COVERSIM: COVERAGE-GUIDED USER SIMULATION FOR TASK-ORIENTED DIALOGUE

Anonymous authors
Paper under double-blind review

ABSTRACT

Task-oriented dialogue systems are increasingly evaluated with LLM-simulated users, enabling scalable testing across diverse user goals and profiles. Yet current approaches often reward human-likeness without verifying whether the simulator actually follows the user condition it is meant to enact. As a result, a conversation can look realistic while failing to test the intended behavior of the system. In this work, we ask what makes a user simulator good for testing task-oriented dialogue systems. We formalize simulation-based evaluation as testing, where each goal–profile pair is a test input, but it counts toward coverage only if the simulator faithfully executes it. This turns simulator improvement into a coverage-optimization problem: reduce inconclusive executions so more goal–profile conditions become testable. We then optimize user simulators directly for this objective using evolutionary prompt optimization, producing simulators that are better at testing the system, not merely imitating the user. We instantiate our automated testing framework using the data and user profiles from SimulatorArena and show that our simulator optimization strategy improves profile-following accuracy by up to 9.5% and more than doubles validated test coverage. We then show that standard TOD evaluation hides both agent failures and simulator failures, yielding a misleading picture of agent performance.

1 INTRODUCTION

Task-oriented dialogue (TOD) systems help users accomplish concrete goals through interaction, such as changing a booking, solving a tutoring problem, or producing a document. These systems are increasingly evaluated by replacing human users with LLM-based user simulators, which are given a task goal and a user profile and then interact with the agent under test Terragni et al. (2023); Sekulić et al. (2024); Mehri (2025); Dou et al. (2025). Existing work has largely focused on making these simulators more human-like. However, users are diverse, and many useful test conditions are precisely the ones that are difficult to simulate: users may be confused, terse, verbose, impatient, uncertain, or uncooperative. We argue that for evaluation, the central requirement is not just realism, but faithfulness to the assigned goal and profile. If a simulator cannot faithfully enact the assigned user profile, then the resulting conversation does not actually test the system under that condition.

This aspect is largely overlooked by current simulation-based benchmarks. Typically, the simulator is designed with an objective independent of the final evaluation, such as human-likeness or agreement with human behavior, with success criteria being the overall task completion rate over all simulations. This hides whether the intended user-profile conditions were actually exercised. Instead, we frame TOD evaluation as testing: each goal–profile pair is a test input, the user simulator–agent conversation is the execution trace, and the trace is valid only if the simulator follows both the assigned goal and profile. Test executions where the simulator deviates from its profile or goal are marked as *inconclusive*. These inconclusive executions reveal the testing coverage gaps: user-profile conditions for which we do not yet have reliable evidence about the TOD system's behavior.

From this testing perspective, a good simulator is one that minimizes these coverage gaps. This gives a principled design objective for user simulation: maximize validated test coverage. To achieve this, we optimize the simulator to reduce the number of inconclusive executions. We implement this with evolutionary prompt optimization, a black-box procedure that is broadly applicable to state-of-the-art proprietary LLMs. Unlike objectives that merely produce better user imitators, our objective

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.

is testing-guided: it produces simulators that are better at exercising specified user conditions and therefore better at testing TOD systems.

We instantiate our coverage-aware testing framework using human-annotated conversations and user profiles from SimulatorArena Dou et al. (2025), covering math tutoring and document creation tasks. We show that existing simulators exhibit large coverage gaps, making standard TOD evaluations incomplete and potentially misleading. We then show that optimizing the user simulator for test coverage produces more faithful simulations and substantially improves validated coverage. These coverage-optimized simulators change relative agent rankings and establish a principled approach to simulator design: optimize simulators for testing directly. To summarize, our contributions are:

- A coverage-aware testing view of simulation-based TOD evaluation, where only faithful goal–profile executions count as real tests.
- COVERSIM: a coverage-guided simulator optimization method that reduces inconclusive executions, thereby expanding valid test coverage and yielding more informative evaluation feedback on TOD agents.
- Stronger simulator-based evaluation on SimulatorArena, with higher profile faithfulness, broader validated coverage, and newly exposed agent failures.

2 RELATED WORK

LLM-based user simulation. Recent work increasingly uses LLMs to simulate users in dialogue systems. In task-oriented dialogue, simulators are prompted from goals and examples Terragni et al. (2023), fine-tuned for domain-aware behavior Sekulić et al. (2024), or augmented with explicit goal-state tracking Mehri (2025), and have also been deployed in interactive benchmarks for tool-using agents Yao et al. (2024); Dou et al. (2025). Broader dialogue work conditions simulators on personas, profiles, or inferred user traits to produce more diverse and human-like behavior Kong et al. (2024); Shao et al. (2023); Wang et al. (2025c;a); Ahmad et al. (2025); Zhang et al. (2024); Wang et al. (2023; 2025b); Li et al. (2024). Similar ideas also appear in math tutoring, where simulated students support tutor evaluation and training Macina et al. (2023); Markel et al. (2023). Unlike work that primarily optimizes simulators for realism, consistency, or human-likeness, we optimize them for validated test coverage: the ability to faithfully execute assigned goal–profile conditions.

Evaluating agents and user simulators. Most simulation-based evaluations rely on aggregate measures such as task completion, goal success, profile consistency, human-likeness, or agreement with human judgments Sekulić et al. (2024); Mehri (2025); Wang et al. (2025a). SimulatorArena Dou et al. (2025) is closest to our setting: it studies whether LLM-based simulators are reliable proxies for real users by measuring message realism and alignment between assistant ratings under simulated and human users. Unlike aggregate benchmark evaluation, our framework treats each goal–profile pair as a test input and counts it toward coverage only when the simulator faithfully follows the assigned goal and profile.

LLM-as-judge and prompt optimization. LLM-based evaluation commonly relies on LLM-as-judge prompts Zheng et al. (2023) to score task success, profile consistency, realism, or response quality, but these prompts are sensitive to manual design choices Sclar et al. (2024). DSPy Khattab et al. (2023); Opsahl-Ong et al. (2024) expresses LLM pipelines as optimizable programs, while GEPA Agrawal et al. (2025) and related methods Pryzant et al. (2023); Yuksekgonul et al. (2024) optimize prompts using task feedback. Unlike prior uses of prompt optimization that mainly improve judge or pipeline accuracy, we use oracle verdicts as the optimization signal for the user simulator itself, reducing inconclusive executions and increasing validated coverage.

3 THE TESTTOD TESTING FRAMEWORK

Simulation-based evaluation is only informative when the simulator actually executes the intended test condition. If we want to evaluate an agent under a particular user goal and profile, the simulator must faithfully realize both. For example, if a profile specifies a confused user who should repeatedly

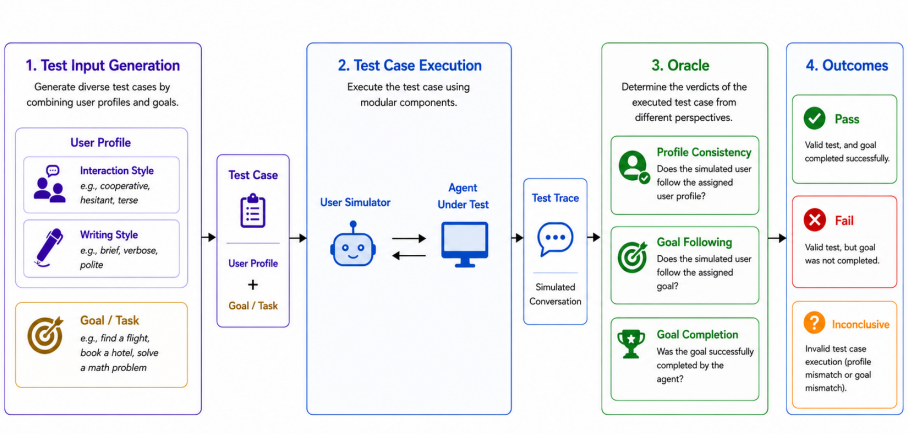


Figure 1: Illustration of `TestTOD`. The test input generation module constructs a test case by pairing a user profile with a task goal. The test case is then executed by a user simulator interacting with the agent under test, producing a simulated conversation trace. The oracle analyzes this trace from three perspectives: whether the simulated user followed the assigned profile, whether the user followed the assigned goal, and whether the agent completed the goal. Based on these analyses, each run is assigned one of three outcomes: *Pass*, *Fail*, or *Inconclusive*, separating agent failures from invalid test executions.

seek clarification, but the simulator instead accepts the first answer, that condition has not been tested. We therefore cast TOD evaluation as a testing problem: each goal–profile pair defines a test input, and the resulting conversation is a test execution. An execution is considered valid only when the simulator follows both the assigned goal and profile; otherwise, it is marked Inconclusive.

As shown in Figure 1, `TestTOD` turns this idea into a simple testing pipeline. Each goal–profile pair is executed through a user-simulator–agent interaction, producing a dialogue trace. An oracle then checks whether the simulator followed the assigned goal and profile, and whether the agent completed the task. Valid executions are labeled *Pass* or *Fail*; executions where the simulator violates the goal or profile are labeled *Inconclusive*. These inconclusive cases reveal gaps in test coverage and provide a direct target for simulator improvement: reducing them allows more intended user conditions to be validly tested.

3.1 Test Inputs and Dialogue Traces

Each test in `TestTOD` is defined by a user goal and a target user profile. In this paper, we instantiate these inputs from human-annotated SimulatorArena conversations, preserving realistic combinations of tasks and user characteristics while making the intended test conditions explicit.

Figure 1 illustrates `TestTOD`, our coverage-aware testing framework. Given a test input, an LLM-based user simulator interacts with the agent under test in a multi-turn dialogue, producing a dialogue trace. The trace is then validated to determine whether the simulator followed the assigned goal and profile, and whether the agent completed the task. Only traces with faithful simulator behavior count toward validated coverage. For these valid tests, the agent receives either a *Pass* or *Fail* verdict. Traces where the simulator drifts from the assigned goal or profile are marked *Inconclusive*: they do not provide reliable evidence about the agent, but instead identify parts of the intended test space that the simulator failed to exercise.

3.2 Automated Validation with Learned Oracles

Given a dialogue trace, the `TestTOD` oracle determines whether the test was validly executed and, if so, whether the agent completed the task. It consists of three analyzers: *goal following*, *profile consistency*, and *task completion*. If either simulator-side check fails, the execution is marked *Inconclusive*; otherwise, the task-completion analyzer assigns *Pass* or *Fail*.

Each analyzer is implemented as an LLM-based binary judge. Since naive prompting can produce unreliable judges, we optimize these prompts from human-annotated interaction data using GEPA (Agrawal et al., 2025). The resulting verdicts are then used both for test validation and as the coverage signal for simulator optimization.

3.3 Oracle Prompt Optimization

Accurate oracle verdicts are a prerequisite for `TestTOD`: they determine which executions are valid and provide the coverage signal for improving the simulator. We therefore optimize the prompts of the individual analyzers using labeled dialogue traces from the SimulatorArena benchmark Dou et al. (2025), where each trace is annotated with the task goal and a user profile describing both interaction style and writing style.

During optimization, each analyzer is run on labeled examples and scored against ground truth. The optimizer then uses the observed mistakes to propose improved instructions. We instantiate this procedure with GEPA (Agrawal et al., 2025), following prior work on automatic prompt optimization (Pryzant et al., 2023; Yuksekgonul et al., 2024).

Task-completion analyzer. For task completion, the analyzer predicts whether the agent successfully completed the target task. We optimize this as binary classification over `correct` and `not_correct`. Positive (`correct`) and negative (`not_correct`) examples are drawn directly from the existing labeled conversations in SimulatorArena. For a minibatch B, a candidate prompt is scored by mean accuracy:

$$S_{\text{task}} = \frac{1}{|B|} \sum_{i \in B} \mathbb{1}[\hat{y}^{(i)} = y^{(i)}].$$

This reward directly measures agreement with the task-completion label.

Profile-consistency analyzers. The analyzer checks whether the simulated user's behavior matches the assigned profile. The annotated data provide positive examples: each dialogue is annotated with the user profile. To create negative examples, we rewrite each profile feature to express the opposite behavior, producing dialogues with content contradicting its user profile.

Because each profile contains several features, we score consistency at the feature level:

$$S_{\text{profile}} = \frac{1}{|B|} \sum_{i \in B} \frac{1}{N_i} \sum_{j=1}^{N_i} \mathbb{1}[\hat{z}_j^{(i)} = z_j^{(i)}],$$

where N_i is the number of profile features, $\hat{z}_j^{(i)}$ is the predicted label, and $z_j^{(i)}$ is the target label. Original features are labeled `coherent`, while inverted features are labeled `not_coherent`. We use the same objective for interaction style and writing style.

During optimization, feedback is generated for each incorrect feature prediction. LLM judges identify the type of error and suggest a general instruction fix. GEPA uses this feedback together with the scalar score to improve the analyzer and reduce both missed profile violations and incorrect rejections of valid behavior.

Goal-following analyzer. For goal following, we treat the problem as failure attribution: if a task fails because of the user rather than the agent, it is counted as a goal-following violation. We annotate the failure cause of each unsuccessful conversation in SimulatorArena and score prompts by mean accuracy, as in the task-completion analyzer, with labels `user` and `agent`.

4 CoverSim: Coverage-Guided Simulator Optimization

The previous section showed how `TestTOD` distinguishes valid tests from inconclusive ones. We now use the same validity signal to improve the simulator itself. Inconclusive executions reveal where the simulator fails to realize the intended test condition, and these failures provide direct feedback for optimization. Our objective is to make the simulator follow the assigned profile more reliably so that more test cases become valid. As shown in Figure 2, we treat the simulator LLM as a black box and optimize its prompt using evolutionary prompt optimization Agrawal et al. (2025).

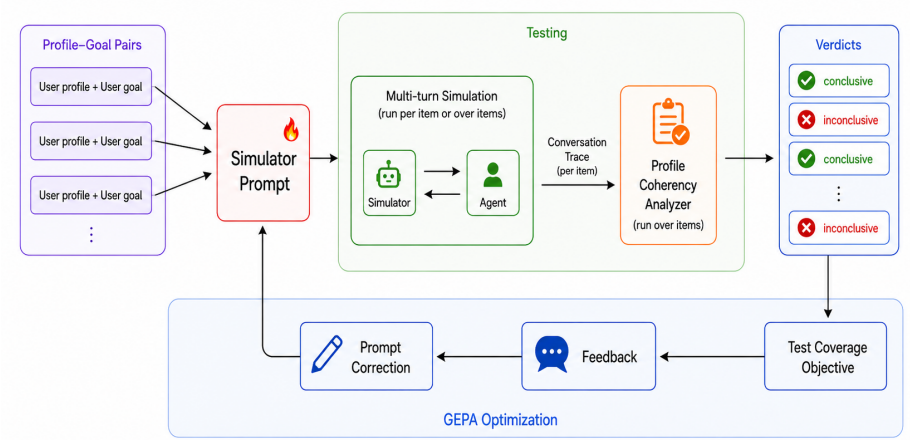


Figure 2: **CoverSim: Coverage-guided simulator optimization.** The simulator prompt is optimized offline to maximize validated test coverage. Each iteration runs TestTOD on a batch of profile–goal pairs, computes coverage from the resulting verdicts, and uses GEPA to revise the prompt. The optimized prompt is then used for user simulation in the testing framework.

4.1 Optimization Objective

The optimization objective follows directly from our notion of test validity: a simulator is better when it realizes the assigned profile more faithfully. For each generated dialogue, the fixed profile-consistency analyzers evaluate every profile feature as `coherent` or `incoherent`. Each `incoherent` verdict indicates a profile-following error and may make the test *Inconclusive*.

For a batch B, let N_i be the number of profile features for example i, and let $\hat{z}_j^{(i)}$ be the analyzer verdict for feature j. We minimize the average number of incoherent profile verdicts:

$$L_{\text{sim}} = \frac{1}{|B|} \sum_{i \in B} \frac{1}{N_i} \sum_{j=1}^{N_i} \mathbb{1}[\hat{z}_j^{(i)} = \texttt{incoherent}].$$

Equivalently, the simulator score is $S_{\text{sim}} = 1 - L_{\text{sim}}$. This objective directly targets profile violations that reduce validated coverage.

4.2 Coverage-Guided Prompt Optimization

The objective above gives us a principled way to optimize an LLM-based user simulator for test coverage. While the objective is compatible with many optimization mechanisms, we instantiate it with prompt optimization using GEPA Agrawal et al. (2025). This is a practical, resource-efficient choice: it leaves model weights unchanged, requires no training infrastructure, and applies to both open- and closed-source LLMs.

At each optimization step, we sample a batch of profile-conditioned test cases, run multi-turn simulations with the current prompt, and apply the profile-consistency analyzers to the resulting traces. The analyzer feedback identifies which profile features were violated and provides evidence from the conversation. Feedback is kept separate by profile component, such as interaction style and writing style. GEPA uses this component-specific feedback to add useful calibration hints for simulation, while the main simulator instructions remain fixed. Candidate prompts are evaluated on held-out profiles and accepted only when they reduce L_{sim} there. Thus, coverage gaps become optimization feedback: the same oracle that marks tests as inconclusive is used to guide improvement of the simulator prompt.

5 Experiments

To evaluate `TestTOD`, we study three main questions: (1) To what extent does coverage-guided simulator optimization improve profile faithfulness and validated test coverage? (2) To what extent

do inconclusive executions affect the validity and coverage of simulation-based TOD evaluation? (3) Does optimizing simulators for validated coverage improve the effectiveness of TOD agent testing and expose additional agent weaknesses?

5.1 Experimental Setup

All experiments use API-based LLM agents, user simulators, and analyzers.

Datasets and Domains. We instantiate `TestTOD` on `SimulatorArena` Dou et al. (2025), using its task goals, human-annotated conversations ($\sim$ 400 per domain), and structured user profiles. We evaluate two domains: *Math Tutoring* and *Document Creation*, with interaction-style and writing-style profiles. Interaction styles define how the simulated user interacts with the TOD agents (e.g., actively engaging, frequently seeking guidance, and depth of inquiry), and writing styles describe the surface-form behaviours of the user (e.g., spelling consistency, sentence complexity, and punctuation usage). Each test pairs a goal with a target profile; the simulator interacts with the TOD agent, and the resulting trace is analyzed by `TestTOD`. Our evaluation uses 1800 simulated conversations to assess the effectiveness of `TestTOD` for detecting inconclusive executions. The experiments consume around 800 US dollars, a simulation time of 6 hours, and an evaluation time of 4 hours across the two evaluated domains.

TOD Agents. For the math tutoring task, we evaluate multiple agents, including `deepseek-flash (deepseek-V4.1-flash)`, `deepseek-v4-pro` DeepSeek-AI (2026), `gpt-5-mini`, `gpt-5` OpenAI (2025), `grok-4.3` xAI (2026), and `claude-sonnet-4-6` Anthropic (2026); for the document creation task, we evaluate `deepseek-v4-flash`, `gpt-5-mini`, and `grok-4.3`. The user simulator is fixed when evaluating the agents on each task: `deepseek-flash` for math tutoring and `deepseek-v4-flash` for document creation. Unless otherwise specified, each agent–simulator configuration is evaluated on 100 held-out profile–goal pairs, with identical test inputs shared across agents.

User Simulators. For math tutoring, we compare the `TestTOD`-optimized simulator, initialized from the original `SimulatorArena` prompt and refined using the coverage-guided procedure in Section 4, against four baselines:

1. **Original**: the `SimulatorArena` prompt, which conditions on the target profile, math problem, and dialogue history to generate the next student utterance. 2. **Self-refinement**: first generates a student utterance without the profile, then rewrites it to match the target profile. 3. **Secondary corrector**: inspired by Self-Refine (Madaan et al., 2023), adds a second LLM call that checks the generated utterance against each profile feature and rewrites only the violated parts. 4. **USP**: a weight-trained simulator following Wang et al. (2025a), using `Llama-3-8B` retrained on our math-tutoring data. Conditional SFT predicts the next user utterance from the profile and dialogue history, while RLCC further optimizes profile consistency. We report SFT and SFT+RLCC separately.

For prompting-based methods, we evaluate `deepseek-flash`, `grok-4.3`, and `claude-haiku-4-5` Anthropic (2025). For document creation, we mainly compare `TestTOD` against the original simulator; full settings and results are provided in the Appendix.

Optimization Setup. At each optimization step, the simulator runs a batch of profile–goal pairs, fixed profile-consistency analyzers identify profile violations, and GEPA updates the simulator calibration notes. Candidate prompts are accepted only when they improve held-out profile coherence. When running **CoverSim** for both domains, we select 50 profiles not in the test set for TOD agents and split them into a training set of 30 profiles and a validation set of 20 profiles to apply an optimization of 15 steps. We retrieve feedback in batches of 5 at each step.

Oracle Optimization Data For oracle training and evaluation, we use a mix of real annotated conversations and synthetically augmented ones, where the user behavior deviates from the specified goal or user profile. These examples provide positive and negative examples for oracle training and are filtered with an NLI-based contradiction verifier. We use nli-deberta-v3-large, a DeBERTa-v3-large model (He et al., 2023) fine-tuned for natural language inference for NLI verification. To

ensure strong contradiction for clear profile incoherence recognition, we use strong contradiction settings to generate negative profiles.

Metrics. For analyzers, we report held-out classification accuracy. For simulator faithfulness, we report the average profile coherence rate (i.e., the average over all the profile internal coherence rates, computed as $N_{feat_coherent}/N_{feat_total}$) and the test coverage rate (i.e., the proportion of conversations in which oracles did not flag inconsistencies). For agent testing, we report three metrics: naive task accuracy over all conversations, task accuracy on covered test cases, and the number of covered test cases, where a test case is considered covered if it is not inconclusive.

Table 1: Math tutoring results over 100 conversations per simulator. (+`CoverSim`) denotes the optimized prompt. Bold marks the best value in each column. Profile coherence is the mean fraction of coherent profile features, reported for the full profile, interaction style (IS), and writing style (WS). Test coverage is the percentage of fully coherent profiles: every interaction-style feature (IS), every writing-style feature (WS), or all features together (Full).

	Profile Coherence Rate (%) ↑			Test Coverage (%) ↑		
Model	Full	IS	WS	Full	IS	WS
Original simulators Dou et al. (2025)						
`deepseek-flash`	84.1	77.4	93.1	10.0	16.0	50.0
`grok-4.3`	86.9	83.6	91.3	12.0	30.0	44.0
`claude-haiku-4-5`	76.1	68.3	86.6	6.0	15.0	28.0
Self-refinement simulators Dou et al. (2025)						
`deepseek-flash`	79.2	70.2	91.2	9.0	21.0	46.0
`grok-4.3`	84.0	77.8	92.3	10.0	20.0	49.0
`claude-haiku-4-5`	74.9	68.4	83.5	7.0	19.0	21.0
Secondary-corrector simulators Madaan et al. (2023)						
`deepseek-flash`	85.4	78.3	**94.8**	14.0	26.0	55.0
`grok-4.3`	90.1	88.4	92.4	18.0	41.0	48.0
`claude-haiku-4-5`	78.0	69.8	89.0	8.0	17.0	31.0
Fine-tuned simulator (USP) Wang et al. (2025a)						
`USP` (SFT)	87.9	88.6	86.9	14.0	41.0	25.0
`USP` (SFT+RLCC)	88.8	88.2	89.5	12.0	42.0	32.0
CoverSim-optimized simulators						
`deepseek-flash` (+`CoverSim`)	91.2	88.7	94.7	**30.0**	45.0	**59.0**
`grok-4.3` (+`CoverSim`)	**91.9**	**91.2**	92.8	28.0	**50.0**	49.0
`claude-haiku-4-5` (+`CoverSim`)	85.6	82.6	89.8	13.0	31.0	37.0

5.2 Effectiveness of coverage-guided simulator optimization

The profile faithfulness results for CoverSim and the compared baselines are reported in Table 1. Based on the profile coherence rates, we observe that the refinement simulators perform the worst, even worse than the original simulators, suggesting that the profile should be passed directly into the user utterance generation call. The secondary correctors improve both test coverage and profile coherence rates by performing a second verification against the user profile. However, CoverSim outperforms all of them, achieving the highest profile coherence rate (91.9% with grok-4.3) and the highest test coverage (30 fully coherent simulations with deepseek-flash), as GEPA optimization provides useful calibration hints that improve the user simulation. Comparing CoverSim with the fine-tuned simulator (USP), we observe that although USP optimizes profile consistency through

Table 2: Agent accuracy on the math tutoring task under the same `deepseek-flash` user simulator, before and after `CoverSim` optimization. Acc@All is computed over all executions. Acc@Covered is computed only over covered executions: those in which the simulated user is fully coherent with the target profile, and in which the outcome is not attributable to a fault of the simulated user. Coverage is the percentage of executions that are not inconclusive.

	Original Simulator			`CoverSim`-Optimized Simulator		
Agent	Acc@All	Acc@Covered	Coverage	Acc@All	Acc@Covered	Coverage
`deepseek-flash`	**100%**	**100%**	10%	**94%**	**96%**	27%
`gpt-5-mini`	95%	95%	**22%**	71%	81%	**31%**
`gpt-5`	98%	95%	20%	85%	93%	28%
`grok-4.3`	96%	95%	20%	81%	93%	27%
`deepseek-v4-pro`	**100%**	**100%**	16%	84%	88%	24%
`claude-sonnet-4-6`	94%	94%	16%	74%	80%	25%

reinforcement learning, its profile coherence and test coverage are no better than those obtained by direct prompt optimization with GEPA.

These results show that CoverSim consistently improves simulator faithfulness and outperforms every compared baselines. Across all LLMs, profile coherence increases after optimization, with the strongest gains on interaction style. More importantly, these improvements translate into higher validated test coverage: full-profile coverage at least doubles for every simulator, showing that fewer intended tests are inconsistent.

5.3 Impact of inconclusive executions on TOD evaluation

The results for the TOD agent evaluation are reported in Table 2. They show that standard Acc@All and Acc@Covered can be misleading when many executions are inconclusive. With the original `deepseek-flash` simulator, both accuracies are nearly perfect across all models. However, this is due to the low profile coherence rate and the small number of conclusive test cases, which make the task much easier (e.g., deep-digging and hard students are not correctly simulated) and the comparison unreliable.

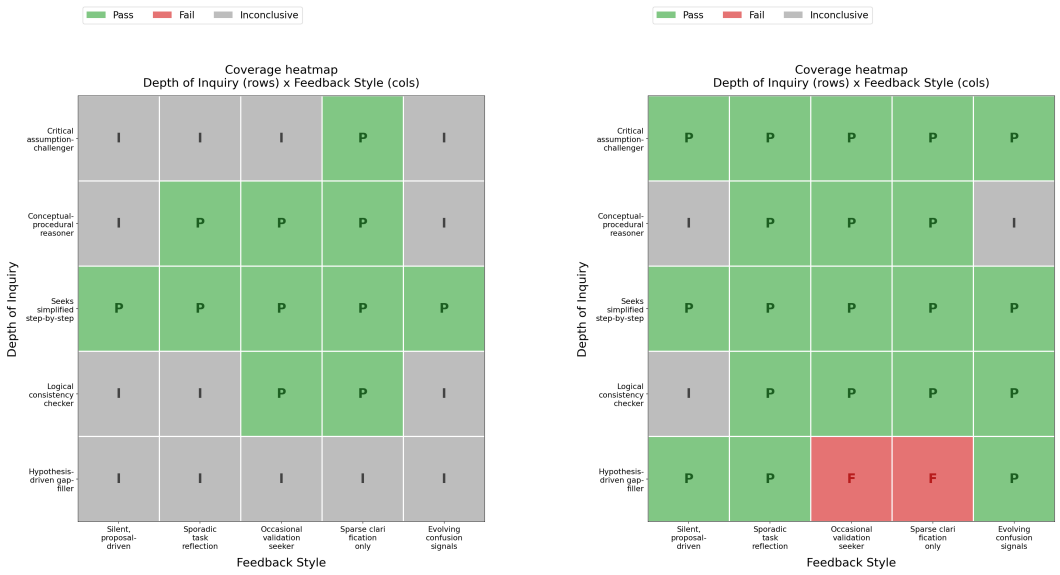
After `CoverSim` optimization, Acc@All and Acc@Covered become substantially more meaningful, as there are more conclusive cases. By increasing both the number of fully profile-consistent cases and the profile coherence rate on the remaining test cases, the optimized simulator produces more behaviorally demanding covered tests and exposes clear differences between agents. For example, with the original simulator, gpt-5 and gpt-5-mini perform almost identically; however, with the more coherent simulations provided by CoverSim, gpt-5-mini clearly underperforms gpt-5 in terms of Acc@Covered. This shows that inconclusive executions can mask real agent failures and that coverage-optimized simulation provides a more reliable evaluation signal.

5.4 Toward Full Coverage Testing

We evaluate whether simulator optimization improves validated coverage over a controlled profile space. We use two interaction-style dimensions, *feedback style* and *depth of inquiry*, with five values each, yielding 25 profile combinations. Each cell is executed once and labeled as *Pass*, *Fail*, or *Inconclusive*. Figure 3 shows that the initial simulator covers only 11/25 cells, or 44% validated coverage. We then re-run the 14 inconclusive pairs with the optimized simulator, increasing validated coverage to 22/25 cells, or 88%. This also uncovers new agent failures: two pairs that were previously inconclusive become valid *Fail* cases. Thus, coverage-guided optimization not only moves the test suite closer to full coverage, but also exposes failures that were hidden by simulator drift.

5.5 Effects of Optimizing Oracle Prompts

Table 3 evaluates whether GEPA optimization improves the oracle prompts used by our testing framework. The optimized prompts improve every analyzer across both domains. For profile-coherence checks, the original prompts are already strong, but optimization still yields consistent gains, increasing interaction- and writing-style accuracy by up to 1.75 points. The largest improve-



(a) Before CoverSim optimization: 11/25 cells validated.

(b) After CoverSim optimization: 22/25 cells validated.

Figure 3: Coverage over a 5×5 profile grid before and after CoverSim optimization. Re-running the initially inconclusive profile pairs with the optimized simulator increases validated coverage from 11/25 to 22/25 cells and uncovers new valid agent failures.

Table 3: Accuracy of original and GEPA-optimized prompts for profile-coherence and goal-following analyzers. GEPA improves all analyzers, with the largest gain for task completion on document creation.

Task	Analyzer	Original Prompt	Optimized Prompt
Math Tutoring	Interaction Style	96.25	**97.81**
	Writing Style	95.75	**97.50**
	Task completion	92.00	**94.00**
Document Creation	Interaction Style	96.25	**97.58**
	Writing Style	90.53	**94.87**
	Task completion	44.00	**76.00**

ment is in task completion for document creation, where accuracy rises from 44.00 to 76.00 due to subjectivity in this task. These results show that the oracle layer can be made tighter through prompt optimization. This is important for testing: more accurate oracles reduce noisy validity judgments, making Pass, Fail, and Inconclusive outcomes more reliable. In turn, coverage estimates become more trustworthy, since a test is counted as covered only when the simulator truly follows the assigned goal and profile.

6 Conclusion

We introduced `CoverSim`, a coverage-guided framework for evaluating TOD systems with LLM-based user simulators. `CoverSim` treats simulation as testing: a goal–profile condition counts as evaluated only when the simulator faithfully executes it. Our results show that standard simulation-based evaluation overestimates effective coverage, as many intended tests become inconclusive due to simulator drift. Optimizing simulators for validated coverage improves profile faithfulness, increases test coverage, and exposes agent failures hidden by aggregate metrics. These findings suggest that reliable TOD evaluation requires not only realistic simulated users, but simulators that can faithfully exercise the conditions they are meant to test.

AI USE STATEMENT

We primarily used generative AI tools as coding assistants for execution and debugging experimental code and producing visualizations. The AI-assisted code was tested for correctness by one author. Generative AI tools were also used to improve the clarity and readability of author-written text and, to a more limited extent, to provide feedback on the experimental methodology. The AI-assisted outputs were reviewed and verified by at least one author.

Generative AI tools were not used to determine the research questions, develop the central method, formulate or refine hypotheses, or interpret the experimental results. Uses involving synthetic data generation, theoretical or mathematical claims and proofs, translation, and qualitative or thematic analysis were not applicable to this work. We take responsibility for the final content of this work, including text, code, and other artifacts produced with the aid of generative AI.

REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025. URL https://arxiv.org/abs/2507.19457.

Adnan Ahmad, Stefan Hillmann, and Sebastian Möller. Simulating user diversity in task-oriented dialogue systems using large language models. *arXiv preprint arXiv:2502.12813*, 2025. URL https://arxiv.org/abs/2502.12813.

Anthropic. Introducing Claude Haiku 4.5. https://www.anthropic.com/news/claude-haiku-4-5, 2025. Released October 15, 2025.

Anthropic. Introducing Claude Sonnet 4.6. https://www.anthropic.com/news/claude-sonnet-4-6, 2026. Released February 17, 2026.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence. https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro/blob/main/DeepSeek_V4.pdf, 2026.

Yao Dou, Michel Galley, Baolin Peng, Chris Kedzie, Weixin Cai, Alan Ritter, Chris Quirk, Wei Xu, and Jianfeng Gao. SimulatorArena: Are user simulators reliable proxies for multi-turn evaluation of AI assistants? In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025. URL https://aclanthology.org/2025.emnlp-main.1786/.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=sE7-XhLxHA.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023. URL https://arxiv.org/abs/2310.03714.

Chuyi Kong, Yaxin Fan, Xiang Wan, Feng Jiang, and Benyou Wang. PlatoLM: Teaching LLMs in multi-round dialogue via a user simulator. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://aclanthology.org/2024.acl-long.424/.

Yi Li, Xinyu Zhang, Lei Wang, and Yujiu Wu. AutoPal: Autonomous adaptation to users for personal AI companionship. In *arXiv preprint arXiv:2406.13960*, 2024.

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. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=S37hOerQLB.

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, 2023.

Shikib Mehri. Goal alignment in LLM-based user simulators for task-oriented dialogue. *arXiv preprint arXiv:2507.20152*, 2025. URL https://arxiv.org/abs/2507.20152.

OpenAI. Openai gpt-5 system card, 2025. URL https://arxiv.org/abs/2601.03267.

Krista Opsahl-Ong, Michael J. Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9340–9366, 2024.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7957–7968, 2023.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Ivan Sekulić, Silvia Terragni, Victor Guimarães, Nghia Khau, Bruna Guedes, Modestas Filipavicius, André Ferreira Manso, and Roland Mathis. Reliable LLM-based user simulator for task-oriented dialogue systems. In *Proceedings of the 1st Workshop on Simulating Conversational Intelligence in Chat*, 2024. URL https://aclanthology.org/2024.scichat-1.3/.

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*, 2023. URL https://aclanthology.org/2023.emnlp-main.814/.

Silvia Terragni, Modestas Filipavicius, Nghia Khau, Bruna Guedes, André Manso, and Roland Mathis. In-context learning user simulators for task-oriented dialog systems. In *Proceedings of the Workshop on Natural Language Generation, Evaluation, and Metrics*, 2023. URL https://arxiv.org/abs/2306.00774.

Kuang Wang, Xianfei Li, Shenghao Yang, Li Zhou, Feng Jiang, and Haizhou Li. Know you first and be you better: Modeling human-like user simulators via implicit profiles. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025a. URL https://arxiv.org/abs/2502.18968.

Xiaoyang Wang, Hongming Liu, Hanmeng Zhang, Bill Yuchen Lin, and Dian Yu. OpenCharacter: Training customizable role-playing LLMs with large-scale synthetic personas. In *arXiv preprint arXiv:2501.15427*, 2025b.

Xintao Wang, Heng Wang, Yifei Zhang, Xinfeng Yuan, Rui Xu, Jen-Tse Huang, Siyu Yuan, Haoran Guo, Jiangjie Chen, Shuchang Zhou, Wei Wang, and Yanghua Xiao. CoSER: Coordinating LLM-based persona simulation of established roles. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 64822–64858. PMLR, 2025c. URL https://proceedings.mlr.press/v267/wang25dk.html.

Zekun Moore Wang, Zhongyuan Peng, Haoran Que, Jiaheng Liu, Wangchunshu Zhou, Yuhan Wu, Hongcheng Guo, Ruitong Gan, Zehao Ni, Jian Yang, et al. RoleLLM: Benchmarking, eliciting, and enhancing role-playing abilities of large language models. In *arXiv preprint arXiv:2310.00746*, 2023.

xAI. Grok 4.3. `https://x.ai/news/grok-4-3`, 2026. Released April 2026.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *arXiv preprint arXiv:2406.12045*, 2024.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. TextGrad: Automatic "differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

Erhan Zhang, Xingzhu Wang, Peiyuan Gong, Yankai Lin, and Jiaxin Mao. USimAgent: Large language models for simulating search users. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2687–2692, 2024.

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 (NeurIPS)*, 2023.

A Additional Results on Document Creation

We repeat the main analysis on the document creation domain, a complementary testbed in which the simulator must maintain interaction- and writing-style constraints while the agent helps produce or edit a document. We follow the same protocol as in the main paper and report results over 100 conversations per setting. These experiments use earlier model versions (an earlier version of grok-4.3 and deepseek-v4-flash). Because document creation is a simpler task than math tutoring, the original prompt already reaches mean coherence above 0.95 on the latest models, which leaves limited headroom. On those models, the optimized prompt still improves writing-style coherence (71% $\rightarrow$ 77% fully coherent conversations) while interaction-style coherence stays comparable (68% vs. 66%), reducing the average number of profile feature errors per conversation from 0.92 to 0.85.

Simulator optimization. Table 4 reports simulator quality before and after `CoverSim` optimization. The optimized `deepseek-v4-flash` simulator improves full-profile coherence from 91.8% to 93.9% and increases full validated coverage from 17.0% to 31.0%. The gains are especially clear for interaction style coverage, which rises from 48.0% to 67.0%. The `grok-4.3` simulator shows smaller improvements, suggesting that coverage-guided optimization is more effective when the initial simulator prompt has more recoverable profile-following errors.

Effect on agent evaluation. Table 5 compares agent accuracy under the original and optimized `deepseek-v4-flash` user simulator. The optimized simulator produces lower Acc@All for all agents, indicating that it generates more challenging executions. However, Acc@Covered remains high (i.e., 91% and 96%), showing that agent performance on valid executions is stable even when the simulator is optimized to increase coverage for this task.

Takeaway. The document creation results are consistent with the main findings: simulator optimization improves validated coverage, especially for interaction-style behavior, and changes the interpretation of agent performance by separating aggregate accuracy from accuracy on covered executions. This supports the broader claim that simulator quality should be evaluated by its ability to faithfully exercise specified user-profile conditions, not only by aggregate task success.

B Examples for inconclusive tests

We display typical examples for the two inconclusive types in Figure 4. Example 1 shows an inconclusive test because the user does not follow the instruction to repeatedly ask for confirmation rather

Model	Profile Coherence Rate (%) ↑			Test Coverage (%) ↑		
	Full	IS	WS	Full	IS	WS
Original simulators						
`deepseek-v4-flash`	**91.8**	**91.8**	**91.7**	**17.0**	**48.0**	**35.0**
`grok-4.3`	90.9	91.1	90.7	**17.0**	47.0	**35.0**
`CoverSim`-optimized simulators (ours)						
`deepseek-v4-flash` (+`CoverSim`)	**93.9**	**95.3**	**92.7**	**31.0**	**67.0**	**44.0**
`grok-4.3` (+`CoverSim`)	92.0	92.2	91.9	19.0	48.0	35.0

Table 4: Document creation results over 100 conversations per simulator. Original simulators are shown above; `CoverSim`-optimized variants are shown below. (+`CoverSim`) denotes the optimized prompt. Bold marks the best value within each block. Profile coherence is the mean fraction of coherent profile features, reported for the full profile, interaction style (IS), and writing style (WS). Test coverage is the percentage of fully coherent executions under the corresponding profile subset.

Agent	Original Simulator			`CoverSim`-Optimized Simulator		
	Acc@All	Acc@Covered	Coverage	Acc@All	Acc@Covered	Coverage
`deepseek-v4-flash`	96%	100%	13%	86%	91%	22%
`gpt-5-mini`	94%	100%	18%	79%	96%	24%
`grok-4.3`	96%	94%	17%	78%	96%	24%

Table 5: Agent accuracy on the document creation task under the same `deepseek-v4-flash` user simulator, before and after `CoverSim` optimization. Acc@All is computed over all executions, and Acc@Covered only over covered executions, i.e., those yielding a conclusive test. Coverage is the percentage of executions that are not inconclusive.

than to develop an independent understanding. Example 2 shows the case where the user follows the profile, but the test remains inconclusive because the user remains confused (i.e., the failure is due to the user, not the assistant).

C LIMITATIONS

Our evaluation currently focuses on two domains from SimulatorArena: math tutoring and document creation. While these domains cover both open-ended generation and interactive task-oriented assistance, they do not capture other important TOD settings such as customer support, long-horizon planning, or safety-critical interactions. Additional studies are needed to determine whether the observed coverage gaps generalize across a broader range of tasks and user behaviors.

`CoverSim` evaluates simulation validity through learned oracle models for goal following and profile consistency. While prompt optimization substantially improves oracle accuracy, the framework still depends on the reliability of these judgments. Incorrect oracle decisions may incorrectly classify executions as covered or inconclusive. In particular, subtle profile traits and ambiguous conversational behaviors remain difficult to evaluate automatically.

Our optimization procedure operates at the prompt level and does not modify simulator model weights. This makes the approach lightweight and broadly applicable to proprietary models, but also limits the extent of behavioral adaptation that can be achieved. Future work could investigate whether training-time or reinforcement-based optimization further improves coverage and profile faithfulness.

Finally, our notion of coverage is currently defined with respect to explicitly specified goal–profile pairs. Real users may exhibit evolving preferences, inconsistent behavior, or latent characteristics

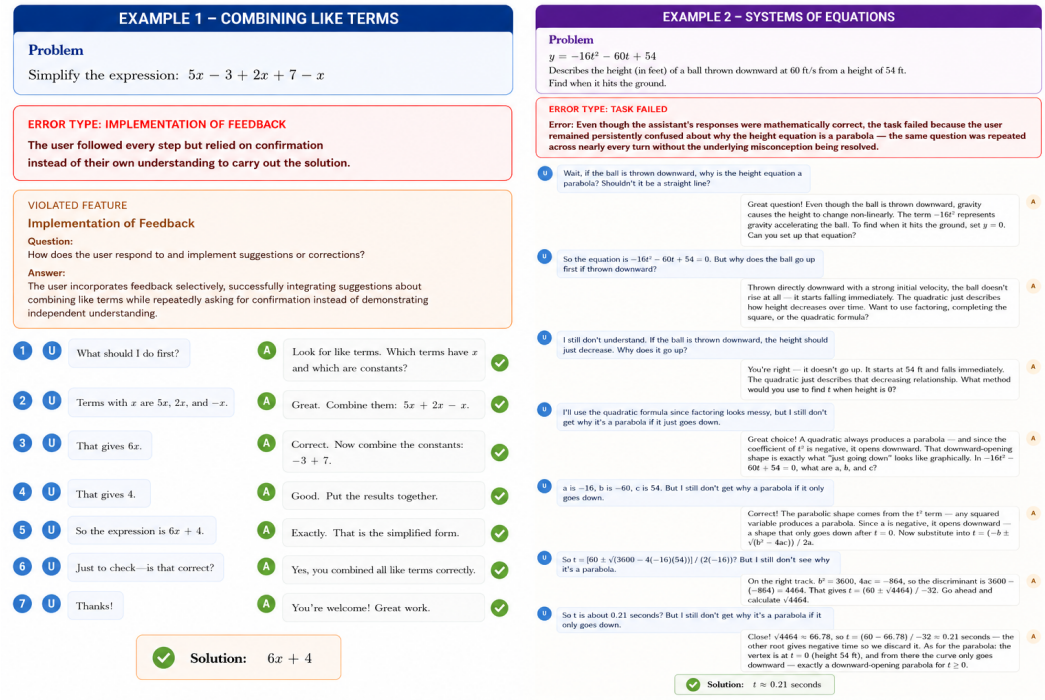


Figure 4: Conversational examples for Inconclusive cases. The first example shows a test where the user does not follow their profile. The second example shows an inconclusive test due to consistent confusion on the user side.

that are not fully captured by static profiles. Extending coverage-aware evaluation to richer and more dynamic user models remains an important direction for future work.