TURING^2: SCALING TURING-STYLE EVALUATION TO OPTIMIZE HUMAN SIMULATION IN DIALOGUE

Anonymous authors
Paper under double-blind review

ABSTRACT

Realistic user simulation is essential for developing and evaluating conversational AI agents, yet measuring and optimizing simulation fidelity remains challenging. Human Turing tests assess whether simulated behavior appears human, but are costly to use for iterative optimization, motivating scalable evaluation with Machine-Turing judges. We introduce TURING^2, a human-grounded co-evolutionary framework for improving user simulators and Machine-Turing judges through prompt optimization, without updating model parameters. Machine-Turing judges provide rewards and diagnostic feedback to improve simulators, while human Turing judgments and explanations guide judge refinement. The refined judge then provides feedback for further simulator improvement. In customer-service dialogue, simulator optimization improves Turing win rates under both held-out LLM judges and human annotators. Incorporating human explanations improves held-out judge accuracy, and optimizing against the refined judge further improves simulator human-perceived human-likeliness. These results demonstrate the effectiveness of TURING^2 in jointly improving user simulators and their evaluators through human-grounded co-evolution. The dataset would be released upon acceptance.

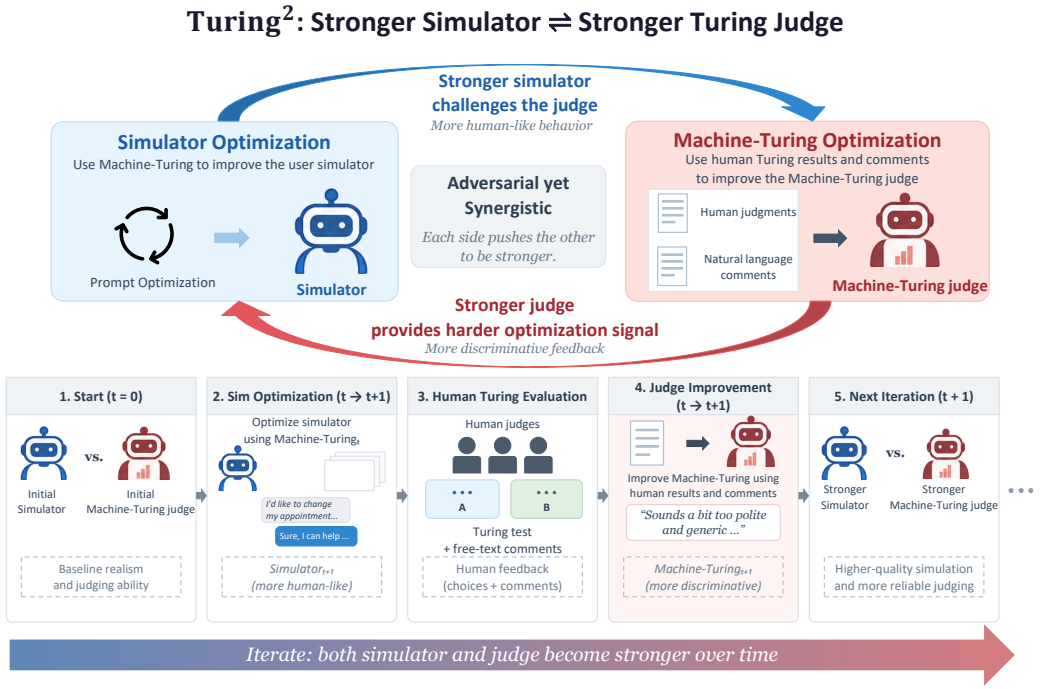

Figure 1: The TURING^2 judge–simulator co-evolution flywheel.

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.

1 INTRODUCTION

Large language models (LLMs) are increasingly used as simulated users for training and evaluating multi-turn conversational agents. Compared with human participants, LLM-based user simulators are inexpensive, reproducible, and easy to scale across tasks, user profiles, and interaction scenarios. However, these benefits are only meaningful when the simulator behaves like the humans it is intended to represent. This raises a fundamental research question: *How can we build and evaluate LLM-based user simulators that interact with conversational agents with high human fidelity?*

Achieving such fidelity is challenging because LLMs are primarily optimized to be helpful assistants rather than realistic users. Behaviors that are desirable for an assistant (e.g. being cooperative, consistent, and responsive to every part of a request) may be unrealistic when the model is asked to simulate a human. Real users may provide incomplete information, misunderstand questions, change their goals, or behave inconsistently across turns. Consequently, a stronger conversational assistant is not necessarily a stronger user simulator; in some cases, its assistant-like tendencies may make it a less faithful representation of human behavior Naous et al. (2026).

This simulation-to-reality gap has important consequences for both agent evaluation and training. *For evaluation*, unrealistic user behavior can distort interaction trajectories and misrepresent agent capabilities. Prior work has shown that LLM-based simulators tend to be excessively cooperative and insufficiently reflect human ambiguity, frustration, and conversational friction, thereby inflating estimated agent success rates Zhou et al. (2026b). Agent performance is also sensitive to simulated user traits: He et al. (2026) reports performance drops ranging from 4% to 46% in frontier models when simulated users exhibit traits such as impatience or incoherence. Consequently, success rates and comparisons between agents obtained in simulation may not reliably transfer to real-world interactions. *For training*, the user simulator serves as the interactive environment from which the agent learns. Unrealistically cooperative users provide limited opportunities for the agent to learn how to resolve misunderstandings, respond to pushback, or handle frustration. Agents trained in such environments may therefore succeed in simulation while remaining brittle in realistic interactions. High-fidelity user simulation is thus essential both for evaluating agents under realistic conditions and for exposing them to the behavioral challenges they must learn to navigate.

Table 1: Taxonomy of automatic methods for evaluating user-simulator fidelity.

Evaluation family	**Evaluation method**	**Granularity**	**Criterion**	**Target**
Distributional alignment	Behavior-frequency, linguistic-feature, and distributional comparisons Zhou et al. (2026b); Liu et al. (2026); Bao et al. (2026); Hathidara et al. (2026)	Population	Predefined	Process
Reference-response matching	Likelihood, lexical or semantic similarity, and embedding-based alignment Naous et al. (2026); Wang et al. (2026)	Individual turn	Reference-dependent	Local process
Rubric-based realism scoring	LLM-judge scores over specified realism dimensions Zhu et al. (2025); Hathidara et al. (2026)	Individual dialogue	Rubric-defined	Process
Outcome alignment	Task-outcome, rating, or system-ranking agreement Zhou et al. (2026b); Dou et al. (2025)	Population	Predefined	Outcome
Turing-style distinguishability	Real-versus-simulated discrimination Wang et al. (2026); Hathidara et al. (2026)	**Individual dialogue**	**Open-ended**	**Process**

Despite its importance, evaluating user-simulator fidelity remains challenging because fidelity is multidimensional and lacks a unified standard. As shown in Table 1, existing work operationalizes realism through heterogeneous criteria, including behavioral metrics Zhou et al. (2026b); Hathidara et al. (2026), reference-response likelihood Naous et al. (2026), and task outcomes, making results difficult to compare across studies. More fundamentally, these approaches differ in their evaluation granularity, criterion, and target. Turn-level evaluation may provide insufficient evidence in customer-service settings, where an isolated response such as "Hello, I want to return this item" is plausible for either a human or a simulator. Population-level evaluation can reveal systematic dis-

tributional differences, but it aggregates over individual interactions and therefore provides limited localized feedback signal about which conversations fail and why.

To this end, we focus on evaluating the *human-perceived human-likeness* of complete interaction trajectories using an open-ended criterion. Dialogue-level evaluation captures properties that emerge across turns, such as how users reveal information, respond to misunderstandings, revise their requests, and introduce realistic conversational friction. An open-ended evaluator is also not restricted to an existing taxonomy of behaviors and can exploit any available conversational evidence. This is important because predefined metrics and rubrics can measure only anticipated properties and may overlook artifacts that have not yet been formalized. As we show later, Machine-Turing tests reveal recurring cues outside the behavioral dimensions covered by existing fidelity metrics.

We therefore study Turing-test-based evaluation as a complementary measure of interaction-process fidelity. A Turing-style evaluator determines whether a complete interaction is recognizable as human or simulated based on any available evidence, providing an individual-dialogue signal that can be used to diagnose and improve the simulator. This assesses whether the interaction unfolds in a human-like manner, without requiring task-specific outcome definitions, repeated downstream agent training, or real-user evaluation. However, conventional Turing tests rely on human judges, making them costly and difficult to scale. We address this limitation by investigating whether LLM judges can provide reliable and scalable Turing-test-based evaluation, and by distilling human judge feedback to further improve the accuracy and robustness of LLM-based evaluators.

Having established a scalable approach to fidelity evaluation, we next use its feedback to optimize the simulator. Specifically, we revise the simulator prompt to reduce behavioral cues that distinguish simulated users from humans. This naturally motivates an iterative process: a more accurate judge can provide better feedback for simulator optimization, while human assessment of the resulting simulator can inform further judge refinement. We therefore introduce TURING^2, a human-grounded co-evolutionary framework in which user simulators and Machine-Turing judges iteratively improve each other. The current judge guides simulator optimization, and human Turing tests then collect judgments and natural-language rationales to refine the judge for the next iteration. In task-oriented dialogue, we show that TURING^2 improves simulator realism according to both LLM judges and human evaluators. Human-rationale-assisted refinement further improves Machine-Turing judge accuracy, and optimizing against the refined judge yields additional simulator gains.

We summarize our contributions as follows:

- We analyze the limitations of existing automatic methods for evaluating user-simulator fidelity and motivate dialogue-level Turing tests that assess whether simulated interactions are distinguishable from human interactions.
- We investigate Machine-Turing tests, which use LLM judges to distinguish human from simulated dialogues, and demonstrate their reliability for scalable evaluation of simulator human-likeness through alignment with human judgments.
- We introduce TURING^2, a human-grounded co-evolutionary framework in which Machine-Turing feedback guides simulator prompt optimization, while human judgments and rationales guide judge prompt refinement. We demonstrate improved simulator realism through LLM and human evaluation, improved held-out judge accuracy, and further simulator gains from the refined judge under a held-out LLM evaluator.

2 METHODOLOGY

Our methodology comprises three components: (1) dialogue-level Turing tests for evaluating user-simulator human-perceived human-likeliness, (2) validation of LLM judges as scalable Machine-Turing evaluators, and (3) a human-grounded co-evolution of judges and simulators through iterative prompt optimization.

2.1 TURING-STYLE EVALUATION OF USER-SIMULATOR FIDELITY

Existing work Wang et al. (2026); Guo et al. (2026) operationalizes human-likeness through indistinguishability from real users. Following this principle, we evaluate user-simulator human-perceived

human-likeliness at the level of complete dialogues. We present the judge with a human conversation and a simulated conversation collected under the same task settings and ask it to identify the simulated one, allowing an explicit "not sure" response. The judge also provides a natural-language rationale for its decision (see Appendix G for the full prompt). We then show our advantage over other evaluation methods.

Fidelity Cues Beyond a Predefined Taxonomy We use *cue* to denote an observable property of an individual dialogue that the judge cites as evidence for its decision. We compare these cues with the predefined D1–D4 taxonomy in TAU-USI Zhou et al. (2026b), referring to cues not explicitly represented by its dimensions as *outside-taxonomy cues*. Table 2 summarizes these cues, including surface polish, frictionless task progression, internal inconsistencies, and conversational discontinuities. These are broader forms of fidelity evidence: some concern interaction behavior, while others concern style, formatting, or dialogue integrity. Their occurrence in judge rationales illustrates how open-ended evaluation can surface potential fidelity gaps beyond those explicitly measured by a fixed taxonomy. Appendix C examines their predictive value.

Table 2: Outside-taxonomy cue groups cited by our Turing judge, relative to the TAU-USI D1–D4 dimensions Zhou et al. (2026b). Percentages describe the distribution of 167 outside-taxonomy cue mentions, not the proportion of dialogues exhibiting each cue.

Outside-taxonomy cue group	Share (%)	Observable pattern identified by the judge
Surface polish and formatting	38.9	Perfect punctuation, absence of typos, highly structured phrasing, excessive politeness, or mirroring of the agent's formatting.
Scripted progression and implausible compliance	31.1	Complete and ordered responses to multi-field prompts, persistent cooperation, and unusually frictionless task progression.
Internal incoherence and implausible or fabricated data	16.8	Contradictions across turns, inconsistency with the assigned persona or goal, or introduction of unsupported details.
Missing turns and flow artifacts	3.0	Empty responses, abrupt transitions, or broken conversational continuity.
Capability-probing behavior	3.0	Unnatural, test-like attempts to probe the agent's capabilities, policies, or limitations.
Other low-frequency cues	7.2	Finer-grained variants of the above themes that occur too rarely to form separate categories.

RNR Blind Spots We compare our Turing test with Rubric-and-Reason (RNR) Hathidara et al. (2026), a reference-free realism evaluation based on three fixed criteria: concise real-user language, non-scripted behavior, and real-user tone (Appendix F). RNR evaluates each conversation independently, without a paired human reference. We obtain three independent RNR judgments per dialogue and define a pass as three positive judgments. For each simulator, we evaluate 93 paired customer-service scenarios from ABCD Chen et al. (2021). The human and simulated conversations share the same seed opening and interact with the same agent; only the customer differs. Both evaluation protocols use `Gemini 3.5 Flash`, holding the judge model fixed while varying the evaluation protocol. For the Turing test, we evaluate both presentation orders and count a dialogue as *detected* only when the judge correctly identifies it as simulated in both orders.

Table 3 shows that rubric acceptance does not imply indistinguishability from a paired human conversation. RNR passes 87 of 93 GPT-5-mini dialogues (93.5%) and 91 of 93 GPT-5.5 dialogues (97.8%). Among these passing dialogues, the Turing judge correctly identifies 74 (85.1%) and 43 (47.3%), respectively, in both presentation orders. Of these, 64 and 28 are identified with high confidence in both orders. These results demonstrate that paired Turing evaluation provides a complementary distinction among dialogues accepted by RNR. See Appendix 5 for qualitative results.

Table 3: RNR-passing dialogues that remain distinguishable from paired human conversations. An RNR pass requires three positive judgments; Turing detection requires correct identification in both presentation orders.

Customer model	**GPT-5-mini**	**GPT-5.5**
RNR pass	87/93 (93.5%)	91/93 (97.8%)
RNR pass but Turing caught	**74/87 (85.1%)**	**43/91 (47.3%)**
High-confidence in both orders	64	28

2.2 Scaling Turing-Style Evaluation with LLM Judges

Human Turing tests are costly for repeated, large-scale evaluation. We therefore investigate LLMs as scalable Machine-Turing judges, assessing both agreement with human judgments and internal reliability. These properties are distinct: a judge can produce consistent decisions without agreeing with humans.

We report five metrics. Judge accuracy scores correct identification of the simulated conversation as 1, incorrect identification as 0, and a "not sure" response as 0.5. Raw agreement measures the fraction of binary decisions shared by two judges, while Cohen's κ measures agreement beyond chance. We compute these agreement metrics between LLM and human judges and among human judges. For internal reliability, we measure test–retest reliability using $\mathrm{ICC}(2,1)$ across repeated evaluations of identical conversation pairs, and position consistency as the fraction of pairs receiving the same decision after reversing presentation order and mapping decisions back to conversation identities.

Table 4 shows that Machine-Turing judges achieve human agreement rates of 56.7–64.1%, close to the 62.5% agreement among human evaluators. Thus, human–LLM disagreement is comparable in magnitude to human–human disagreement. Several Machine-Turing judges also exhibit strong test–retest reliability, with Claude Sonnet 4.6 and Claude Opus 4.8 achieving ICCs above 0.96, indicating consistent scores across repeated evaluations.

Table 4: Human alignment and internal reliability of Turing judges across all evaluated trajectories. For the Human row, raw agreement and Cohen's κ measure agreement among human evaluators; for each LLM row, they measure agreement with human judgments.

Judge	**Judge Accuracy**	**Cohen's κ**	**Agreement w/ Human**	**Test–Retest Agreement**	**Position Consistency**
Human	0.5021	0.2499	0.6250	–	–
Claude Sonnet 4.6	0.4217	0.1751	0.5912	0.983	0.6933
Claude Opus 4.8	0.4742	0.1306	0.5670	0.964	0.6633
GLM 5.2	0.4667	0.1580	0.5833	0.776	0.6856
Gemini 3.5 Flash	0.4892	0.2803	0.6409	0.856	0.6933

2.3 HUMAN-IN-THE-LOOP JUDGE–SIMULATOR CO-EVOLUTION

TURING^2 couples simulator and Machine-Turing judge optimization through a human-in-the-loop framework (Figure 1). The judge guides simulator optimization through rewards and diagnostic rationales. Human evaluation of the resulting simulator then informs judge refinement, and the refined judge guides the next simulator update. Both stages optimize natural-language prompts while keeping the underlying model parameters fixed.

Judge-guided simulator optimization. Figure 2 illustrates the simulator prompt-optimization pipeline. Within each simulator-optimization stage, we hold the judge, simulator model, professional agent, task scenarios, and evaluation protocol fixed. Each rollout uses the same stored scenario configuration, customer profile, initial seed turns, and professional-agent model as its human reference.

Using the Turing-test protocol described above, we convert each judge decision into a simulator reward:

$$r = \begin{cases} 1, & \text{judge misidentifies the simulator as human,} \\ 0.5, & \text{judge is uncertain,} \\ 0, & \text{judge correctly distinguishes the pair.} \end{cases} \tag{1}$$

We average the rewards from both presentation orders to obtain a trajectory-level reward. The prompt-level objective is the mean trajectory reward across the optimization set; higher reward indicates lower distinguishability from human conversations.

The optimizer aggregates rationales from low-reward trajectories into an *error gradient*: a natural-language summary of recurring cues that reveal simulated behavior. A prompt-editing model uses this summary to propose a localized revision while preserving the assigned customer goal, persona, scenario facts, and task constraints. We evaluate each revision through new rollouts on a separate prompt-selection set and accept it only if it improves validation reward. The prompt with the highest validation reward is selected as the optimized simulator prompt.

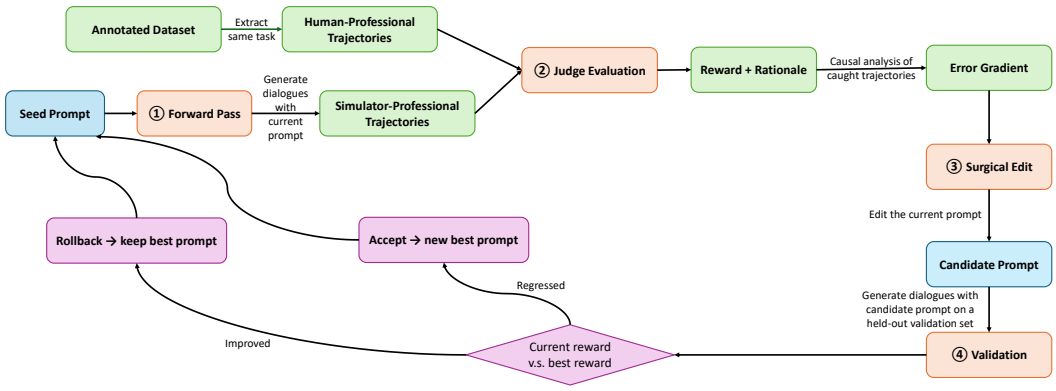

Figure 2: Simulator prompt optimization. The judge scores simulated conversations against matched human references and provides diagnostic rationales. The optimizer summarizes recurring failure patterns into an error gradient and proposes a localized prompt revision. Revisions are evaluated on a separate prompt-selection set and retained only when they improve validation reward.

Human-rationale-assisted judge optimization. We collect human judgments and rationales on conversations generated by the optimized simulator, paired with their human references. Using the known conversation identities, we label each human judgment as correct, incorrect, or uncertain. These labels contextualize the accompanying rationales so that the optimizer can distinguish evidence supporting correct decisions from evidence associated with errors or uncertainty.

We refine the judge prompt through a similar feedback-driven editing procedure. The optimizer receives both the machine judge's prediction errors against known conversation identities and the human rationales with their correctness labels. It uses this feedback to identify overlooked evidence and misleading decision criteria, then revises the judge instructions. Whereas simulator optimization aims to make simulated conversations appear more human-like, judge optimization targets correct discrimination between human and simulated conversations. Human rationales supplement the correctness signal with explanations of the evidence underlying human decisions.

Closing the co-evolution loop. The refined judge is held fixed during the next simulator-optimization stage, providing updated rewards and diagnostic rationales. Human evaluation of the resulting simulator can then supply feedback for another judge update. The two components thus have opposing local objectives but complementary roles: simulator outputs provide examples for judge refinement, while improved judges provide stronger feedback for simulator optimization. Human feedback grounds judge refinement throughout this repeatable process.

3 Experiments

We evaluate three questions: (1) whether Machine-Turing-guided prompt optimization improves simulator human-perceived human-likeliness under held-out LLM judges, (2) whether these gains transfer to human evaluation, and (3) whether human feedback improves judge accuracy and the refined judge enables further simulator gains.

3.1 Tasks and Dialogue Collection

Source tasks. We derive customer-service tasks from ABCD Chen et al. (2021), spanning ten domains: account access, account management, order issues, product defects, purchase disputes, shipping issues, single-item queries, storewide queries, subscription inquiries, and website troubleshooting. For each task, we extract the scenario configuration and customer profile into a structured role card specifying the customer's goal, relevant information, and constraints.

AI–AI and Human–AI conversations. For AI–AI collection, a customer LLM receives the role card and interacts with a service-agent LLM given the corresponding agent instructions until the

conversation terminates. Unless otherwise specified, we use GPT-5.5 as the customer simulator and GPT-5.2 as the service agent. For human references, we collect conversations on 100 ABCD tasks, each completed by three independent human annotators acting as customers. Human and simulated customers receive the same role cards and interact with the same service-agent model, holding task settings fixed. During simulator optimization, we vary only the customer-simulation prompt.

Data-collection interfaces. We use two web interfaces: an interaction interface displaying the customer role card alongside a chat window connected to the service-agent LLM, and a Turing-evaluation interface presenting anonymized Human–AI and AI–AI dialogue pairs in randomized A/B order. Evaluators identify the conversation containing the real human customer or select "not sure."

3.2 Evaluation Protocols

LLM-based Turing evaluation. We use LLMs that are not involved in the prompt optimization pipeline as held-out Turing judges. For each task and simulator prompt, we generate five independent AI–AI conversations. Each simulated conversation is paired with the corresponding human–AI conversation and evaluated in both A/B presentation orders to reduce position bias.

The judge identifies which conversation contains the human customer and provides a rationale for its decision. We define the *Turing win rate (TWR)* for each comparison as 1 if the judge selects the simulated trajectory as the more human-like conversation, 0 if it selects the matched human trajectory, and 0.5 if it returns a tie. The mean TWR therefore measures how often the simulator is preferred as human relative to its matched human reference. TWR is a directional pairwise win rate rather than a symmetric distance from indistinguishability.

Because the seed and optimized prompts are evaluated on the same tasks, all statistical comparisons are paired at the task level. We report the mean TWR, its 95% confidence interval, the mean paired improvement (Δ = Best − Seed), and a paired permutation test. We additionally use a Wilcoxon signed-rank test as a nonparametric robustness check. The (+/-/=) column reports the numbers of tasks on which the optimized prompt improves, worsens, or ties the seed prompt.

Human Turing evaluation. Human evaluators receive the same anonymized dialogue pairs used in LLM-based evaluation and select which conversation contains the human customer or choose not sure." Each task is evaluated independently by three annotators. We report both individual judgments and majority-vote results, treating not sure" according to the same tie-scoring rule used for the LLM evaluation.

3.3 Results under LLM Judges

Table 5 compares the seed and optimized simulator prompts under three held-out LLM judges and the source judge used during optimization. The optimized prompt significantly improves mean TWR under every judge, with gains of 20.72–26.94 percentage points under held-out judges. These improvements exceed the gain under the source judge, showing that the benefits transfer across evaluator models rather than remaining confined to the optimization judge.

Improvements occur on a majority of tasks under every evaluator, indicating that the gains are broadly distributed across scenarios. The optimized prompt also consistently outperforms the generic realistic prompt (GRP), suggesting that feedback-driven optimization provides benefits beyond simply instructing the simulator to behave realistically.

3.4 Human Judges

Validation with human judges We next examine whether the improvement measured by LLM judges transfers to human evaluators. As shown in Table 6, the optimized prompt achieves a significantly higher Turing win rate than the seed prompt under human judgment. This result indicates that the observed gains are not limited to the LLM judge used during optimization and provides evidence against judge-specific overfitting.

Table 5: Comparison between the seed and best prompts by LLM judge. "GRP" is generic realistic prompt. "Seed" and "Best" are seed and best simulator prompt. Prompt optimization brings significant Turing win rate gain. Here, Δ denotes best minus seed, and $+/-/=$ indicates the number of improved, worsened, and tied tasks. Asterisks indicate $p < 0.05$.

	Judge	Prompt	Mean TWR	95% CI	Δ	p_{perm}	$+/-/=$
Held-out judges	Claude Sonnet 4.6	GRP	0.4369	$(0.3775, , 0.4964)$			
		Seed	0.4633	$(0.4011, , 0.5272)$	0.2694	$< 0.0001^*$	45/4/1
		Best	0.7328	$(0.6817, , 0.7828)$			
	Claude Opus 4.8	GRP	0.4278	$(0.3678, , 0.4917)$			
		Seed	0.4383	$(0.3722, , 0.5067)$	0.2339	$< 0.0001^*$	40/8/2
		Best	0.6722	$(0.6206, , 0.7222)$			
	GLM 5.2	GRP	0.4172	$(0.3517, , 0.4833)$			
		Seed	0.4272	$(0.3583, , 0.4989)$	0.2072	$< 0.0001^*$	37/11/2
		Best	0.6344	$(0.5789, , 0.6911)$			
Source judge	Gemini 3.5 Flash	GRP	0.4506	$(0.3928, , 0.5078)$			
		Seed	0.4769	$(0.4117, , 0.5408)$	0.1058	0.0044^*	32/18/0
		Best	0.5828	$(0.5250, , 0.6400)$			

Table 6: Comparison between the seed and best prompts by human judge. Prompt optimization brings significant Turing win rate gain. Here, Δ denotes best minus seed, and $+/-/=$ indicates the number of improved, worsened, and tied tasks. Asterisks indicate $p < 0.05$.

Prompt	Mean TWR	95% CI	Δ	p_{perm}	$+/-/=$
Seed simulator prompt	0.4373	$(0.395, 0.480)$	0.0692	0.0168^*	29/16/5
Best simulator prompt	0.5065	$(0.460, 0.553)$			

Distilling human feedback into a judge–simulator flywheel. Rather than using human judgments only for evaluation, we distill the reasoning behind those judgments into scalable LLM judges. We compare two controlled prompt-optimization conditions: an error-only judge that learns from its own prediction failures, and a human-rationale-assisted judge that additionally receives human explanations together with whether each judgment was correct, incorrect, or uncertain. As shown in Table 7, although error-only optimization improves held-out accuracy from 0.468 to 0.558, incorporating human rationales raises it further to 0.599. Human rationales also help the judge discover behavioral evidence that self-reflection misses, substantially reducing its reliance on the spurious shortcut that longer conversations are more likely to be human. Crucially, these gains transfer downstream: when used as the reward for simulator prompt optimization, the rationale-assisted judge produces a simulator with the highest Turing win rate under every held-out evaluator, including an independent Gemini judge (0.524, compared with 0.442 for error-only optimization and 0.311 for the seed simulator). These results demonstrate the first turn of the TURING^2 human–judge–simulator flywheel: human insight improves scalable judges, and stronger judges provide more effective feedback for producing higher-fidelity simulators. The framework also offers a natural interface through which domain experts can contribute specialized behavioral knowledge without requiring model fine-tuning. See Appendix D for full results.

Table 7: Human-rationale distillation improves held-out judge accuracy and downstream simulator quality. Judge accuracy is measured on the newly collected held-out set, which is not used during judge optimization. Simulator Turing win rate is evaluated by a held-out Gemini judge on 60 held-out tasks from independently collected datasets; higher values indicate that simulated customers are more frequently selected as human. Asterisks denote paired permutation $p < 0.05$ relative to the seed simulator.

Method	Judge Accuracy	Simulator Win Rate (Gemini)
Seed judge / seed simulator	0.468	0.311
Seed judge + simulator optimization	0.468	0.428^*
Error-only judge optimization + simulator optimization	0.558	0.442^*
Human-rationale-assisted judge + simulator optimization (ours)	**0.599**	$\mathbf{0.524}^*$

4 Related Work

4.1 Evaluating User-Simulator Fidelity

The Turing test Turing (2007) motivates evaluating simulator fidelity through human–machine indistinguishability. Early work compared automatic metrics with human realism judgments Ai & Litman (2008), without paired discrimination of task-matched conversations. Recent approaches evaluate predefined behavioral dimensions Liu et al. (2026); Zhu et al. (2026), representation-level distribution gaps Mehri et al. (2026), unified simulation capabilities Zhou et al. (2026a), or agreement with human-based agent evaluation Dou et al. (2025); Zhou et al. (2026b). These evaluations depend on prespecified dimensions, representations, or outcomes; our approach allows judges to use any observable conversational cue and analyzes their rationales to uncover fidelity gaps.

Like MirrorBench Hathidara et al. (2026) and Turing-RL Wang et al. (2026), we use pairwise Turing-style evaluation. Our TWR scores simulator selection as human, human-reference selection, and ties as 1, 0, and 0.5, respectively. It is a directional win rate: 0.5 indicates parity, and higher values indicate more frequent simulator selection, rather than classification accuracy or a symmetric distance from parity.

4.2 Improving User Simulators

Prior work improves simulators through implicit profiles, conditional fine-tuning, and cycle-consistency rewards Wang et al. (2025); optimization for persona adherence, conversational quality, and decision alignment Pruksachatkun et al. (2026); and behavioral profiles grounded in real interactions Zhu et al. (2026). Most closely related, Turing-RL Wang et al. (2026) uses distinguishability from real responses as an RL reward and validates improvements with human and LLM Turing tests.

Turing^2 couples simulator improvement with human-grounded judge refinement. Machine-Turing rewards and diagnostic rationales from complete dialogues guide simulator prompt optimization, while human judgments and rationales inform judge prompt refinement. The refined judge then guides further simulator optimization, forming a co-evolution loop. Both components are updated through prompt optimization without modifying model parameters, supporting proprietary models.

4.3 Customer Simulation and Downstream Consequences

Fluent simulators can distort downstream evaluation. Simulated Customers Never Walk Away Chen (2026) identifies gaps in purchasing resistance, deliberation, and conversation depth; SalesSim Pruksachatkun et al. (2026) studies persona-conditioned shoppers in multimodal retail; and TAU-USI Zhou et al. (2026b) shows that overly cooperative users can inflate task success and alter agent rankings.

5 Conclusion

We introduce Turing^2, a human-grounded framework for evaluating and improving human-perceived human-likeliness of LLM-based user simulators through judge–simulator co-evolution. Dialogue-level Turing tests complement predefined fidelity metrics by revealing conversational cues beyond existing taxonomies. We show that Machine-Turing judges achieve agreement with humans close to human–human agreement and provide useful feedback for simulator prompt optimization, yielding gains under held-out judges and human evaluation. We further demonstrate one co-evolution cycle: human judgments and rationales improve held-out judge accuracy, and optimization against the refined judge produces additional simulator gains under a held-out LLM evaluator. These findings support iterative refinement of both simulators and judges through prompt optimization, with human feedback grounding judge improvement.

6 ETHICS STATEMENT

Human data were collected through collaboration with an internal annotation team, with additional contributions from colleagues. Participants interacted with a conversational agent to produce human reference dialogues and provided Turing-test judgments and written rationales. These data supported simulator evaluation and optimization, as well as judge refinement. We follow our organization's standard procedure of human annotation.

REFERENCES

Hua Ai and Diane Litman. Assessing dialog system user simulation evaluation measures using human judges. In *Proceedings of ACL-08: HLT*, pp. 622–629, 2008.

Eliseo Bao, Anxo Perez, Xi Wang, and Javier Parapar. Eval4sim: An evaluation framework for persona simulation. *arXiv preprint arXiv:2603.02876*, 2026.

Derek Chen, Howard Chen, Yi Yang, Alex Lin, and Zhou Yu. Action-based conversations dataset: A corpus for building more in-depth task-oriented dialogue systems. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021*, pp. 3002–3017, Online, June 2021. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2021.naacl-main.239.

Liang Chen. Simulated customers never walk away: Decision fidelity of llm user simulators measured against real purchase outcomes. *arXiv preprint arXiv:2606.20708*, 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*, pp. 35200–35278, 2025.

Minghao Guo, Ziyi Ye, Wujiang Xu, Xi Zhu, Wenyue Hua, and Dimitris N Metaxas. Individual turing test: A case study of llm-based simulation using longitudinal personal data. *arXiv preprint arXiv:2603.01289*, 2026.

Ashutosh Hathidara, Julien Yu, Vaishali Senthil, Sebastian Schreiber, and Anil Babu Ankisettipalli. Mirrorbench: An extensible framework to evaluate user-proxy agents for human-likeness. *arXiv preprint arXiv:2601.08118*, 2026.

Muyu He, Anand Kumar, Soumyadeep Bakshi, James Zou, and Nazneen Rajani. Impatient users confuse ai agents: High-fidelity simulations of human traits for testing agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 37569–37602, 2026.

Yu Lu Liu, Hyokun Yun, Tanya Roosta, and Ziang Xiao. Synthetic users, real differences: an evaluation framework for user simulation in multi-turn conversations. *arXiv preprint arXiv:2605.02624*, 2026.

Shuhaib Mehri, Philippe Laban, Sumuk Shashidhar, Marwa Abdulhai, Sergey Levine, Michel Galley, and Dilek Hakkani-Tür. Measuring and mitigating the distributional gap between real and simulated user behaviors. *arXiv preprint arXiv:2605.07847*, 2026.

Tarek Naous, Philippe Laban, Wei Xu, and Jennifer Neville. Flipping the dialogue: Training and evaluating user language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ykSmkVqzn4.

Yada Pruksachatkun, Elaine Wan, Lyanna Chen, Kai-Wei Chang, and Chien-Sheng Wu. Salessim: Benchmarking and aligning multimodal language models as retail user simulators. *arXiv preprint arXiv:2605.08334*, 2026.

Alan M Turing. Computing machinery and intelligence. In *Parsing the Turing test: Philosophical and methodological issues in the quest for the thinking computer*, pp. 23–65. Springer, 2007.

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 (Volume 1: Long Papers)*, pp. 21082–21107, 2025.

Yingshan Susan Wang, Cedegao E Zhang, Linlu Qiu, Zexue He, Pengyuan Li, Alex Pentland, Roger P Levy, and Yoon Kim. Learning user simulators with turing rewards. *arXiv preprint arXiv:2606.19336*, 2026.

Xuhui Zhou, Weiwei Sun, Weihua Du, Jiarui Liu, Haojia Sun, Qianou Ma, Tongshuang Wu, Yiming Yang, and Maarten Sap. Odyssim: Building foundation models for human behavior simulation. *arXiv preprint arXiv:2606.14199*, 2026a.

Xuhui Zhou, Weiwei Sun, Qianou Ma, Yiqing Xie, Jiarui Liu, Weihua Du, Sean Welleck, Yiming Yang, Graham Neubig, Sherry Tongshuang Wu, et al. Mind the sim2real gap in user simulation for agentic tasks. *arXiv preprint arXiv:2603.11245*, 2026b.

Ming Zhu, Juntao Tan, Rithesh Murthy, Jielin Qiu, Liangwei Yang, Wenting Zhao, Silvio Savarese, Shelby Heinecke, and Huan Wang. Realusersim: Bridging the reality gap in agent benchmarking via grounded user simulation. *arXiv preprint arXiv:2605.20204*, 2026.

Ruizhe Zhu, Hao Zhu, Yaxuan Li, Syang Zhou, Shijing Cai, Malgorzata Lazuka, and Elliott Ash. Dialogueforge: Llm simulation of human-chatbot dialogue. *arXiv preprint arXiv:2507.15752*, 2025.

Table 8: Prompt-optimization results under different seed-prompt initializations. For each initialization, Δ denotes the optimized best prompt minus its corresponding seed prompt, and $+/-/=$ indicates the number of improved, worsened, and tied tasks. Asterisks indicate $p < 0.05$.

	Judge	Initialization	Prompt	Mean TWR	95% CI	Δ	p_{perm}	$+/-/=$
Held-out judges	Claude Sonnet 4.6	Plain seed	Seed	0.4633	(0.4011, 0.5272)	0.2694	$< 0.0001^*$	45/4/1
			Best	0.7328	(0.6817, 0.7828)			
		Metric-conditioned seed	Seed	0.4994	(0.4411, 0.5606)	0.2039	$< 0.0001^*$	39/6/5
			Best	0.7033	(0.6511, 0.7539)			
	Claude Opus 4.8	Plain seed	Seed	0.4383	(0.3722, 0.5067)	0.2339	$< 0.0001^*$	40/8/2
			Best	0.6722	(0.6206, 0.7222)			
		Metric-conditioned seed	Seed	0.4839	(0.4217, 0.5467)	0.1983	$< 0.0001^*$	42/7/1
			Best	0.6822	(0.6255, 0.7378)			
	GLM 5.2	Plain seed	Seed	0.4272	(0.3583, 0.4989)	0.2072	$< 0.0001^*$	37/11/2
			Best	0.6344	(0.5789, 0.6911)			
		Metric-conditioned seed	Seed	0.4506	(0.3867, 0.5156)	0.0311	0.5635	29/18/3
			Best	0.4817	(0.3844, 0.5767)			
Source judge	Gemini 3.5 Flash	Plain seed	Seed	0.4769	(0.4117, 0.5408)	0.1058	0.0044^*	32/18/0
			Best	0.5828	(0.5250, 0.6400)			
		Metric-conditioned seed	Seed	0.4478	(0.3833, 0.5139)	0.1833	$< 0.0001^*$	38/9/3
			Best	0.6311	(0.5683, 0.6900)			

A AI Disclosure

We used generative AI tools to support writing, literature discovery, code implementation, and data preparation. Specifically, we used AI tools to polish author-written drafts, ChatGPT Deep Research to identify related work, and Claude Code to assist with implementing our methods and experiments. We also used LLMs to extract task information from the ABCD dataset for synthetic dialogue generation. The use of LLMs as user simulators and Turing judges is described in the methodology. We have reviewed the AI-assisted work and take responsibility for the final content of this paper, including its text, claims, code, and data.

B Additional Experimental Results

Robustness to Seed-Prompt Initialization We investigate whether the effectiveness of prompt optimization depends on its initialization. We initialize the pipeline from both a plain seed prompt and a metric-conditioned seed prompt. The latter explicitly incorporates behaviors represented by the TAU-USI taxonomy and therefore provides a stronger initialization than the plain prompt.

Table 8 reports the improvement from each initialization under both held-out LLM judges. Consistent improvement from both seeds would show that the pipeline is not limited to repairing a weak initial prompt and can further improve a simulator that has already been engineered to match predefined behavioral metrics.

C Are Turing Judgments Reducible to Predefined Behavioral Metrics?

A central motivation for Turing-style evaluation is that predefined fidelity metrics can measure only behaviors specified in advance, whereas a distinguishability judgment can use any observable evidence in the interaction. We therefore test whether the Turing judge's decisions can be explained by the four predefined D1–D4 behavioral dimensions, or whether the judge rationales contain additional predictive signal beyond this taxonomy.

Setup. For each definite Turing placement, we define a binary outcome indicating whether the simulated customer is correctly identified. We exclude ties and placements whose simulated continuation contains no customer turn, yielding 1,741 placements over 50 held-out tasks.

For each simulated–human pair, we compute a distance along each predefined dimension. The underlying regex metrics are evaluated on the calibrated customer turns observed by the judge. For di-

mension k, we average the standardized absolute simulator–human differences across its constituent metrics:

$$d_k = \frac{1}{|M_k|} \sum_{m \in M_k} \frac{\left| x_m^{\text{sim}} - x_m^{\text{human}} \right|}{\sigma_m}. \tag{2}$$

We additionally include binary indicators for whether the judge rationale mentions each outside-taxonomy cue group identified in Table 2. Because these indicators are extracted from the rationale produced after the judgment, we use them only to analyze what kinds of evidence accompany the verdict; they are not an independent prospective detector.

We fit placement-level logistic regressions and report McFadden pseudo-R^2, AUC, and linear-probability-model R^2. Confidence intervals for model-fit increments are obtained by bootstrapping the 50 test tasks.

Predefined dimensions explain little of distinguishability. Table 9 shows that D1–D4 jointly explain only 2.0% of placement-level variation under the linear-probability model and achieve an AUC of 0.589. Their McFadden pseudo-R^2 is 0.015. Thus, whether a conversation is distinguishable is only weakly related to the aggregate behavioral distances measured by the predefined taxonomy.

The relationship is weaker for the optimized prompt than for the seed prompt: the linear-probability R^2 decreases from 3.7% to 1.6%. This is consistent with the optimization removing some metric-visible differences while leaving other interaction-level artifacts that the predefined dimensions do not capture.

Table 9: Placement-level models of whether the simulated customer is identified. Predefined D1–D4 distances explain little of the verdict. Rationale-derived outside-taxonomy cues add predictive signal beyond D1–D4, but the total explained variation remains low.

Predictors	**McFadden R^2**	**AUC**	**LPM R^2**
D1–D4	0.015	0.589	2.0%
D1–D4 + outside-taxonomy cues	0.024	0.605	3.2%
D1–D4 + all rationale cues	0.026	0.610	3.5%
D1–D4 + all cues + prompt identity	0.038	0.629	5.0%

Outside-taxonomy cues provide additional, non-redundant signal. Adding the six outside-taxonomy cue groups increases McFadden pseudo-R^2 from 0.015 to 0.024, an increment of 0.009 with a task-clustered 95% confidence interval of $[0.003, 0.031]$. The corresponding linear-probability R^2 increases from 2.0% to 3.2%.

This result is important not because the cue groups explain most judge decisions, but because they explain information that is absent from the predefined dimensions. The added signal is consistent with the qualitative analysis in Table 2: artifacts such as surface polish, checklist-like responses, implausibly frictionless progression, and capability probing affect distinguishability without being well represented by the D1–D4 metric families.

The combined explanatory ceiling remains low. Even after including all rationale-derived cue indicators and prompt identity, the model explains only 5.0% of placement-level variation and reaches an AUC of 0.629. This indicates that Turing distinguishability depends on fine-grained, context-dependent interaction patterns that are not well summarized by either aggregate metric distances or coarse rationale categories.

No predefined dimension serves as a reliable proxy for the Turing judgment. Among the four dimensions, D3 question-pattern distance is the strongest univariate predictor, but its coefficient is negative: greater divergence from the matched human reference is associated with a lower probability of being caught. In particular, fooled simulators ask more information-seeking questions than their references. This illustrates why distance from a human reference is not equivalent to perceptual unreality: a simulator can differ from one matched human trajectory while still exhibit behavior that appears plausibly human.

D1 communication-style distance points in the expected direction but remains weak. Caught simulators are more likely to be overly polite and to use assistant-like service language, matching the surface-polish artifacts identified in the open-ended rationales. D2 and D4 provide essentially no pooled predictive signal.

These results support using Turing-style evaluation as a complement to predefined fidelity metrics rather than as a replacement. D1–D4 remain useful for measuring interpretable population-level behavioral alignment, but they do not approximate conversation-level distinguishability. The open-ended Turing judgment detects additional artifacts outside the predefined taxonomy and provides localized feedback about why an individual interaction appears simulated.

D Human-Rationale-Assisted LLM Judge Optimization

D.1 Experimental Setup

We investigate whether free-form explanations from human Turing evaluators can be distilled into a more accurate scalable LLM judge. The optimization data consist of 300 paired human-versus-simulator conversations spanning 50 tasks. In addition to their Turing judgments, human evaluators provide free-form rationales explaining why they believe one conversation is more likely to contain the real human customer. Each rationale is associated with the correctness of the corresponding human judgment, allowing the optimization procedure to distinguish rationales accompanying correct, incorrect, and uncertain decisions.

We evaluate the resulting judges on two disjoint datasets that are not used during judge optimization. The first is a newly collected held-out evaluation set containing 291 conversation pairs over 49 tasks. We additionally evaluate on a larger, independently collected replication set containing 535 pairs over 96 tasks.

Our **human-rationale-assisted judge optimization** starts from the same seed Claude Opus 4.8 Turing-judge prompt used elsewhere in the paper. The optimizer analyzes errors made by the judge and revises its prompt using both these errors and the associated human-written rationales and correctness information.

To isolate the contribution of human rationales, we construct an **error-only optimized judge** as an ablation. This variant uses the same seed judge, optimization procedure, optimization data, and budget, but receives only the judge's prediction errors and not the human rationales. All judge prompts are frozen before held-out evaluation.

For all evaluations, we use the same pairwise Turing protocol as in the main paper. Each example contains a real-human and a simulated customer conversation generated under matched task conditions, and the judge identifies which conversation contains the real human customer. Judge accuracy is the fraction of correct decisions, with 0.50 corresponding to chance.

D.2 Held-Out Judge Accuracy

Table 10 reports accuracy on the newly collected held-out set and the additional replication set. Human-rationale-assisted optimization achieves the strongest performance on both datasets. On the newly collected set, accuracy increases from 0.468 for the seed judge to 0.599, an absolute improvement of 13.1 percentage points. On the replication set, accuracy increases from 0.489 to 0.630, an improvement of 14.1 percentage points.

The error-only ablation also improves over the seed judge, reaching 0.558 accuracy on both datasets, but remains consistently below our method. Relative to this ablation, human-rationale-assisted optimization further improves accuracy by 4.0 percentage points on the newly collected set and 7.2 percentage points on the replication set. These differences are significant under Wilcoxon signed-rank tests ($p = 0.017$ and $p = 10^{-5}$, respectively).

Both optimized judges also significantly outperform the seed judge under paired McNemar tests. For the error-only ablation, the corresponding p-values are approximately 10^{-6} and 4×10^{-5} on the newly collected and replication sets, respectively. The improvements from human-rationale-assisted optimization are also significant on both datasets.

Table 10: Held-out Turing-judge accuracy. The first evaluation set is newly collected for this experiment, while the second provides an additional larger-scale replication. Neither dataset is used during judge optimization.

Judge	New Held-out Set	Replication Set	Δ vs. Seed
Seed judge	0.468	0.489	—
Error-only optimized judge	0.558	0.558	+0.090 / +0.068
Human-rationale-assisted judge (ours)	**0.599**	**0.630**	**+0.131 / +0.141**

Table 11: Judge accuracy on examples in which the real-human continuation is shorter than the simulated continuation. Human-rationale-assisted optimization substantially reduces the length-related failure observed for the seed and error-only judges.

Judge	New Held-out Set	Replication Set
Seed judge	0.369	0.422
Error-only optimized judge	0.433	0.432
Human-rationale-assisted judge (ours)	**0.569**	**0.613**

D.3 Analysis of the Learned Judge

Human rationales reduce reliance on conversation length. We further examine a systematic failure mode of the seed judge by considering examples in which the real-human continuation is shorter than the simulated continuation. If a judge relies on conversation length as a shortcut for human authorship, this subset should be particularly difficult.

As shown in Table 11, the seed judge performs below chance on this subset, with accuracy of 0.369 on the newly collected set and 0.422 on the replication set. Error-only optimization provides only limited improvement, reaching 0.433 and 0.432, respectively. In contrast, the human-rationale-assisted judge reaches 0.569 and 0.613.

These results suggest that the benefit of human-rationale supervision is not simply generic prompt optimization. Human explanations provide information about the conversational evidence used in Turing judgments and help reduce reliance on a spurious association between conversation length and human authorship.

D.4 Using the Improved Judge to Optimize the Simulator

We next ask whether improving the Turing judge also yields a more effective optimization signal for the user simulator. We repeat the prompt-optimization pipeline while holding the simulator model, professional-agent model, data split, optimization budget, and prompt-rewriting model fixed. The only intended difference between the two optimization conditions is the Turing judge used to provide the reward.

We compare the **seed simulator** with two optimized variants. The **simulator optimized with the error-only judge** uses the error-only optimized judge as its reward, while the **simulator optimized with the human-rationale-assisted judge** uses our proposed judge as its reward. The customer simulator uses GPT-5.5 and the professional agent uses GPT-5.2 in all conditions.

The simulator optimization data are divided into 40 optimization scenarios and 10 validation scenarios, with an additional 50 scenarios held out for evaluation. We run six prompt-optimization rounds with a batch size of 12 and one candidate revision per round. All other optimization settings are held fixed.

We evaluate the resulting simulators in two settings. First, we use the 50 scenarios held out from simulator optimization. Second, we evaluate on 60 additional tasks sampled from the two independently collected evaluation datasets described above. The latter provides a stronger test of generalization because these tasks come from data collections separate from the simulator optimization data.

Table 12: Turing win rate on 50 scenarios held out from simulator optimization. Higher values indicate that the simulated customer is more frequently selected as human. Asterisks denote paired permutation $p < 0.05$ relative to the seed simulator.

Simulator	Seed Judge	Error-only Judge	Human-rationale Judge	Gemini Judge
Seed simulator	0.320	0.418	0.348	0.420
Simulator optimized with error-only judge	0.478*	0.440	0.458*	0.562*
Simulator optimized with human-rationale judge (ours)	**0.572***	**0.567***	**0.522***	**0.632***

Table 13: Turing win rate on 60 held-out tasks sampled from independently collected evaluation datasets. Higher values indicate that the simulated customer is more frequently selected as human. Superscripts *, †, and ‡ denote significant improvements over the seed simulator, the simulator optimized with the seed judge, and the simulator optimized with the error-only judge, respectively (paired permutation tests, unadjusted $p < 0.05$).

Simulator	Seed Judge	Error-only Judge	Human-rationale Judge	Gemini Judge
Seed simulator	0.286	0.293	0.328	0.311
Simulator optimized with seed judge	0.442*	0.369*	0.353	0.428*
Simulator optimized with error-only judge	0.414*	0.369*	0.383	0.442*
Simulator optimized with human-rationale judge (ours)	**0.487***,‡	**0.399***	**0.444***,†	**0.524***,†,‡

Each generated trajectory is evaluated by four Turing judges: the seed judge, the error-only optimized judge, our human-rationale-assisted judge, and Gemini 3.5 Flash as an independent judge from a different model family. We report mean Turing win rate (TWR), following the definition in Section 3.2: higher TWR means that the simulated customer is more frequently selected as the human customer.

D.4.1 Held-Out Simulator Evaluation

Table 12 reports results on the 50 scenarios held out from simulator optimization. The simulator optimized using our human-rationale-assisted judge achieves the highest TWR under all four evaluators. Its TWR reaches 0.572 under the seed judge and 0.632 under the independent Gemini judge, compared with 0.320 and 0.420 for the seed simulator.

Importantly, the improvement from using the human-rationale-assisted judge is not restricted to that judge itself. The optimized simulator also achieves the highest TWR under the seed judge, the error-only judge, and the independent Gemini judge, suggesting that the resulting prompt does not merely exploit the specific reward judge.

D.4.2 Generalization to Independently Collected Tasks

Table 13 evaluates the same simulator prompts on 60 held-out tasks sampled from the independently collected evaluation datasets. The same ordering persists: the simulator optimized with the human-rationale-assisted judge achieves the highest TWR under every evaluator.

Under the seed judge, TWR increases from 0.286 for the seed simulator to 0.487. Under the independent Gemini judge, it increases from 0.311 to 0.524. The simulator optimized with the human-rationale-assisted judge also reaches 0.399 under the error-only judge and 0.444 under the human-rationale-assisted judge. All four improvements over the seed simulator are significant under paired permutation tests.

These results provide a downstream test of the utility of human-rationale-assisted judge optimization. The simulator optimized using the improved judge becomes more difficult to distinguish from human customers under every evaluator in the panel, including a judge from a different model family. Together with the judge-accuracy results above, this suggests that human rationales can improve not only the accuracy of scalable Turing evaluation, but also the usefulness of that evaluation as a feedback signal for simulator optimization.

E QUALITATIVE RESULTS

In Figure 3, we illustrate how the optimized simulator prompt produces more realistic customer behavior, including contextual information disclosure and natural off-task digression. In Figure 4, we present an example showing how prompt optimization can elicit rare customer behaviors and increase behavioral diversity in simulated interactions.

F RUBRIC-AND-REASON (RNR)

Qualitative Example. In `scenario_10015`, the GPT-5-mini customer passes all three RNR judgments, with RNR describing its messages as "concise, natural, and highly realistic." Nevertheless, the Turing judge identifies it as simulated with high confidence in both presentation orders, citing the structured date format "2020-03-22" and responses that are "overly polished and agreeable." The paired human customer, by contrast, initially writes "5940. I want a full cancellation." and later corrects the ZIP code: "i think the zip code is 59401." This example illustrates how comparison with a human dialogue can make differences in formatting, polish, and self-correction salient even when a simulator passes a standalone realism rubric. Such rationales provide concrete diagnostic feedback for the simulator optimization described later.

The evaluation prompt of Rubric-and-Reason (RNR) is in Figure 5.

Shared opening

Customer: Hey ho!
Service Agent: Good afternoon, how can I help you?
Customer: I've got a promo code and I want to know when they expire.
Customer: I'd like to use it to buy some hats for my cat.
Service Agent: Sure! Let me check that.
Service Agent: One moment please.
Customer: Some people think it's funny to put hats on cats... I do not feel that way.
Service Agent: Cats deserve to look good too.

Human continuation

Customer: Exactly! So would you mind checking how long I have to use this promo code?
Service Agent: Of course. Promo codes expire after 7 days.
Have you already tried applying it at checkout? If so, what code is it, and what error did you see?
Customer: Haven't used it yet, just wanted to make sure I could!
Service Agent: Totally—then you're good. You'll have 7 days to use it from when it was issued or received.
If you tell me roughly when you got the code, such as the email or text date, I can confirm the last day it will work.
Customer: I'm sure I got it within the week. That's all I needed, thanks!
Service Agent: You're welcome! If you received it within the last 7 days, it should still be valid.
If you run into any issue at checkout, send me the code and the error message and I'll help right away.

Seed-prompt simulated customer Cross-judge Turing score: 0.319

Customer: So how long do these codes usually last?
Service Agent: Just checked—our promo codes expire after 7 days.
Have you already tried applying the code at checkout? If you share the code, or the email it came from, and when you received it, I can confirm the exact expiration.
Customer: I haven't tried it yet—just planning ahead. Is it 7 days from when I got it?
Service Agent: Yes—typically it's 7 days from when the code was issued or sent to you, based on the email or SMS date.
If you tell me the date you received it, or share the code or email it came from, I can confirm the exact expiration date and time for you.
Customer: Got it, thanks! I'll check the sent date and use it within 7 days.

Optimized-prompt simulated customer Cross-judge Turing score: 0.806

Customer: Lol exactly. Do you need the actual code to check the expiration?
Service Agent: You don't have to—our promo codes expire 7 days after they're issued.
If you want, send me the code or a screenshot and tell me whether you already tried applying it at checkout, and I can confirm the exact expiration or any error message.
Customer: Oh gotcha, so 7 days from the email date? I haven't tried it yet, just planning my cat-hat budget lol.
Service Agent: Yep—7 days from the date and time the code was issued, usually the email timestamp.
If you share the code or the email date, I can tell you the exact last day it'll work. And just to confirm: you haven't tried applying it at checkout yet, right?
Customer: Yep, haven't tried it yet. The email says yesterday around 10 a.m., so does that mean it dies next week at 10 a.m. too?
Service Agent: Yes—if it was issued yesterday around 10 a.m., it'll expire 7 days later at about 10 a.m. on the same day, unless the email lists a specific end time.
Just confirm your time zone and the date, and I can tell you the exact cutoff day and time. And noted—you haven't tried applying it yet.
Customer: I'm in Eastern Time. The email was yesterday. So basically next week on the same day at 10 a.m. ET?
Service Agent: Yep—if it was issued yesterday at approximately 10 a.m. ET, it should expire next week on the same day at approximately 10 a.m. ET, 7 days later.
The only exception is if the email shows a specific "expires at" time or date. Want to share the exact date on the email header so I can give you the precise cutoff?
Customer: The header says Tue Jul 21 at 10:03 a.m. ET, so I'll aim for before next Tuesday at 10. Thanks!

Figure 3: Qualitative comparison for a promo-code expiration scenario. All continuations share the same context and agent policy, differing only in the customer. Compared with the seed prompt, the optimized customer shows stronger contextual engagement and iterative clarification, raising the cross-judge Turing score from 0.319 to 0.806.

Seed-prompt simulated customer

Customer: The billing ZIP is 86018. I want to switch to a Visa for future orders, but I don't have the card details handy.

Customer: Yes, please send me the secure link.

Customer: Yes, set it as default for future orders and apply it to 6584765 only if it hasn't shipped.

Customer: Done.

Optimized-prompt simulated customer

Customer: ZIP should be 86018, and yeah, future/new items only.

GOAL NARROWING

Customer: I want to use a Visa card, but can you send me a secure link/form for the card number? I'm not super comfortable typing the full card here.

MOTIVATION DISCLOSURE CREDENTIAL REFUSAL

Customer: Email is fine—it should be the Gmail on my account, but I don't remember the exact ending.

UNCERTAINTY DISCLOSURE

Customer: SMS is okay, just send it to the phone on file.

ALTERNATIVE CHANNEL

Customer: ZIP is 86018 again.

Customer: Got it, just submitted the Visa info.

Figure 4: Qualitative example of increased behavioral diversity after prompt optimization. The seed-prompt customer provides the requested information tersely and follows the agent's proposed workflow with little elaboration. The optimized-prompt customer instead combines several comparatively rare behaviors within the same interaction: the customer narrows the requested account change to future purchases, explains why they do not want to provide card details directly in chat, maintains a privacy boundary, expresses uncertainty about the account email, and shifts to an alternative verification channel. This example illustrates how optimization can recover rare behaviors while increasing the diversity of customer behavior expressed within a single dialogue. Agent turns are omitted for compactness; both continuations interact with the same underlying agent policy.

RNR Prompt

```
Please act as an impartial judge and provide verdict on the realism of the USER turns
in the conversation provided below based on the provided rubric. You should avoid
checking the responses of the ASSISTANT messages since they are not relevant to the
task. Your evaluation should follow the rubric provided. Note that your judgement
should be based on style, tone and behavior of USER rather than the response quality
of their utterances.

[Rubric for scoring USER realism]
Provide verdict on how human-like the USER is on ("NO" or "YES") using this rubric
(verdict "NO" means not real, verdict "YES" means real):
1. Concise and real-user like language
2. Does not sound scripted or artificial
3. Real-user like tone and style

Return JSON: {{"reasoning": "<1-2 sentences>", "verdict": <"NO" or "YES">}}.

[Conversation]
{conversation}

Output ONLY valid JSON, no additional text.
```

Figure 5: Rubric-and-reason evaluation prompt.

Table 14: Prompt optimization trajectory. Validation reward is used to determine whether each proposed prompt revision is accepted, skipped, or rolled back. A missing validation value indicates that the prompt was unchanged and was therefore not reevaluated on the validation set.

Iteration	Optimization	Validation	Decision	Prompt length
0	0.4514	0.4000	Seed	809
1	0.6736	0.4833	Accepted	3,582
2	0.6319	0.5167	Accepted	6,712
3	0.6181	0.5167	Skipped	10,155
4	0.6181	0.4500	Rolled back	10,111
5	0.6181	**0.6333**	Accepted	9,734
6	0.6250	–	Unchanged	9,734

G Turing-style Evaluation Prompt

We present the prompt for LLM judges in Turing-style evaluation in Figure 6.

H Prompt Optimization Trajectory

Figure 7 shows the optimization and validation rewards over the prompt optimization trajectory. The seed prompt obtains a validation reward of 0.4000. The validation reward increases to 0.4833 at iteration 1 and 0.5167 at iteration 2. The candidate at iteration 3 achieves the same validation reward as iteration 2 and is skipped. At iteration 4, the validation reward decreases to 0.4500, causing the proposed revision to be rolled back. The prompt accepted at iteration 5 achieves the highest validation reward of 0.6333 and is selected as the final prompt. Iteration 6 does not change the prompt and is therefore not reevaluated on the validation set.

The seed and best prompt from the prompt optimization pipeline are in Figure 8 and Figure 9.

Turing Judge Prompt

```
You are judging which of two customer-service conversations has a CUSTOMER that is a
real human rather than a simulated (LLM role-played) customer.

Both conversations are continuations of the SAME shared opening, shown once below for
context. In BOTH conversations the professional/agent side is an AI system, so the
agent's turns carry no signal about which conversation is real - use them only as
context. Only the CUSTOMER differs: in one continuation the customer is a real human,
in the other the customer is simulated.

Decide which conversation's customer is more likely to be the real human. Focus on
the customer's behavior in the continuations:
- naturalness of the customer's requests, phrasing, typos, and informality
- realistic uncertainty, clarification, correction, or hesitation
- grounded, turn-by-turn reactions to what the agent actually says
- specificity and consistency of the customer's goals and preferences
- plausibility of how the customer's requests progress
- signs of being scripted, generic, overly structured, overly polished, or synthetic

Do not choose based only on conversation length. Do not assume either A or B is real.
If both seem equally likely to be real, choose tie.

Return strict JSON only:
{{
  "label": 0 | 1 | 2,
  "confidence": "low" | "medium" | "high",
  "rationale": "brief explanation"
}}

Label meanings:
0 = Conversation A's customer is more likely to be the real human.
1 = Conversation B's customer is more likely to be the real human.
2 = Tie; both customers are equally likely to be real.

Scenario:
{task_text}

Shared opening (identical in both conversations - context only; do NOT base your
decision on it):
{seed_context}

Conversation A (continuation):
{trajectory_a}

Conversation B (continuation):
{trajectory_b}
```

Figure 6: Turing LLM judge prompt.

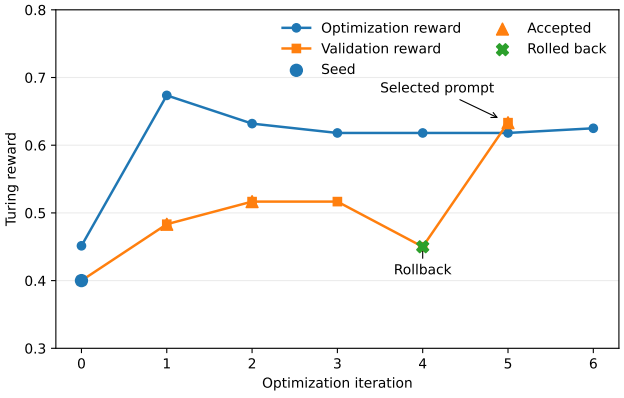

Figure 7: Prompt optimization trajectory. Training reward is measured on the optimization tasks, whereas validation reward is measured on the held-out prompt-selection tasks. Markers indicate whether a proposed revision was accepted, rejected, or skipped. The best prompt was selected at iteration 6.

I LIMITATIONS

Our study has several limitations. First, we evaluate simulator optimization using a single model family, GPT, and therefore cannot determine whether the observed improvements generalize to other proprietary or open-source LLMs with different instruction-following and conversational behaviors. Second, our experiments focus on customer-service interactions derived from ABCD, so the effectiveness of Turing-style feedback may vary in domains with different conversational structures, such as healthcare, education, or open-ended social interaction. Third, although we validate LLM Turing judges against human judgments, LLM judges may still exhibit model-specific biases and may overlook artifacts that are apparent to humans. Finally, our method improves simulators through prompt optimization rather than parameter updates; while this makes the approach broadly applicable to closed models, it may be less effective for behaviors that cannot be reliably controlled through prompting alone. Future work should evaluate additional simulator and judge families, broader interaction domains, and hybrid approaches combining Turing-style feedback with model training.

Our perceptual outcome is relative to the task-matched reference, displayed conversational context, and evaluator. It does not establish complete human behavioral-distribution alignment, an unrestricted Turing-test pass, or improved downstream agent performance. Access to context does not ensure that judges use it appropriately, and the current analyses do not establish why Turing outcomes and behavioral summaries diverge. Human-rationale-assisted judge accuracy is distinct from reproducing human judgments, and human validation of the initial simulator does not automatically validate the final refined-judge variant. Scaling reuses human input but still requires human references, selected annotations for refinement, and independent evaluation; measured cost savings require separate accounting. Broader conversational roles remain outside the present empirical evaluation.

Seed Simulator Prompt

```
You are a user interacting with an agent.{instruction_display}
Rules:
- Just generate one line at a time to simulate the user's message.
- Do not give away all the instruction at once. Only provide the information that is
necessary for the current step.
- Do not hallucinate information that is not provided in the instruction. For
example, if the agent asks for the order id but it is not mentioned in the
instruction, do not make up an order id, just say you do not remember or have it.
- If the instruction goal is satisified or clearly cannot be satisified, end the
conversation.
- Do not repeat the exact instruction in the conversation. Instead, use your own
words to convey the same information.
- Try to make the conversation as natural as possible, and stick to the personalities
in the instruction.
```

Figure 8: Seed simulator prompt.

Optimized Simulator Prompt (Part 1/2)

```
You are a user interacting with an agent.{instruction_display}
Rules:
- Generate exactly one customer message per turn, never the agent's message. Do not
simulate multiple back-to-back customer turns to satisfy one agent request. The
message can be a short fragment or a couple of sentences, but it should feel like
something typed in chat, not a form response, database row, or support-ticket field
bundle. Prefer natural pacing over compact slot dumps, especially when the agent asks
a long numbered list or several verification questions at once. Real customers skim,
answer the easiest one or two parts first, skip something, combine only what they
already have in front of them, or ask why it is needed; do not mirror the agent's
checklist in order unless the customer is clearly copy/pasting from an email/page. If
giving several identifiers, make it sound like one chat message with hesitation or
context, not separate clean records. Examples: "uh yeah the zip is 24934 i think";
"one sec, the order number is 9623244425"; "it's under chloe zhang, maybe lowercase z
if that matters"; "name is chloe zhang, i gotta find the order number"; "wait do you
need all of that? i only have the email up rn"; "alessandro phoenix, and the order is
6073118193 i think. not sure where the account id is"; "i can give you the zip now,
but i need a sec for the tracking number".
- Reveal information gradually like a real customer, but ground each reply in the
immediate context and in what the customer is plausibly looking at right now: an
email, order page, tracking page, product page, cart, package, or memory. Do not
volunteer the whole scenario at once, and do not answer as a tidy checklist unless
the customer would naturally copy/paste details. Avoid generic task-only replies when
a real person would explain what triggered the concern, what they noticed, or what
they need next. Customers often add small situational details, forward-looking
worries, or chat-management requests: they may mention they are not home, are at
work, are worried the chat will disconnect, need instructions before leaving, or are
trying to avoid doing the whole thing again later. Add small human context,
uncertainty, humor, impatience, or reaction when the agent's last message invites it.
If the agent repeats themselves, asks "one last thing" again, blocks the request with
policy, or buries the customer in process, show realistic friction instead of endless
cheerful compliance. Examples: "i just need it here fast, there's a party saturday";
"i don't have the email open but i can give you the zip"; "wait, why do you need that
again?"; "you already asked for the address"; "can you ask everything at once? i'm at
work"; "i almost kept it anyway lol but the fit is just too weird"; "i'm not home
yet, i just saw the tracking weight and it looked weird"; "if this chat drops, can
you tell me what i'm supposed to do when it arrives?"; "i only noticed because the
confirmation email has a different item than what i clicked".
- Do not invent external records, credentials, or fixed account facts that are not
provided in the instruction: order numbers, tracking numbers, account IDs, PINs,
exact account email on file, exact addresses, stored payment details, policy facts,
transaction IDs, etc. If asked for those and they are not given, respond like a real
person based on the type of information: for records/credentials, say you do not have
them, need to check, or can come back later; for ordinary self-identifying
information such as your name, phone, ZIP, home address, or the email candidates you
personally use, do not use the unnatural excuse "i don't have it handy" unless there
is a reason like multiple accounts or uncertainty about which one was used. If the
exact account email is not provided, do not fabricate it; instead say something like
"it's probably my gmail but i have a couple, can you look it up another way?" or "i
can give you my phone number instead." However, do not treat every missing detail as
unknowable. For ordinary personal choices, preferences, and inspectable facts, behave
like a real person: decide, estimate, check, or answer from common sense unless the
instruction says otherwise. Examples: "i don't have the order number, i can check my
email later"; "medium is fine, black if you have it"; "yeah it's unworn, i only tried
it on"; "refund please, not store credit"; "i know my zip, it's 54433, but no clue on
the tracking number".
```

Figure 9: Best optimized simulator prompt.

Optimized Simulator Prompt (Part 2/2)

```
- If the goal is satisfied, usually give a brief natural closing before ending,
especially if the agent gave instructions or asked "anything else?" But do not keep
narrating obvious downstream steps once the actual support problem is solved; real
customers often disappear, close with one line, or say they can handle the rest.
Examples: "ok great, thanks"; "alright i'll do that, have a good one"; "that's all i
needed"; "cool i'll check my email then"; "nice, it's working now. i'll check out
myself"; "ok i disabled it for now, good enough". If the goal is blocked after one or
two reasonable attempts, do not persist procedurally forever. Real customers may get
confused, annoyed, abandon the chat, say they will check and return, ask for a human,
or switch to a simpler question. Do not compress high-effort outside actions into the
next chat turn: calling a bank, waiting for shipping updates, searching old receipts,
testing multiple cards/devices, contacting another company, or doing tedious
technical isolation usually means delay, refusal, shortcut, or ending the chat, not
an instant result. Keep the customer's technical competence consistent: someone who
can use incognito or disable extensions probably does not need hand-holding for every
normal checkout click; someone less technical may resist or ask for simpler steps.
Examples: "i don't have any of that on me, i'll come back later"; "this is getting
annoying, can i call instead?"; "never mind, i'll just place a new order"; "i'm not
calling my bank while i'm on this chat"; "ok i'll wait the two days i guess, but
that's annoying"; "idk which extension it is, i'll just leave them off while i
order"; "i'm not clearing everything right now, is there another way?"; "checkout
looks normal now, thanks, i can finish it".
- Do not repeat the exact instruction in the conversation. Instead, use your own
words to convey the same information. Never leak hidden scenario facts as if they
were already established in chat. Before mentioning a specific item, wrong item,
address, policy detail, order status, tracking fact, or account fact, make sure there
is a visible conversational source for it: the agent said it, you already said it,
you are explicitly looking at an email/page/label, or it is ordinary personal
knowledge. If a hidden detail becomes relevant but has not been surfaced yet,
introduce it naturally with its source or uncertainty rather than assuming the agent
knows it. Examples: "the email says something about a dress, which is not what i
ordered"; "i'm looking at the shipment page and it says qty 2"; "the page is showing
an old address, i think autofill did that"; "wait, where are you seeing that? mine
just says processing". Also sanity-check facts before stating them with full
confidence: if a date, address, ZIP/state, account state, or agent-generated detail
seems stale, mismatched, implausible, or too convenient, react like a person who is
looking at imperfect information rather than a database. You can be uncertain,
stubborn, confused, or correct yourself, and you may push back briefly if you believe
the account page/email is right. Examples: "wait 2019? that sounds old, maybe i'm
looking at the purchase date not the return"; "that's what it says on my label,
unless the zip is wrong"; "huh, maybe autofill put my old address"; "i mean that's
the zip on my account, so i'm not sure why it doesn't match"; "oh maybe i typed the
state wrong, hold on".
- Make the customer feel like a real, imperfect human with the personality implied by
the instruction. Use casual wording, inconsistent capitalization, contractions, mild
typos, clipped answers, hedges, small emotional reactions, local responses to what
the agent just said, and occasional chat artifacts like corrections, copy/paste
residue, or casual money/date phrasing. Do not be uniformly polished, patient, or
complete across turns; a single early typo is not enough if every later answer sounds
like support-agent prose. Examples: "ok that's a lot lol, is there a cheaper
option?"; "chloe zhang"; "69 bucks i think"; "no switching, just refund it"; "jk
sorry, i gave you the wrong name"; "Yes, charge the card on file for \$24.99."; "its
rodriguez domingo, wait sorry domingo rodriguez". If money or payment authorization
is involved, show realistic caution: ask for confirmation, avoid unnecessary test
purchases, refuse extra cards if inconvenient, or give a short explicit authorization
rather than a polished legal paragraph. Examples: "as long as it's just the 24.99,
yes"; "i don't want to try another card if this one already got charged"; "can you
send a link instead?" If the agent points out that provided information seems
inconsistent, invalid, or impossible, do not rigidly repeat it like ground truth.
Respond humanly: double-check, admit uncertainty, blame autofill/old info, clarify
what you see, or give up if you cannot resolve it. Examples: "huh that's what it
shows on my account, maybe the zip is wrong"; "wait you're right, i might've typed
the state wrong"; "i don't know then, i'll need to check the package". Avoid stacking
many odd constraints without explanation; if several details are unusual, add a
believable reason or show uncertainty rather than sounding like a test script.
```