larger of the two exact p values of self-preference. Sections 4.3 and 4.4 and the tables of Appendices A and B keep the t tests.

None of the tests addresses whether the simulator is right. They establish only that the differences reported here are properties of the recorded matrix rather than of an optimistic error model. Section 7 leaves open the question of whether a fitted knowledge-tracing model orders tutoring policies as real learners do, and no resampling inside this design answers it. Likewise, no relabelling can supply the two comparisons the design lacks, a dose response on more than one corpus and a scorer whose validity is established outside these data.

F THE SCORING ROLLOUT AS A SOURCE OF DISAGREEMENT

A scorer replays a learner over the taught items and appends sampled responses, so a leaderboard reflects the randomness of these responses as well as the draw of learners. The search-budget arms of Appendix C separate the two sources. The arms share their learners, probe and pool, and a control that queries a model without searching is built identically in every arm. For such controls, two arms differ only in the scoring draws. The five per-model versions of GREEDY are scored on a random

stream that is restored after them, so their draws repeat in every arm and they are left out. Over GREEDY, 10 scorers and three arm pairs, the rollout has a standard deviation of 0.0709 per learner and no detectable shift between arms ($p = 0.99$).

Table 12: The rollout noise of each scoring architecture on ASSIST2017, measured in the search-budget arms, and the agreement between its two seeds. The standard deviation is per learner, on a gain scale where neighbouring policies under one scorer differ by 0.0049 on average.

Scoring architecture	Rollout s.d.	τ, its two seeds
SAKT	0.0193	0.643
DKVMN	0.0268	0.714
simpleKT	0.0503	0.857
AKT	0.0833	0.214
DKT	0.1206	0.929

That size is not negligible. Under one scorer, the average gap in gain between two neighbouring policies is 0.0049. On a half sample, the rollout noise in a policy mean is 0.69 times that gap. A scorer on a half sample agrees with itself at $\tau = 0.703$ when two independent draws of noise with the rollout s.d. of its architecture are applied to the recorded means. Redrawing the learners as well gives a floor of 0.586 averaged over 60 splits, whereas two scorers of different architecture reach 0.239. Under this approximation, the rollout alone costs 72% of the agreement lost at the floor, so most of that loss comes from the scorer replaying each learner once rather than from the draw of learners.

Both parts of the lost agreement fall as the sample grows, so neither is irreducible. The costs of reducing them differ. Averaging several rollouts per learner reduces variance at a computational cost but requires no new learners. It therefore remains available once a corpus is exhausted. The run reported throughout draws 287, 894 and 881 learners, compared with the 7,300 extrapolated in Section 4.1. Unlike the changes of Section 5, averaging rollouts needs new scoring runs. Rollout noise also bears on equal weighting, since the naive average treats a scorer with rollout noise 0.0193 and one with 0.1206 alike.

Weighting the panel by precision is not a solution. The measured noise gives the least noisy architecture 39 times the weight of the noisiest, so an average weighted by $1/\sigma^2$, the inverse square of the rollout s.d., puts 87% of its weight on SAKT and DKVMN. Compared with the naive average, this weighting moves three of eight positions on ASSIST2009, four on ASSIST2017 and two on NIPS34, with τ of 0.857, 0.643 and 0.786. It changes the reported winner on ASSIST2017. The weights come from the one corpus where the arms allow them to be measured and are applied to all three, so the weighting is an illustration rather than a proposal. It shows that the naive average is a convention and that a defensible change of convention reorders the top of the leaderboard.

Table 12 also bears on the disagreement between architectures. Across the five architectures, rollout noise does not track how well the two seeds agree. DKT is the noisiest and its seeds agree best, at 0.929, whereas AKT is the second noisiest and its seeds agree worst, at 0.214. Even the three least noisy architectures agree with each other at only 0.357, far below the floor of 0.586, whereas the two noisiest agree at 0.071. Noise alone does not explain why the architectures disagree.

The rollout standard deviation of 0.0709 is the root of a mean variance. Taken the same way over policies and scorers, gain varies across learners by 0.1287 on ASSIST2017. The other two corpora give 0.1062 and 0.0917. The rollout therefore accounts for about 30% of the variance over all policies and 33% of that of GREEDY, on which it is measured. It accounts for most of the agreement lost at the floor, because learner effects shared by all eight policies cancel in a ranking and a rank statistic saturates. The estimate uses GREEDY, the only policy whose sequences no search budget changes and whose scoring draws differ between arms. A planned sequence may carry more or less rollout noise. Two rollouts per learner separate the two components directly but double the scoring computation.