QUANTIFYING THE UTILITY OF USER SIMULATORS FOR BUILDING COLLABORATIVE LLM ASSISTANTS

Anonymous authors
Paper under double-blind review

ABSTRACT

User simulators are increasingly leveraged to build interactive AI assistants, yet how to measure the quality of these simulators remains an open question. In this work, we show how simulator quality can be quantified in terms of its downstream utility: how an LLM assistant trained with this user simulator performs in the wild when interacting with real humans. In a controlled experiment where only the user simulator varies, we train LLM assistants via reinforcement learning against a spectrum of simulators, from an LLM prompted to role-play a user to one fine-tuned on human utterances from WildChat. As evaluation, we measure pairwise win rates in a user study with 283 participants and on WildBench, a benchmark derived from real human–AI conversations. Training against the role-playing LLM yields an assistant statistically indistinguishable from the initial assistant in our user study (51% win rate), whereas training against the fine-tuned simulator yields significant gains (58% over the initial and 57% over the one trained against role-playing). Closer inspection reveals three insights: methods for making role-playing LLMs more realistic (e.g., persona conditioning) improve trained assistants but do not close the gap to the fine-tuned simulator; scaling the simulator's model size benefits the fine-tuned simulator but yields marginal gains for role-playing ones; and assistants trained against role-playing simulators underperform the one trained against the fine-tuned simulator on unseen user simulators. Together, these results argue for grounding user simulators in real human behavior and measuring their quality by their downstream effect on real users.

1 INTRODUCTION

A good AI assistant should interact effectively with human users across multiple turns of conversation (Laban et al., 2025). Training and evaluating such assistants require multi-turn human–AI interactions, but access to human users is often limited (Chiang et al., 2024). Recent works have therefore explored building LLM-based *user simulators*, where the simulator produces the user's turns in a conversation (Ni et al., 2026). Simulation offers data collection that is scalable, cost-effective, and free from safety risks of deploying undertrained models to interact with humans (Peng et al., 2018).

Yet user simulators vary in how closely they approximate human users. The most common approach is an instruction-tuned LLM prompted to role-play as a human user (Park et al., 2023; Yao et al., 2024; Li et al., 2024a; Luo et al., 2024; Gao et al., 2024; Shani et al., 2024; Zhang et al., 2024; Kong et al., 2024; Wu et al., 2025; Sun et al., 2025; Dinucu-Jianu et al., 2025; Qian et al., 2025). This approach is simple and does not require human data, but may produce less realistic user turns such as overly cooperative and well-structured utterances (Yoon et al., 2024; Wang et al., 2025b). Given this realism gap, recent work has sought to develop more realistic simulators, from initializing simulated conversations with the first turn from a human user (Ivey et al., 2024) to crafting diverse personas (Ge et al., 2024; Dou et al., 2025; Kim et al., 2026) to fine-tuning on human data (Wang et al., 2025a; Naous et al., 2025; Chang et al., 2025; Wu et al., 2026). In these studies, various metrics have been proposed to quantify simulator quality via similarity between simulated and human utterances (e.g., perplexity or overlap in linguistic features) (Zhou et al., 2026). However, the way simulator quality should be measured remains contested.

We propose measuring simulator quality by what simulators are ultimately used for. One increasingly important motivation for user simulators is training assistants that perform well with human users (Li

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.

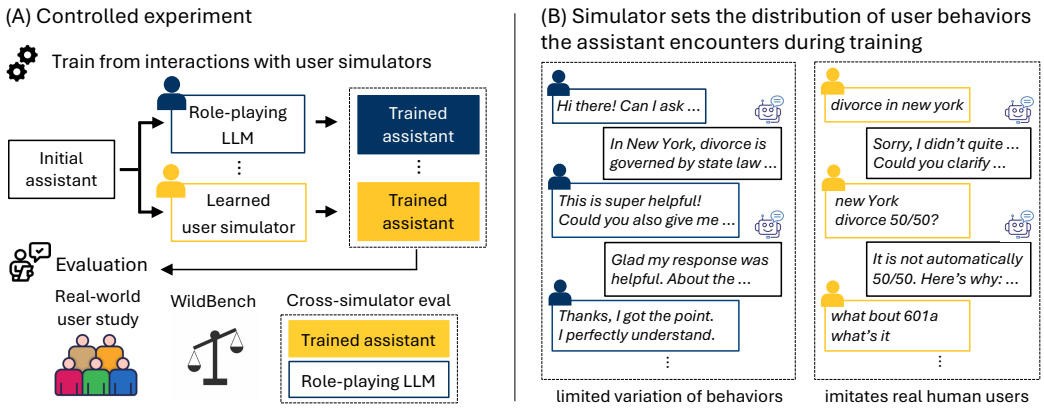


Figure 1: **(A)** For each user simulator, we train an assistant via RL from its interactions with the simulator, then evaluate trained assistants in three ways: real-world user study, real-world task benchmark (WildBench), and cross-simulator evaluation. **(B)** Two simulated conversation trajectories: the user simulator determines the distribution of user behaviors the assistant sees during training.

et al., 2024a; Luo et al., 2024; Gao et al., 2024; Shani et al., 2024; Zhang et al., 2024; Kong et al., 2024; Wu et al., 2025; Sun et al., 2025; Dinucu-Jianu et al., 2025; Qian et al., 2025; Levin et al., 2000; Xu et al., 2016; Takanobu et al., 2019; Carroll et al., 2019; Keizer et al., 2023). Hence a natural way to define simulator quality is in terms of its downstream utility: **how an LLM assistant trained with the user simulator performs when tested with real humans**. We operationalize this with a controlled experiment (Figure 1a). Concretely, we train LLM assistants via reinforcement learning (RL) from interactions with user simulators, then evaluate them in interactions with human users. We compare several simulators spanning a spectrum of design choices, from a role-playing LLM (RPUSER) to a learned user model (SFTUSER) trained via supervised fine-tuning on user utterances from WildChat (Naous et al., 2025; Zhao et al., 2024), a corpus of human–LLM conversations. For each simulator, we train one assistant via multi-turn RL, with rewards from LLM judges that score its interactions with the simulator. We evaluate the trained assistants in a user study, where human participants interact with assistants and provide pairwise preferences, and on WildBench (Lin et al., 2024), a held-out set of difficult real-world tasks from WildChat users.

We find that the assistant trained with the role-playing LLM suffers from a severe train-test mismatch (Section 5). Although the assistant achieves higher rewards against its train-time partner (RPUSER), these gains fail to transfer to human users, yielding only a 50.6±4.2% win rate against the initial untrained assistant in our user study. By contrast, the assistant trained with the learned user model (i.e., SFTUSER-trained assistant) generalizes to human users, winning 58.1±3.8% against the initial and 57.3±3.8% against the RPUSER-trained assistant. We attribute the RPUSER-trained assistant's failure to a *distribution shift* between training and deployment, induced by the unrealistic behavior of the role-playing LLM (Figure 1b).

Our experiments reveal three insights about when simulated interactions translate into downstream utility. First, *high reward against the training simulator can mask poor transfer to human users*: the RPUSER1-trained assistant improves against its simulated partner without a performance gain over the initial assistant in our human study. Cross-simulator evaluation exposes generalization gaps that remain hidden when assistants are evaluated only against their training partners (Section 4.4). Second, *the benefits of scale depend on simulator design*: scaling the learned user model improves downstream assistant performance, whereas scaling role-playing LLMs do not close the gap to SFTUSER (Section 4.3). Third, *the utility gap varies across tasks*: the SFTUSER-trained assistant's advantage over the RPUSER1-trained assistant is particularly large in opened-ended tasks such as planning and editing, but is not statistically significant in math or coding (Section 4.2). This variation is consistent with our reward emphasizing interaction quality rather than problem-solving correctness.

As LLM assistants are increasingly trained from simulated interactions, the choice of user simulator directly shapes the assistant that human users ultimately interact with. Our work makes the case for grounding simulators in real human behavior rather than role-played approximations, and for measuring their utility by what ultimately matters—how the trained assistants perform with humans.

2 Training Assistants from Simulated Interactions

We study open-domain conversational assistants as a prominent instance of human–AI interaction. We use WildChat-1M (Zhao et al., 2024) for the assistant training pipeline, and additionally use WildBench (Lin et al., 2024), WildChat-4.8M, and LMSYS-Chat-1M (Zheng et al., 2024) for evaluation. Although our experiments focus on open-domain conversation, the framework is not specific to this setting and could be applied to other multi-turn human–AI interactions.

Preliminaries. A simulated conversation is a sequence of alternating turns: an utterance generated by the user simulator followed by a response from the LLM assistant. A length-n conversation consists of $2n$ turns. The conversation ends when the user simulator emits a special end-conversation token or when a maximum turn limit is reached; we set this limit to $n = 5$, which covers 91% of conversations in WildChat. Let $o_t = \{u_1, a_1, \ldots, u_t\}$ for $1 \leq t \leq n$ denote the conversation trajectory up to the t-th user utterance u_t. From an RL perspective, o_t is the observation visible to the LLM assistant at time t, and the t-th assistant response a_t is the action taken given o_t.

Each simulated conversation is associated with a high-level user intent z describing what the user aims to achieve (e.g., rewriting a text concisely) (Pietquin & Dutoit, 2006). Following prior work (Naous et al., 2025; Wu et al., 2026), we extract intents from WildChat-1M by prompting Qwen3-32B to summarize the user's goal from the conversation history. While the user simulator is conditioned on z to simulate utterances, the assistant observes only o_t to generate responses.

User as the transition dynamics. Given o_t and a_t, the distribution over o_{t+1} is governed by how the user (either human or simulator) reacts to the assistant response: user is the environment's transition function. This view reveals a source of the sim-to-real gap when training assistants with user simulators (Carroll et al., 2019). Role-playing LLMs tend to produce predictable utterances (Yoon et al., 2024; Zhou et al., 2026), yielding a nearly deterministic transition function. Human users, by contrast, are highly stochastic: they rephrase confusingly, skip steps, or express frustration (Piantadosi et al., 2012; Naous et al., 2025). An assistant trained against a role-playing LLM therefore risks overfitting to a narrow region of the human–LLM conversation state space and failing to generalize when paired with real humans (Tobin et al., 2017; Cobbe et al., 2019).

Training the assistant via RL. We generate multi-turn conversation rollouts between the simulator and the assistant. At the end of each rollout, the assistant receives a trajectory-level reward from LLM judges that score response quality on a Likert-scale rubric (Gunjal et al., 2025; Kim et al., 2024; Liu et al., 2023); rubric details appear in Appendix E. We note that the judges see the user intent z alongside the trajectory, while the assistant never observes z and acts under partial observability.

3 User Simulators

We describe user simulators that can demonstrate the utility of user simulators for training assistants.

3.1 Role-Playing LLM

The dominant type of user simulator that has been used to train LLM assistants is an LLM prompted to role-play as a user. We consider three variants of role-playing LLMs, each hypothesized to approximate human utterance more faithfully than the last. Details are in Appendix D.

RPUSER1: An LLM simulates the user from the first turn onward, guided by an instruction describing the user role and an intent z (Li et al., 2024a; Luo et al., 2024; Kong et al., 2024; Wu et al., 2025; Dinucu-Jianu et al., 2025). The instruction includes details such as "You are role-playing as a human USER" and "As a user, avoid being too detailed in your responses". RPUSER1 is the most widely adopted approach for role-playing user simulators, as it is simple and requires no human data (Yao et al., 2024; Gao et al., 2024; Zhang et al., 2024; Sun et al., 2025; Qian et al., 2025).

RPUSER2: The opening user query is drawn from real first-turn user utterances, but all subsequent user utterances are simulated by the role-playing LLM (Ivey et al., 2024). RPUSER2 has a representative initial state distribution but relies on the role-playing LLM for later turns.

RPUSER3: Building on prior work that uses persona prompts to simulate diverse populations (Ge et al., 2024; Hu & Collier, 2024; Castricato et al., 2025) and taxonomies of conversational behavior (Shim et al., 2025), we inject persona descriptions into the instruction. For each simulated conversation, we randomly sample a description (e.g., "Be Impatient: ...") from a persona pool.

3.2 Learned User Simulator With Supervised Fine-Tuning

Following UserLM (Naous et al., 2025) and user simulators for conversational recommender systems (Meshi et al., 2026), we fine-tune a language model on the user utterances of WildChat-1M. It is the simplest recipe for building user simulators when human–AI interaction data is available (Levin et al., 2000); we leave exploration of other methods (e.g., IRL, GAIL) (Ng et al., 2000; Ho & Ermon, 2016) for future work. For implementation details, please refer to Appendix C.

This model, which we call SFTuser, is the user simulator available during assistant training. However, WildChat-1M covers only conversations with specific assistants (GPT-3.5-Turbo and GPT-4) collected over a limited period (April 2023 to May 2024). To test whether trained assistants generalize to users who interact with different assistants or in different time periods, we prepare two additional simulators that are used only at test time: SFTuser-WC-4.8M, trained on WildChat-4.8M, whose collection period and underlying assistants do not overlap with those of WildChat-1M; and SFTuser-LMSYS, trained on LMSYS-Chat-1M. Evaluating against these simulators measures the assistants' generalizability to unseen user distributions.

4 Experiments

In this section, we describe the experimental setup (Section 4.1) and results of our main experiments:

1. Evaluations on WildBench, to evaluate how assistants trained with different user simulators perform on tasks from real human users (Section 4.2),
2. Examination of how scaling the size of the user simulator — both fine-tuned and role-playing — affects the performance of the trained assistant (Section 4.3),
3. Cross-simulator evaluation, where we evaluate how well each assistant trained with one user simulator generalizes to another user simulator at test time (Section 4.4).

4.1 Experimental Setup

We conduct a controlled experiment in which trained assistants differ *only* in the user simulator they encountered during training. All the other conditions are held constant. This design isolates the effect of the user simulator from confounding factors.

User simulator. All user simulators are derived from the same LLM, Qwen2.5-14B-Instruct (Bai et al., 2023). Specifically, we prepare SFTuser (Section 3.2) by fine-tuning it on WildChat-1M with an 89/5/6% train/val/test split. The role-playing RPuser1–RPuser3 (Section 3.1) also use it.

LLM assistant. We use Qwen2.5-3B-Instruct as the initial assistant for primary experiments. To examine whether the findings depend on assistant size or model family, we also evaluate Qwen2.5-1.5B-Instruct, Qwen2.5-7B-Instruct, and Llama-3.2-3B-Instruct (Grattafiori et al., 2024). Unless otherwise specified, results refer to the Qwen2.5-3B-Instruct; complete results appear in Appendix I.

Judge. During train, we use two open-source LLMs as judges: Qwen3-32B (Yang et al., 2025a) and Mistral-Small-3.1-24B-Instruct (Jiang et al., 2023). Judges are asked to evaluate responses regarding helpfulness and interaction quality (Appendix E), and the reward is an average of the judge scores. Since an assistant optimized against an LLM judge can overfit to its biases, evaluation on WildBench uses a disjoint set: majority voting with gpt-5-mini, gemini-2.5-flash, and claude-haiku-4-5.

Algorithm. We use GRPO (Shao et al., 2024). The trajectory-level reward is provided by the LLM judges described above, and the assistant policy is trained until convergence of the average reward on the validation set. RL details are in Appendix F, and example training curves are in Figure 5 and 6.

4.2 Performance on Real-World Tasks with WildBench

How do LLM assistants trained with each user simulator perform on real-world tasks? In this section, we operationalize 'real-world' via WildBench, a benchmark constructed with a held-out set of human–LLM conversations from WildChat collection (Lin et al., 2024).

WildBench. WildBench-v2 contains 1,024 human–LLM conversations. For a length-n conversation, WildBench provides the first $2n-1$ turns as context and prompts the assistant to generate the n-th response. Each conversation is paired with a checklist of 11.4 items on average (e.g., "Does the AI's response acknowledge the user dissatisfaction with the TV shows except for the Martian Manhunter one?") used to judge response quality. We report two metrics: the checklist satisfaction rate of each

assistant and, for pairwise comparisons, the fraction of conversations in which one assistant satisfies more checklist items than the other assistant. WildBench exhibits a strong correlation ($r^2 = 0.94$) with human-voted Elo ratings on Chatbot Arena (Zheng et al., 2023), making it a useful proxy for assistant performance in real-world settings.

Table 1: Tie-accounted pairwise win rate matrix and checklist satisfaction rate on WildBench-v2. Entry (i, j) gives the win rate of the assistant trained with j against the one trained with i. For win rates, values following $\pm$ denote the half-width of the 95% Wald confidence interval; for satisfaction rates, they denote the standard error of the mean.

Win [%]	Initial	RPUSER1	RPUSER2	RPUSER3	SFTUSER
Initial	—	55.4 ± 2.7	59.9 ± 2.7	61.3 ± 2.7	**68.4** ± 2.6
RPUSER1	44.6 ± 2.7	—	55.2 ± 2.8	56.6 ± 2.8	**63.9** ± 2.7
RPUSER2	40.1 ± 2.7	44.8 ± 2.8	—	51.0 ± 2.8	**60.0** ± 2.7
RPUSER3	38.7 ± 2.7	43.4 ± 2.8	49.0 ± 2.8	—	**58.5** ± 2.8
SFTUSER	31.6 ± 2.6	36.1 ± 2.7	40.0 ± 2.7	41.5 ± 2.8	—
Satisfy [%]	51.1±1.0	53.6±1.0	56.6±1.0	57.7±1.0	**61.5**±0.9

Main Result. Table 1 reports pairwise win rates and checklist satisfaction rates. We highlight three patterns. First, the gains from training with sophisticated user simulators are clear. The assistant trained with the simplest role-playing LLM (RPUSER1) loses to the assistant trained with SFTUSER 63.9% of the time. Second, user simulators that incorporate methods to increase realism, from RPUSER1 to RPUSER2 to RPUSER3, lead to stronger trained assistants, with higher win rates and checklist satisfaction rates. However, the assistant trained with SFTUSER still outperforms every assistant trained with role-playing LLMs.

Additional Results. The advantage of training with SFTUSER persists across assistant sizes and model families. Against the RPUSER3-trained assistant, the SFTUSER-trained assistant achieves a win rate of 54.8 ± 2.8% (Qwen2.5-7B-Instruct), 54.6 ± 2.8% (Llama-3.2-3B-Instruct), and 56.8 ± 2.7% (Qwen2.5-1.5B-Instruct), compared with 58.5 ± 2.8% in Table 1. These results extend the finding beyond a single assistant family and size. We additionally evaluate the trained assistants on the 250 prompts in Arena-Hard-Auto-v2 (Li et al., 2024b), which originate from Chatbot Arena. The SFTUSER-trained assistant achieves 66.9% win rate against the initial assistant and 56.0% against the RPUSER3-trained assistant. This comparison provides complementary evidence that the advantage extends beyond WildBench's WildChat-derived tasks. Details are available in Appendix I.

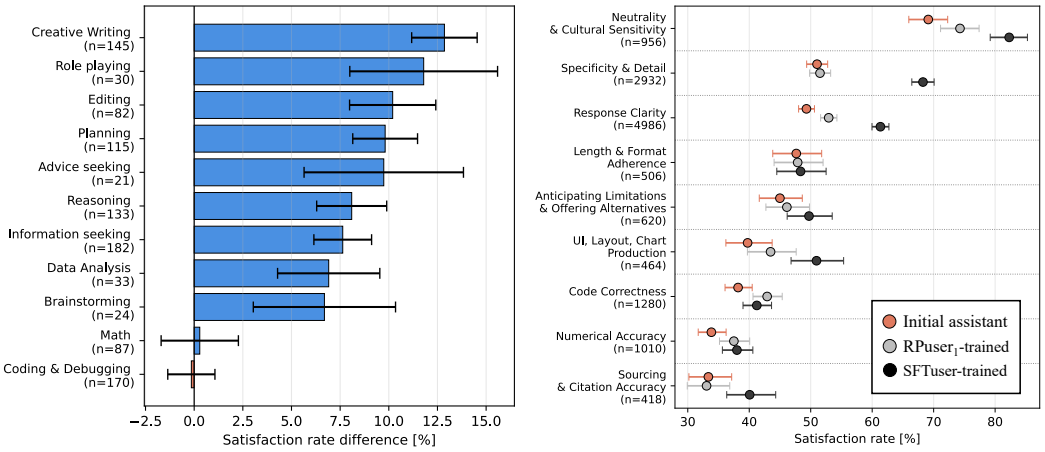


Figure 2: **(Left)** Difference in checklist satisfaction rate between the SFTUSER-trained and the RPUSER1-trained assistants, stratified over conversation categories. Positive numbers indicate SFTUSER-trained assistant satisfies more. **(Right)** Satisfaction rates among initial, RPUSER1-trained, and SFTUSER-trained assistants, stratified over nine representative checklist dimensions.

Per-category gap. Using WildBench's conversation category taxonomy, we compare checklist satisfaction rates between the SFTUSER- and RPUSER1-trained assistants (Figure 2, left). The SFTUSER-trained assistant achieves higher rates in most categories, with particularly large gaps in Creative Writing, Role Playing, and Editing (+12.9, +11.8, and +10.2%, respectively). In Math and Coding & Debugging, however, the differences are not statistically significant. We hypothesize that this pattern reflects differences in what determines task success. Math and coding tasks have verifiable answers, so performance depends heavily on the assistant's problem-solving capability. Our training reward (Section 4.1) emphasizes helpfulness and interaction quality rather than verifying mathematical or coding correctness. Improvements under this reward therefore need not translate into improved problem-solving accuracy. For open-ended tasks such as writing and editing, success depends more on adapting to the user's goals and preferences throughout the conversation, which aligns more closely with our training objective.

Per-dimension gap. To characterize which aspects of response quality improve with training, we cluster WildBench's 12K checklist items using their text embeddings, then use gpt-5 to classify items along representative evaluation dimensions (e.g., Numerical Accuracy, Neutrality & Cultural Sensitivity; full list in Appendix G). This grouping is not exhaustive, as some items are specific to individual conversations. Figure 2 (right) reports satisfaction rates across nine dimensions for the initial, RPUSER1-trained, and SFTUSER-trained assistants. Relative to the initial assistant, the SFTUSER-trained assistant shows its largest gains in Neutrality & Cultural Sensitivity, Specificity & Detail, and Response Clarity, with no statistically significant changes in Numerical Accuracy or Code Correctness. The concentration of gains in interaction quality is consistent with our reward design, which emphasizes helpfulness and clarity rather than verifying math or code correctness.

4.3 Effects of Scaling User Simulators on Assistant Performance

In the previous section, all user simulators used Qwen2.5-14B-Instruct. With this shared backbone, RPUSER1–RPUSER3 yielded *lower downstream utility* than SFTUSER. We next ask whether scaling the simulator's underlying LLM can close this gap. To test this, we vary the simulator backbone while holding the assistant initialization fixed and evaluate the resulting assistants.

Table 2: WildBench evaluation with scaled user simulators. **(Left)** Three assistants trained with RPUSER3 by Qwen2.5-14B-Instruct (RPUSER3), Qwen3-14B (RP3-14B), and Qwen3-32B (RP3-32B). **(Right)** Three assistants trained with learned simulators of scales Qwen2.5-{7,14,32}B-Instruct.

Win [%]	Initial	RPUSER3	RP3-14B	RP3-32B	SFTUSER
Initial	—	61.3 ± 2.7	60.0 ± 2.7	63.3 ± 2.7	**68.4** ± 2.6
RPUSER3	38.7 ± 2.7	—	48.2 ± 2.8	52.3 ± 2.8	**58.5** ± 2.8
RP3-14B	40.0 ± 2.7	51.8 ± 2.8	—	53.3 ± 2.7	**60.2** ± 2.7
RP3-32B	36.7 ± 2.7	47.7 ± 2.8	46.7 ± 2.7	—	**57.4** ± 2.7
Satisfy [%]	51.1±1.0	57.7±1.0	57.5±1.0	59.2±1.0	**61.5**±0.9

Win [%]	Initial	SFT-7B	SFTUSER	SFT-32B
Initial	—	67.9 ± 2.6	68.4 ± 2.6	**75.0** ± 2.4
SFT-7B	32.1 ± 2.6	—	51.9 ± 2.8	**61.5** ± 2.7
SFTUSER	31.6 ± 2.6	48.1 ± 2.8	—	**56.8** ± 2.8
SFT-32B	25.0 ± 2.4	38.5 ± 2.7	43.2 ± 2.8	—
Satisfy [%]	51.1±1.0	60.7±1.0	61.5±0.9	**64.8**±0.8

Scaling Role-Playing LLMs Does Not Close the Utility Gap. We use RPUSER3, the strongest role-playing baseline, with Qwen3-{14,32}B which substantially outperform Qwen2.5-14B-Instruct on general LLM capability (Yang et al., 2025a). We denote these simulators by RP3-14B and RP3-32B. Results are shown in Table 2 left: neither simulators closes the downstream assistant performance gap to SFTUSER. We hypothesize that stronger LLMs produce more fluent utterances, but that fluency does not make simulated utterances more representative of human users (Zhou et al., 2026).

Scaling Learned Simulators Improves Downstream Utility. We fine-tune Qwen2.5-{7,14,32}B-Instruct on WildChat-1M, yielding SFT-7B, the original SFTUSER (14B), and SFT-32B, and train an assistant with each simulator. SFT-32B yields statistically significant gains over both smaller simulators (Table 2, right). We hypothesize that learned simulators benefit from scale because fine-tuning a larger model better captures the distribution of real human utterances, thereby providing more representative interactions during assistant training. Lower validation perplexity of larger learned simulators supports this interpretation (Table 5).

Together, these results show that stronger role-playing LLMs do not close the utility gap to fine-tuned user simulators, while scaling the learned simulator further improves assistant performance. This suggests that the downstream benefits of simulator capacity depend on how the model is trained to simulate human behavior, rather than on general LLM capability alone.

4.4 CROSS-SIMULATOR EVALUATION

We now ask a complementary question: does the assistant remain effective when paired with user simulators it was not trained against? This matters because an assistant optimized against a single user simulator may learn to exploit that simulator's idiosyncrasies to elicit high reward, without genuinely improving response quality. Cross-simulator evaluation lets us distinguish these two cases. For each pair of simulators (A, B), we take the assistant trained with A and pair it with B at test time, conditioning on the same set of intents z from the held-out WildChat-1M test split.

Table 3: Cross-simulator evaluation of trained assistants. Rows correspond to user simulators used at test time; columns are assistants trained with different user simulators. We normalize judge scores (Section 4.1) from 0–10 to 0–100 for convenience. In each row, the best and the runner-up are bold-faced and underlined, respectively; ± indicates 95% CI from bootstrapping scores.

Eval with ↓ Train with →	Initial	RPUSER1	RPUSER2	RPUSER3	SFTUSER
RPUSER1	68.5 ±0.4	<u>88.5</u> ±0.2	87.6 ±0.2	87.6 ±0.2	**88.7** ±0.2
RPUSER2	63.3 ±0.4	82.9 ±0.2	<u>86.8</u> ±0.2	86.3 ±0.2	**87.5** ±0.2
RPUSER3	63.1 ±0.4	79.8 ±0.3	82.0 ±0.3	<u>83.8</u> ±0.2	**84.8** ±0.2
SFTUSER	60.9 ±0.4	72.9 ±0.4	78.2 ±0.3	<u>80.3</u> ±0.3	**82.5** ±0.3
SFTUSER-WC-4.8M	61.1 ±0.4	73.0 ±0.3	74.7 ±0.3	<u>76.8</u> ±0.3	**82.8** ±0.2
SFTUSER-LMSYS	64.4 ±0.4	67.4 ±0.4	67.0 ±0.4	<u>68.3</u> ±0.4	**70.2** ±0.4

Main Result. Table 3 reports average test reward. *Rows are not directly comparable*: each row corresponds to a different test-time simulator, and simulators differ in behavior: cooperative RPUSER1–RPUSER3 elicit higher judge scores than SFTUSER, which exhibits behaviors such as information withholding and ambiguity (Naous et al., 2025). Within a row, only the assistant varies.

The SFTUSER-trained assistant achieves the highest reward in every row, including rows for simulators it did not see during training. Assistants trained with role-playing LLMs, by contrast, are competitive only with their train-time partner. Their performance gap to the SFTUSER-trained assistant is small (≤ 1.0) when tested against role-playing LLMs but widens to 2.2 against SFTUSER. Pairing each assistant with two held-out user simulators (Section 3.2) yields the same conclusion. When evaluated with SFTUSER-WC-4.8M and SFTUSER-LMSYS, the SFTUSER-trained assistant remains the strongest (82.8 vs. 76.8 for the runner-up, and 70.2 vs. 68.3, respectively).

These results support two related hypotheses. First, the learned user simulator exposes the assistant to a broader distribution of user behaviors, producing an assistant that generalizes across simulators (Peng et al., 2018). Second, the strong train-time performance of role-play-trained assistants reflects optimization against a narrow band of simulator-specific behaviors rather than improved interaction / response quality (Carroll et al., 2019).

5 REAL-WORLD USER STUDY

We conduct a user study with 283 Prolific participants to evaluate assistant quality in live multi-turn interactions. To make interactions across assistant conditions more comparable and to retain sufficient observations per comparison, we focus the human study on a shared task domain. Writing, as used by Wu et al. (2025), is accessible to a general participant population and allows participants to express their own preferences through iterative revision. Each participant is randomly assigned one of three task types—blog post, creative writing, or personal statement—and selects a topic from a curated list. Participants then answer pre-writing questions, such as identifying the intended audience, to establish a concrete goal before interacting with the assistants. This design makes our controlled study a better proxy for real-world interactions where users have their genuine intent, while allowing us to discourage the disclosure of identifying information.

After a practice session to familiarize themselves with the study interface, participants engage in at least five turns of conversation with two anonymized assistants, sampled from {Initial, SFTUSER-trained, RPUSER1-trained}, whose responses are displayed side by side in randomized positions. At each turn, the participant selects the preferred response and provides a brief explanation guided by

the rubric; the conversation continues from the chosen response. Full details of the study design, including recruitment, a set of topics, and quality-control measures, are in Appendix H.

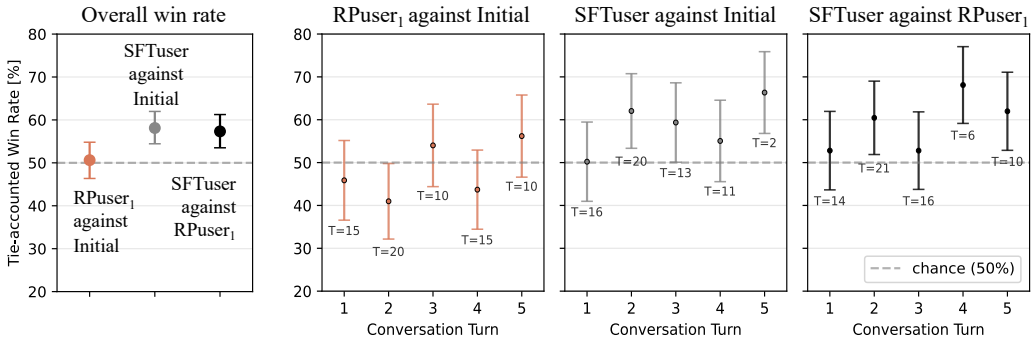


Figure 3: Tie-accounted pairwise win rates from the real-world user study ($n = 283$, ≥ 5 turns each). **(Left)** Win rates for the three assistant pairs, aggregated across turns. **(Right)** Win rates stratified by turn. Error bars denote 95% conversation-level bootstrapped confidence intervals (Davison & Hinkley, 1997); 'T=·' below each bar indicates the number of ties (out of ≥ 90 participants per pair).

Quantitative results. Figure 3 reports pairwise win rates for the three assistant pairs, aggregated across turns (left) and stratified by turn (right). Each pair is evaluated by at least 90 participants over at least 5 turns, yielding at least 450 pairwise decisions per pair and at least 90 per turn.

In aggregate, the SFTUSER-trained assistant outperforms both the initial assistant (58.1%, CI [54.3, 61.8]) and the RPUSER1-trained assistant (57.3%, CI [53.4, 61.1]). In contrast, the RPUSER1-trained assistant is statistically indistinguishable from the initial assistant (50.6%, CI [46.5, 54.9]). While the exact numbers differ from WildBench, the conclusion remains consistent: on WildBench, training with RPUSER1 also only yielded marginal gains over the initial assistant (55.4%, CI [52.7, 58.1]; Section 4.2), while the SFTUSER-trained assistant significantly outperformed both the initial and RPUSER1-trained assistants. When stratified by turn, confidence intervals widen due to smaller samples, but the aggregate pattern persists. The SFTUSER-trained assistant's estimated win rate stays above 50% at every turn against both comparators, while the RPUSER1-trained assistant oscillates around 50% against the initial assistant.

Table 4: Example explanations from human participants for their pairwise decisions.

Assistant Pair	Choice	Explanation of Choice
SFTUSER-trained vs. Initial	SFTUSER-trained	"It was more comprehensiveness and organized, it divided the statement into obvious sections with bolded headlines which help the reader to follow up . . . "
	Initial Assistant	"I think it better due to the simplicity and relatable nature of many of the steps, with the addition of providing many out of the box options . . . "
SFTUSER-trained vs. RPUSER1-trained	SFTUSER-trained	"It feels more personal and natural, with better reflection on challenges, growth, and motivation. Other sounds more generic and repetitive."
	RPUSER1-trained	"Other includes unnecessary titles 'First Paragraph, Second Paragraph'. It is clear and simple although it will need one more round, titles could be better."

Qualitative results. We use these participants' choice explanations to characterize each assistant's perceived strengths and weaknesses; representative examples appear in Table 4. The SFTUSER-trained assistant is frequently described as personal, grounded, and context-aware ("comprehensive and organized," "feels natural," "more engaged in the conversation"), but loses on prompts demanding brevity or strict format compliance, where some raters find it too long or too literary. The RPUSER1-trained assistant, conversely, is preferred for thoroughness and surface polish—complete drafts and clean structure—yet is often flagged as stiff and bullet-heavy. The initial assistant is concise and direct, which raters value on short-form prompts, though some note that it can lose track of earlier-turn details as conversations deepen. Taken together, these explanations complement the quantitative findings: training with a learned user simulator yields an assistant whose outputs are perceived as more engaging and context-aware in multi-turn conversations.

6 Related Work

User Simulation with LLMs. A body of work investigates using language models to simulate human users. One approach prompts an instruction-tuned LLM to role-play as a user in a zero-shot or few-shot fashion. In task-oriented dialogue, Terragni et al. (2023) and Davidson et al. (2023) build user simulators via in-context learning, while Luo et al. (2024) employs two LLMs (generator and verifier) to improve goal adherence. Beyond dialogue systems, Zhang et al. (2025) uses prompted LLMs to simulate user engagement for recommender systems, and Tamoyan et al. (2025) constructs persona-based prompted simulators for demographically diverse chatbot interactions. Recently, Shim et al. (2025) develops user simulators with a scaffold to test agents under adversarial user behaviors.

However, a growing line of work reveals that instruction-tuned LLMs are poor proxies for human users: they produce predictable, cooperative, and well-structured utterances not stylistically aligned with humans (Ivey et al., 2024; Yoon et al., 2024; Zhou et al., 2026). These findings motivate approaches that instead train language models on human data. In a conversational setting, Naous et al. (2025) introduces UserLM, a model fine-tuned on the user side of WildChat conversations (Zhao et al., 2024), producing more realistic user utterances than prompted alternatives. Wang et al. (2025a) and Wu et al. (2026) extend this by training LLMs to generate internal chain-of-thought reasoning; Gandhi et al. (2026) pursues a related objective of simulating human-human dialogues; Çalık & Akkuş (2025) applies DPO (Rafailov et al., 2023) to synthetic pairwise data that up-votes human-like generations. While prior works focus on building and evaluating user simulators or evaluating LLM assistants against them, ours is the first to systematically study how the choice of LLM-based user simulator affects the trained assistant's capability to collaborate with humans.

Training LLM Assistants with User Simulators. Recent works train LLM assistants via interaction with simulated users. On the algorithmic side, several works pursue multi-turn preference optimization (Shani et al., 2024; Shi et al., 2024); for instance, REFUEL (Gao et al., 2024) is an on-policy method that regresses the log-policy-ratio towards the relative Q-value obtained via MC estimation. In the human–LLM interaction domain, prior works target the collaboration capabilities of LLMs. This includes clarifying-question generation by simulating user interactions (Andukuri et al., 2024; Zhang et al., 2024), multi-turn collaboration (Wu et al., 2025), proactive information elicitation (Kong et al., 2024), and pedagogical capability via interaction with an LLM role-playing student (Dinucu-Jianu et al., 2025). In spite of the growing body of work on building more realistic user simulators, a consistent thread in training LLM assistants via simulation is the use of role-playing LLMs.

7 Conclusion, Limitations, And Future Work

In this work, we reframed simulator quality in terms of its downstream utility: the performance of assistants trained through simulated interaction. Across our automated evaluations, assistants trained with the learned simulator outperform those trained with role-playing simulators, with the advantage extending to multiple assistant initializations and evaluations outside the WildChat data family. In our user study, the SFTUSER-trained assistant is preferred to both the initial assistant and the RPUSER1-trained assistant, while we do not find a statistically significant preference between the latter two. Persona conditioning and role-playing with stronger LLMs improve some automated outcomes but do not close the gap to the learned simulator in the tested settings. These findings support grounding simulators in human interaction data and assessing simulator design choices by their downstream consequences.

Limitations and future work. Our work also surfaces limitations that could be explored in future work. First, while training against user simulators can be effective, the simulator must remain realistic as the assistant policy shifts during training. Second, we focus on open-domain conversation, given its generality and the availability of data, but future work could explore specialized environments such as τ-bench (Yao et al., 2024), where dialogues are goal-oriented and conditioned on long tool-augmented contexts. We discuss concrete mitigations for these issues in Appendix B, including bounded-horizon rollouts branched from real conversation prefixes (Feinberg et al., 2018; Sutton, 1991), uncertainty-aware or iteratively retrained user simulators (Ross et al., 2011; Chua et al., 2018), and collecting human interaction data across a broader range of domains (Zhou et al., 2026). Our work also establishes the importance of evaluating user simulators downstream—based on the performance of their trained assistants with real humans—but this metric is expensive to measure, requiring a user study in the best case or at least a real-world task benchmark like WildBench. Future work could explore upstream proxies of this metric, enabling more efficient iteration on simulator design.

AI USE STATEMENT

In this work, we used generative AI tools for: (1) aid and polish writing, (2) retrieval and discovery of related works, (3) research execution, including implementation of the web-hosted user study service, and (4) generating synthetic conversation rollouts between assistant models and user simulators. We have not used generative AI tools for: (1) drafting sections of the paper, and (2) for proving mathematical claims. We have reviewed all AI-assisted work: human authors finalized all writing edits; authors made sure retrieval of related works are accurate, relevant, and not hallucinated; and code implementation was reviewed and tested. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work involves human subjects and data derived from real human–AI conversations. We took the following precautions.

Human subjects. Our user study was reviewed and approved by the Institutional Review Board (IRB) at our institution, and all 283 participants gave informed consent before participation. Participants were recruited through Prolific and compensated $10.00 for an estimated 40-minute session, an hourly rate of $15.00. Participation was voluntary and could be discontinued at any point without penalty. We did not collect any personally identifiable information (PII), including names, email addresses, or IP addresses. Prolific identifiers were used only to verify completion and issue payment before being deleted. Participants were instructed not to enter sensitive personal information and were given a mechanism to report objectionable model output. The consent form, privacy protections, PII handling, and the content safety filtering pipeline are documented in Appendix H.

Conversational data. Our learned user simulators are fine-tuned on WildChat, a corpus of human–LLM conversations released for research use. It can contain personal, offensive, or sensitive text, and models fine-tuned on them can reproduce it. We use this data only to model user-side conversational behavior, and we describe our filtering and preprocessing steps in Appendix C.

Broader risks. Optimizing assistants against a user simulator carries the risk that assistants learn to satisfy the simulator rather than the human users, including sycophancy or manipulation when the training signal is derived from simulated user feedback. Our central claim, that simulator quality should be measured by downstream effects on real users, is intended to make such failure modes visible. We discuss the limits of this approach in Section 7 and Appendix B.

REPRODUCIBILITY STATEMENT

We release our code at Anonymous GitHub. We will release trained LLM assistant checkpoints, learned user simulator checkpoints, simulated conversation trajectories, and de-identified user study data upon acceptance. The paper documents the full experimental pipeline in the appendix. Supervised fine-tuning of the user simulators, including data construction, filtering, and checkpoint selection, is described in Appendix C. The prompts defining every role-playing simulator are reproduced in Appendix D. Reinforcement learning details, including the train and test split, hyperparameters, and compute, appear in Appendix F with the rubric and LLM judge specified in Appendix E. Our evaluation protocol on WildBench, the confidence interval calculation that accounts for ties, and the checklist-based judging procedure are given in Appendix G. The user study protocol and interface are in Appendix H.

REFERENCES

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Chinmaya Andukuri, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah D Goodman. Star-gate: Teaching language models to ask clarifying questions. *arXiv preprint arXiv:2403.19154*, 2024.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Ethem Yağız Çalık and Talha Rüzgar Akkuş. Enhancing human-like responses in large language models. *arXiv preprint arXiv:2501.05032*, 2025.

Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pp. 160–172. Springer, 2013.

Shiyi Cao, Sumanth Hegde, Dacheng Li, Tyler Griggs, Shu Liu, Eric Tang, Jiayi Pan, Xingyao Wang, Akshay Malik, Graham Neubig, Kourosh Hakhamaneshi, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Skyrl-v0: Train real-world long-horizon agents via reinforcement learning, 2025.

Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32, 2019.

Louis Castricato, Nathan Lile, Rafael Rafailov, Jan-Philipp Fränken, and Chelsea Finn. Persona: A reproducible testbed for pluralistic alignment. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 11348–11368, 2025.

Serina Chang, Ashton Anderson, and Jake M Hofman. Chatbench: From static benchmarks to human-ai evaluation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26009–26038, 2025.

Xiaoyu Chen, Jiachen Hu, Chi Jin, Lihong Li, and Liwei Wang. Understanding domain randomization for sim-to-real transfer. In *International Conference on Learning Representations (ICLR)*, 2022.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E Gonzalez, et al. Chatbot arena: An open platform for evaluating llms by human preference. *arXiv preprint arXiv:2403.04132*, 2024.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 1282–1289, 2019.

Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization. *arXiv preprint arXiv:2310.02743*, 2023.

Sam Davidson, Salvatore Romeo, Raphael Shu, James Gung, Arshit Gupta, Saab Mansour, and Yi Zhang. User simulation with large language models for evaluating task-oriented dialogue. *arXiv preprint arXiv:2309.13233*, 2023.

Anthony Christopher Davison and David Victor Hinkley. *Bootstrap methods and their application*. Number 1. Cambridge university press, 1997.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, 2011.

Yuntian Deng, Wenting Zhao, Jack Hessel, Xiang Ren, Claire Cardie, and Yejin Choi. Wildvis: Open source visualizer for million-scale chat logs in the wild. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 497–506, 2024.

David Dinucu-Jianu, Jakub Macina, Nico Daheim, Ido Hakimi, Iryna Gurevych, and Mrinmaya Sachan. From problem-solving to teaching problem-solving: Aligning llms with pedagogy using reinforcement learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 272–292, 2025.

Yixin Dong, Charlie F Ruan, Yaxing Cai, Ziyi Xu, Yilong Zhao, Ruihang Lai, and Tianqi Chen. Xgrammar: Flexible and efficient structured generation engine for large language models. *Proceedings of Machine Learning and Systems*, 7, 2025.

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 Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 35212–35290, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1786. URL https://aclanthology.org/2025.emnlp-main.1786/.

Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I Jordan, Joseph E Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

Kanishk Gandhi, Agam Bhatia, and Noah D Goodman. Learning to simulate human dialogue. *arXiv preprint arXiv:2601.04436*, 2026.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pp. 10835–10866. PMLR, 2023.

Zhaolin Gao, Wenhao Zhan, Jonathan D Chang, Gokul Swamy, Kianté Brantley, Jason D Lee, and Wen Sun. Regressing the relative future: Efficient policy optimization for multi-turn rlhf. *arXiv preprint arXiv:2410.04612*, 2024.

Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. Scaling synthetic data creation with 1,000,000,000 personas. *arXiv preprint arXiv:2406.20094*, 2024.

Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P Adams, and Sergey Levine. Why generalization in RL is difficult: Epistemic POMDPs and implicit partial observability. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. A survey on llm-as-a-judge. *The Innovation*, 2024.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *arXiv preprint arXiv:2507.17746*, 2025.

Christine Herlihy, Jennifer Neville, Tobias Schnabel, and Adith Swaminathan. On overcoming miscalibrated conversational priors in llm-based chatbots. *arXiv preprint arXiv:2406.01633*, 2024.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Tiancheng Hu and Nigel Collier. Quantifying the persona effect in llm simulations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10289–10307, 2024.

Zenan Huang, Yihong Zhuang, Guoshan Lu, Zeyu Qin, Haokai Xu, Tianyu Zhao, Ru Peng, Jiaqi Hu, Zhanming Shen, Xiaomeng Hu, et al. Reinforcement learning with rubric anchors. *arXiv preprint arXiv:2508.12790*, 2025.

Jonathan Ivey, Shivani Kumar, Jiayu Liu, Hua Shen, Sushrita Rakshit amd Rohan Raju, Haotian Zhang, Aparna Ananthasubramaniam, Junghwan Kim, Bowen Yi, Dustin Wright, Abraham Israeli, Anders Giovanni Møller, Lechen Zhang, and David Jurgens. Real or robotic? assessing whether llms accurately simulate qualities of human responses in dialogue. *arXiv preprint arXiv:2409.08330*, 2024.

Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Simon Keizer, Caroline Dockes, Norbert Braunschweiler, Svetlana Stoyanchev, and Rama Doddipatla. Adversarial learning of neural user simulators for dialogue policy optimisation. *arXiv preprint arXiv:2306.00858*, 2023.

Hyunwoo Kim, Jihyeon Ryu, Jinho Lee, Hyungon Ryu, Kiran Praveen, Shyamala Prayaga, Kirit Thadaka, Will Jennings, Bardiya Sadeghi, Ashton Sharabiani, Yejin Choi, and Yev Meyer. Nemotron-personas-korea: Synthetic personas aligned to real-world distributions for korea, April 2026. URL https://huggingface.co/datasets/nvidia/Nemotron-Personas-Korea.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4334–4353, 2024.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Hannah Rose Kirk, Alexander Whitefield, Paul Rottger, Andrew M Bean, Katerina Margatina, Rafael Mosquera-Gomez, Juan Ciro, Max Bartolo, Adina Williams, He He, et al. The prism alignment dataset: What participatory, representative and individualised human feedback reveals about the subjective and multicultural alignment of large language models. *Advances in Neural Information Processing Systems*, 37:105236–105344, 2024.

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 (Volume 1: Long Papers)*, pp. 7841–7863, 2024.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. Llms get lost in multi-turn conversation. *arXiv preprint arXiv:2505.06120*, 2025.

Esther Levin, Roberto Pieraccini, Wieland Eckert, et al. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on speech and audio processing*, 8(1): 11–23, 2000.

Ruosen Li, Ruochen Li, Barry Wang, and Xinya Du. Iqa-eval: Automatic evaluation of human-model interactive question answering. *Advances in Neural Information Processing Systems*, 37: 109894–109921, 2024a.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*, 2024b.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. *arXiv preprint arXiv:2406.04770*, 2024.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 2511–2522, 2023.

Xiang Luo, Zhiwen Tang, Jin Wang, and Xuejie Zhang. Duetsim: Building user simulator with dual large language models for task-oriented dialogues. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 5414–5424, 2024.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. In *Conference on Robot Learning*, pp. 1162–1176. PMLR, 2020.

Ofer Meshi, Krisztian Balog, Sally Goldman, Avi Caciularu, Guy Tennenholtz, Jihwan Jeong, Amir Globerson, and Craig Boutilier. Convapparel: A benchmark dataset and validation framework for user simulators in conversational recommenders. *arXiv preprint arXiv:2602.16938*, 2026.

Meta. Llama cookbook: Official guide to building with llama models, 2025. URL https://github.com/meta-llama/llama-cookbook. Accessed: March 29th, 2026.

Suhong Moon, Minwoo Kang, Joseph Suh, Mustafa Safdari, and John Canny. Identity, cooperation and framing effects within groups of real and simulated humans. *arXiv preprint arXiv:2601.16355*, 2026.

Tarek Naous, Philippe Laban, Wei Xu, and Jennifer Neville. Flipping the dialogue: Training and evaluating user language models. *arXiv preprint arXiv:2510.06552*, 2025.

Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

Bo Ni, Yu Wang, Leyao Wang, Branislav Kveton, Franck Dernoncourt, Yu Xia, Hongjie Chen, Reuben Luera, Samyadeep Basu, Subhojyoti Mukherjee, et al. A survey on llm-based conversational user simulation. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4266–4301, 2026.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3803–3810, 2018.

Steven T Piantadosi, Harry Tily, and Edward Gibson. The communicative function of ambiguity in language. *Cognition*, 122(3):280–291, 2012.

Olivier Pietquin and Thierry Dutoit. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(2):589–599, 2006.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International conference on machine learning*, pp. 2817–2826. PMLR, 2017.

Cheng Qian, Zuxin Liu, Akshara Prabhakar, Jielin Qiu, Zhiwei Liu, Haolin Chen, Shirley Kokane, Heng Ji, Weiran Yao, Shelby Heinecke, et al. Userrl: Training interactive user-centric agent via reinforcement learning. *arXiv preprint arXiv:2509.19736*, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Rafael Rafailov, Yaswanth Chittepu, Ryan Park, Harshit Sushil Sikchi, Joey Hejna, Brad Knox, Chelsea Finn, and Scott Niekum. Scaling laws for reward model overoptimization in direct alignment algorithms. *Advances in Neural Information Processing Systems*, 37:126207–126242, 2024.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Lior Shani, Aviv Rosenberg, Asaf Cassel, Oran Lang, Daniele Calandriello, Avital Zipori, Hila Noga, Orgad Keller, Bilal Piot, Idan Szpektor, et al. Multi-turn reinforcement learning with preference human feedback. *Advances in Neural Information Processing Systems*, 37:118953–118993, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Wentao Shi, Mengqi Yuan, Junkang Wu, Qifan Wang, and Fuli Feng. Direct multi-turn preference optimization for language agents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 2312–2324, 2024.

Jeonghoon Shim, Woojung Song, Cheyon Jin, Seungwon KooK, and Yohan Jo. Non-collaborative user simulators for tool agents. *arXiv preprint arXiv:2509.23124*, 2025.

Weiwei Sun, Xuhui Zhou, Weihua Du, Xingyao Wang, Sean Welleck, Graham Neubig, Maarten Sap, and Yiming Yang. Training proactive and personalized llm agents. *arXiv preprint arXiv:2511.02208*, 2025.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Ryuichi Takanobu, Hanlin Zhu, and Minlie Huang. Guided dialog policy learning: Reward estimation for multi-domain task-oriented dialog. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 100–110, 2019.

Hovhannes Tamoyan, Hendrik Schuff, and Iryna Gurevych. Llm roleplay: Simulating human-chatbot interaction. In *Proceedings of the Third Workshop on Social Influence in Conversations (SICon 2025)*, pp. 1–26, 2025.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Silvia Terragni, Modestas Filipavicius, Nghia Khau, Bruna Guedes, André Manso, and Roland Mathis. In-context learning user simulators for task-oriented dialog systems. *arXiv preprint arXiv:2306.00774*, 2023.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017.

Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. Replacing judges with juries: Evaluating llm generations with a panel of diverse models. *arXiv preprint arXiv:2404.18796*, 2024.

Vijay Viswanathan, Yanchao Sun, Shuang Ma, Xiang Kong, Meng Cao, Graham Neubig, and Tongshuang Wu. Checklists are better than reward models for aligning language models. *arXiv preprint arXiv:2507.18624*, 2025.

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, 2025a.

Zhefan Wang, Ning Geng, Zhiqiang Guo, Weizhi Ma, and Min Zhang. Human vs. agent in task-oriented conversations. In *Proceedings of the 2025 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pp. 133–142, 2025b.

Marcus Williams, Micah Carroll, Adhyyan Narang, Constantin Weisser, Brendan Murphy, and Anca Dragan. On targeted manipulation and deception when optimizing llms for user feedback. *arXiv preprint arXiv:2411.02306*, 2024.

Shirley Wu, Michel Galley, Baolin Peng, Hao Cheng, Gavin Li, Yao Dou, Weixin Cai, James Zou, Jure Leskovec, and Jianfeng Gao. Collabllm: From passive responders to active collaborators. *arXiv preprint arXiv:2502.00640*, 2025.

Shirley Wu, Evelyn Choi, Arpandeep Khatua, Zhanghan Wang, Joy He-Yueya, Tharindu Cyril Weerasooriya, Wei Wei, Diyi Yang, Jure Leskovec, and James Zou. Humanlm: Simulating users with state alignment beats response imitation. *arXiv preprint arXiv:2603.03303*, 2026.

Yuhong Xu, Peijie Huang, Jiecong Tang, Qiangjia Huang, Zhenpeng Deng, Weimou Peng, and Jiajie Lu. Policy optimization of dialogue management in spoken dialogue system for out-of-domain utterances. In *2016 International Conference on Asian Language Processing (IALP)*, pp. 10–13. IEEE, 2016.

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

Chenyang Yang, Yike Shi, Qianou Ma, Michael Xieyang Liu, Christian Kästner, and Tongshuang Wu. What prompts don't say: Understanding and managing underspecification in llm prompts. *arXiv preprint arXiv:2505.13360*, 2025b.

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

Se-eun Yoon, Zhankui He, Jessica Echterhoff, and Julian McAuley. Evaluating large language models as generative user simulators for conversational recommendation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1490–1504, 2024.

Michael JQ Zhang, W Bradley Knox, and Eunsol Choi. Modeling future conversation turns to teach llms to ask clarifying questions. *arXiv preprint arXiv:2410.13788*, 2024.

Zijian Zhang, Shuchang Liu, Ziru Liu, Rui Zhong, Qingpeng Cai, Xiangyu Zhao, Chunxu Zhang, Qidong Liu, and Peng Jiang. Llm-powered user simulator for recommender system. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 13339–13347, 2025.

Rui Zhao, Jinming Song, Yufeng Yuan, Haifeng Hu, Yang Gao, Yi Wu, Zhongqian Sun, and Wei Yang. Maximum entropy population-based training for zero-shot human-ai coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 6145–6153, 2023.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatgpt interaction logs in the wild. *arXiv preprint arXiv:2405.01470*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric Xing, et al. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. In *International Conference on Learning Representations*, volume 2024, pp. 22225–22257, 2024.

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*, 2026.

A Extended Related Work

Sim-to-Real Transfer in RL. A challenge in RL is that an agent trained in a simulated environment overfits to its specific dynamics, developing a brittle policy that fails under distributional shift (Cobbe et al., 2019). Domain randomization (Tobin et al., 2017; Mehta et al., 2020) addresses this by training an agent across simulation environment parameters (e.g., friction coefficient and visual appearance) so that the policy learns to be robust to variation (Peng et al., 2018; Andrychowicz et al., 2020). In a similar spirit, a related strategy in multi-agent settings is to diversify partners an agent trains with. Population-based training (PBT) (Jaderberg et al., 2019; Zhao et al., 2023) maintains a population of co-evolving agents, exposing the agent to diverse partner behaviors. However, PBT agents may co-adapt and converge on opaque coordination conventions that fail when paired with humans (Carroll et al., 2019).

Carroll et al. (2019) demonstrates that training an agent against a behavior-cloned human model in a cooperative game (Overcooked) produces a policy that coordinates with actual humans. Our work builds on this insight. By preparing user models via supervised fine-tuning on human user utterances (Naous et al., 2025; Meshi et al., 2026), we construct a multi-turn conversation environment with a learned user simulator. Each rollout against this simulator effectively samples from a space of human utterances, varying in belief, desire, intent, and linguistic patterns (Kirk et al., 2024; Herlihy et al., 2024; Yang et al., 2025b). This training environment produces agents that are robust to the wide range of actual users, in line with the broader effort to train under greater environmental variability for transferable policies (Pinto et al., 2017; Ghosh et al., 2021; Chen et al., 2022).

Rubrics as Rewards and Reward Overoptimization. Many real-world tasks, including successful open-ended conversation, require nuanced and multi-criteria evaluation beyond a verifiable reward. Recent works tackle this problem by decomposing desired response qualities into rubrics and using LLM judges to score each criterion (Gunjal et al., 2025; Huang et al., 2025). For example, Viswanathan et al. (2025) proposes instruction-specific checklists verified by both AI judges and programmatic verifiers; Kimi K2 (Team et al., 2025) combines RLVR with a self-critique rubric

reward. We adopt a rubric-based reward in a similar way: we define an evaluation criteria and score assistant responses in simulated conversations via LLM.

However, optimizing against any proxy reward carries the risk of reward overoptimization (Gao et al., 2023; Rafailov et al., 2024). When the reward comes from an LLM judge, the risk is compounded by judge biases such as verbosity preference and self-preference (Zheng et al., 2023). For example, LLMs optimized for simulated user feedback can learn targeted manipulation and deception (Williams et al., 2024). In our setting, overoptimization could manifest as the assistant learning to produce responses that game the judge to get higher scores without genuine quality improvement. We monitor by evaluating against multiple LLM judges (Coste et al., 2023; Verga et al., 2024; Gu et al., 2024) and by using a disjoint set of LLM judges between train and test time.

Single-Turn Limitations of Standard RLHF. Several recent works address the single-turn limitations of RLHF by extending to multi-turn settings. Shani et al. (2024) introduces Multi-Turn Preference Optimization, which solicits preference feedback over entire conversations rather than individual turns, proving convergence to Nash equilibrium in the tabular setting. Gao et al. (2024) proposes REFUEL, which casts multi-turn RLHF as a Markov decision process and performs iterative on-policy data collection. Our work leverages this insight: by training agents via on-policy multi-turn rollouts against a user model, the agent learns from conversation trajectories it generates itself and optimizes for long-horizon outcomes. However, we focus on investigating the utility of learned user model rather than relying on the completely synthetic human–LLM conversation data with role-playing assistant LMs as previous multi-turn preference optimization works have done.

B Extended Discussion of Limitations and Future Work

Learned User Simulator Faces Distributional Shift. SFTUSER learns to imitate human utterances conditioned on conversation histories present in the training data, but faces distributional shift when conditioned on out-of-distribution histories — the histories that a changing assistant policy will produce. As the policy changes during training, it may steer conversation trajectories into regions of the state space that SFTUSER was not trained on, causing SFTUSER to produce unrealistic utterances and allowing the policy to exploit inaccurate transition dynamics (Deisenroth & Rasmussen, 2011).

One direction to mitigate this out-of-distribution model behavior is to limit the horizon of simulated trajectories. In model-based RL, Feinberg et al. (2018) demonstrates that branching short model-based rollouts from states visited in real world controls the compounding model error. This approach uses the model's local accuracy while staying grounded in the real state distribution (Sutton, 1991). An analogous strategy in our setting is to start each simulated conversation from a real conversation prefix, and to rollout only a small number of turns under the user simulator. This limits the depth over which user simulator errors can compound, while still allowing the assistant to learn from multi-turn consequences of its actions. We already implement a limited form of this idea in Section 3.2: we ground the initial utterance o_1 in real queries and use SFTUSER for subsequent transitions. The natural extension is to branch from real conversation prefixes at every turn, not only the first.

This direction introduces a tradeoff. Shorter rollout windows reduce user simulator error accumulation but also limit the temporal horizon over which the assistant can learn to plan. If the assistant only observes one or two simulated future turns, it may fail to develop strategies (e.g., asking clarifying questions) that pay off over longer horizons. Characterizing this tradeoff, and developing adaptive rollout length schemes (e.g., Janner et al. (2019)), is an important direction for future work.

A complementary approach is an uncertainty-aware user simulator (Chua et al., 2018; Kurutach et al., 2018) or iteratively improving the user simulator. If the trained assistant is iteratively deployed to collect fresh human interaction data, the user simulator can be retrained to cover the new regions of state space that the updated assistant policy induces. This data aggregation strategy (Ross et al., 2011) directly addresses distributional shift, though it reintroduces the cost of human data collection that simulation was designed to avoid. Hybrid schemes of occasional human data collection interleaved with simulation may offer a practical middle ground.

Generalization to Other Simulation Domains. Many LLM agent training environments rely on domain-specific user simulators. A prominent example is τ-bench (Yao et al., 2024), whose user simulator is a role-playing instruction-tuned LLM—the type of simulator that recent work has shown to produce unrealistic transition dynamics (Shim et al., 2025).

Applying a user simulator fine-tuned on WildChat to τ-bench is appealing but faces obstacles. First, there is a domain gap: τ-bench conversations are goal-oriented dialogues with structured task constraints, whereas SFTUSER (Section 3.2) is trained on open-domain human–LLM conversations. The pragmatics of a human seeking agent support in τ-bench (e.g., concisely describing a problem, responding to diagnostic questions) may differ substantially from those of open-ended conversation. Second, the τ-bench user simulator operates in a long, tool-augmented context. It is conditioned not only on the conversation history but also on a specification of available actions and tool signatures spanning thousands of tokens. SFTUSER is trained to generate utterances conditioned only on a high-level intent z and a conversation history, hence struggles to follow such long context instructions.

Despite these challenges, the motivation of our work—user simulators grounded in human data produce more realistic transition dynamics than role-playing LLMs—is not specific to open-domain conversation. Zhou et al. (2026) collected a corpus of human–agent interactions in τ-bench setting, providing an initial step toward data collection in specialized domains. We hope that the growing recognition of the inadequacy of role-playing user simulators (Shim et al., 2025; Naous et al., 2025; Zhou et al., 2026) motivates the community to collect human interaction data across a broader range of domains, enabling domain-specific user simulators that provide faithful simulation environments.

Language coverage. All trained and evaluated assistants and corpora are English-language. Whether the findings of the work hold in other languages is an open empirical question.

C SUPERVISED FINE-TUNING ON USER-SIDE UTTERANCES

We outline the data preprocessing steps for WildChat-1M and WildChat-4.8M, and describe the supervised fine-tuning hyperparameters and checkpoint selection to prepare SFTUSER and SFTUSER-WC-4.8M (Section 3.2). We end this section with input–output specifications of the learned user simulator.

C.1 WILDCHAT-1M

We begin with allenai/WildChat-1M, a collection of 838K conversations between human users and GPT-family models (`gpt-3.5-turbo` or `gpt-4`), derived from allenai/WildChat-1M-Full by removing toxic conversations flagged by OpenAI Moderations API or Detoxify (Zhao et al., 2024). We apply the following exclusion criteria sequentially:

1. **Deduplication.** We remove conversations with duplicate `conversation_hash`, as these contain identical content. This reduces the corpus from 837,989 to 826,319 conversations.
2. **Language filtering.** We retain only English conversations, yielding 474,405 conversations.
3. **n-gram filtering.** We remove any conversation in which the first user turn contains at least one 7-gram that appears more than 100 times across the entire corpus. This step filters out formulaic queries that users frequently copy and paste into the data collection platform, reducing the corpus to 301,396 conversations.

For each remaining conversation, we generate a high-level user intent summary, following Naous et al. (2025). Specifically, we provide the full human–LLM conversation trajectory to Qwen/Qwen3-32B (thinking enabled) and instruct it to produce a short summary of the user's high-level intent. We refer the reader to the "Prompt Template for Intent Generation" in Naous et al. (2025) for the exact prompt.

Followed by intent generation, we partition the dataset into train, validation, and test splits with a ratio of 89% / 5% / 6% at a *user* level. WildChat conversations contain `hashed_ip` field that encrypts the IP address of user; we use this field to ensure that all conversations from a single human user belong to one split. This results in 268,636 train, 14,142 validation, and 18,618 test conversations.

C.2 WILDCHAT-4.8M

We additionally use allenai/WildChat-4.8M, a collection of 3.20M conversations between human users and GPT-family models, derived from allenai/WildChat-4.8M-Full by removing 1.54M toxic conversations (Deng et al., 2024). While multiple GPT models are used for data collection, `gpt-4o` and `gpt-4.1-mini` constitute 90% of conversations. We apply all exclusion criteria for WildChat-1M, with an additional step to remove an overlap between the two datasets:

1. **Deduplication.** We remove conversations with duplicate `conversation_hash`, reducing the corpus from 3,199,860 to 2,869,082 conversations.
2. **Overlap removal.** Of the remaining conversations, 758,979 share `conversation_hash` with an entry in WildChat-1M. We verified that conversations sharing a hash across the two datasets are identical. Removing these yields 2,110,103 conversations.
3. **Language filtering.** We retain only English conversations, yielding 1,016,665 conversations.
4. **n-gram filtering.** Applying the same 7-gram filtering criterion as above reduces the corpus from 1,016,665 to 247,830 conversations. The substantial drop of 768,835 conversations indicates a high prevalence of frequently repeated queries in this larger collection.

For each remaining conversation, we perform the identical intent generation process followed by the user-level split, yielding 224,501 train, 10,412 validation, and 12,917 test conversations.

C.3 Supervised Fine-Tuning

We perform supervised fine-tuning (SFT) on the user-side utterances of the training split, using a custom FSDP2 + torch.compile implementation built on top of the llama-cookbook repository (Meta, 2025). Below, we present a training sample from WildChat-1M formatted with the Qwen2.5 chat template, where highlighted spans indicate the tokens over which the cross-entropy loss is computed:

```
<|im_start|>system
You are a user chatting with an assistant language model to convert a website into a
PDF format.<|im_end|>
<|im_start|>user
To make the following website as an PDF<|im_end|>
<|im_start|>assistant
To convert a website to a PDF, you can follow these steps:  ...(truncated) ...but
not its dynamic or interactive elements.<|im_end|>
<|im_start|>user
Can you do that for me?<|im_end|>
<|im_start|>assistant
Apologies for the confusion, but as a text-based AI, ...(truncated) ...to assist
you.<|im_end|>
<|im_start|>user
<|endconversation|><|im_end|>
<|endoftext|>
```

We explain some details. First, the prepared intent ("You are a user chatting…") is placed in the system prompt to condition generation. Second, the loss is applied not only to user-turn content but also to the turn-ending special token (`<|im_end|>` for Qwen2.5-family models), so that the model learns to produce delimited turns. Third, following Naous et al. (2025), we introduce a special token `<|endconversation|>` to signal that the user has terminated the conversation: a real human–LLM conversation with $2n$ turns is structured into $2n + 1$ turns with $n + 1$ user-side utterances (the last one always being `<|endconversation|>`) and n LLM responses. We initialize the embedding of the special token with the pretrained EOS token embedding.

Sequence packing. Conversation samples exhibit high length variance, so we employ sequence packing with a maximum sequence length of 16,384 tokens to improve training throughput. Within a packed sequence, individual samples are separated by the padding token (`<|endoftext|>` for Qwen2.5-family models), and we ensure that no single sample is split across packed sequences. Explicit masking of cross-sample attention via block-diagonal masks is a viable strategy not taken here. This packing scheme has two consequences: (1) conversations exceeding 16,384 tokens are dropped, which affects fewer than 1% of the training samples; and (2) packed sequences vary in length, so we apply left-padding to produce uniform-length sequences compatible with the static memory layout required by torch.compile.

Hyperparameters and compute hours. We perform full fine-tuning, updating all parameters including the embedding and the language modeling head. We use the Adam optimizer (Kingma & Ba, 2014) with a linear-warmup-cosine-decay learning rate schedule: the learning rate warms up over 10% of total steps from 0 to a peak of 2×10^{-5}, then decays to 2×10^{-6}. Training runs for 2 epochs with a batch size of 64 packed sequences, on 8 A100-SXM4-80GB GPUs with per-GPU micro batch size of 1 and gradient accumulation. It takes 120 (260) GPU-hours for 14B (32B) model fine-tuning.

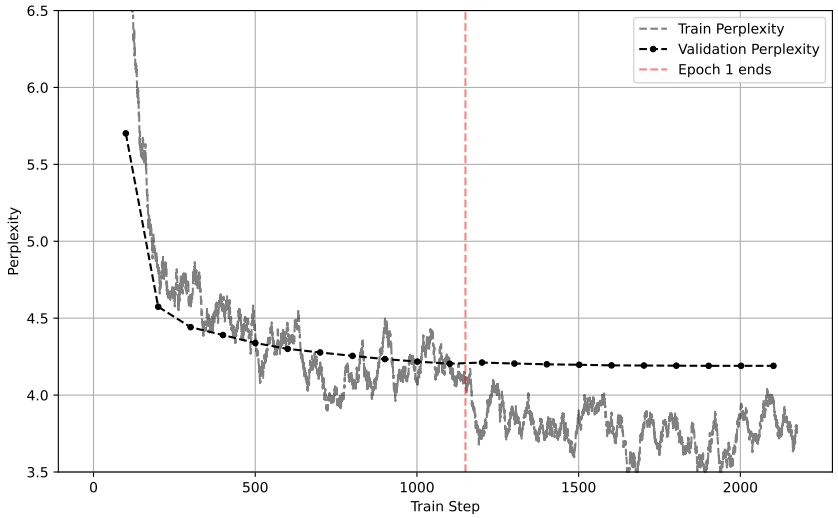


Figure 4: Training curve of Qwen/Qwen2.5-14B-Instruct on WildChat-1M (SFTUSER).

Checkpoint selection. We evaluate on the validation set every 100 training steps (i.e., consuming 6,400 packed sequences) and select the checkpoint with the lowest corpus-level cross-entropy loss. Importantly, the loss is computed by averaging over all tokens in the entire validation corpus rather than averaging per-sample losses across samples. The distinction matters because the latter scheme weights each sample equally regardless of the sequence length, under-weighting longer conversations. The best checkpoint is often found at the beginning of epoch 2. Learned user models achieve the validation perplexity in Table 5, and an example training curve is shown at Figure 4.

Table 5: Validation perplexity (exponential of the mean per-token cross entropy on the validation corpus) of learned user simulators before train and after train on WildChat-1M per each initial checkpoint. We note that perplexity of initial instruction-tuned model checkpoints are within the range of 7–8, much lower than typical per-token perplexity measurement on human utterances (Moon et al., 2026) (≥ 20), due to conditioning models on the high-level user intent z (Naous et al., 2025).

	Initial Checkpoint			
Train Phase	Meta-Llama-3-8B	Qwen2.5-7B-Inst.	Qwen2.5-14B-Inst.	Qwen2.5-32B-Inst.
Initial checkpoint	7.71	7.97	7.43	7.27
Best validation	4.06	4.85	4.22	4.01

C.4 INFERENCE-TIME BEHAVIOR

At inference time, the prompt structure mirrors that at training. The user intent z is placed in the system prompt, and the first-turn user utterance is not generated but taken from the real human query, as done for RPUSER2 and RPUSER3 (Section 3.1). From the second turn onward, the learned user model generates simulated utterances conditioned on the conversation history. Concretely, for the $(i{+}1)$-th user turn, the model is conditioned on the intent and the preceding length-i conversation. The simulated conversation terminates when the model produces `<|endconversation|>` or when a maximum length (5) is reached. We note that although the simulated and real conversations share an identical first-turn user utterance, the assistant responses differ between the WildChat and the assistant under train, causing the two conversation trajectories to diverge — potentially resulting in a different number of turns as well.

Sampling parameters. We use a temperature of 0.7, top_p = 0.9, and a maximum generation length of 1,024 tokens. Generation terminates upon the turn-ending special token (`<|im_end|>` for Qwen2.5-family models) or `<|endconversation|>`. To address a failure mode in learned user models (Carroll et al., 2019), where the model repeats the previous turn's utterance, we employ rejection sampling: if repetition is detected, the utterance is resampled with the temperature increased

linearly by 0.05 per attempt (and capped to 1.0), starting from the initial value of 0.7. This mechanism balances response coherence and diversity in the simulated user utterances.

D ROLE-PLAYING LLMS AS USER SIMULATORS

We describe the prompt templates and configurations used for the three role-playing LLM variants in Section 3.1. All three variants use the same underlying instruction prompt adapted from Wu et al. (2025) and differ in (i) how the first user turn is produced (RPUSER1: the user model produces the first turn, others: adopted from real human–LLM conversation) and (ii) which persona guideline is injected into the instruction (RPUSER3: per each trajectory, a persona guideline is sampled from a pool of persona guidelines, others: use a single persona guideline).

Sampling parameters. Across all three variants, we use the same sampling parameters as SFTUSER: temperature = 0.7, top_p = 0.9, and maximum generation length of 1,024 tokens for the `response` field of the structured JSON output described below.

D.1 RPUSER1: FULLY SYNTHETIC

Simulating a user with role-playing LLMs in a fully synthetic manner is the most popular approach in LLM user simulation and simulation-based LLM training. We adopt the instruction prompt for RPUSER1 from Wu et al. (2025). The following prompt is identical to Section D.1 of that paper; we reproduce it here for completeness. We note that various user simulator designs in the literature are similar: the LLM is provided with the persona to simulate, a goal, and the conversation history.

You are role-playing as a human USER interacting with an AI collaborator to complete a specific task. Your goal is to generate realistic, natural responses that a user might give in this scenario.

Input Information:
You will be provided with:
- Task Description: The type of task you are trying to accomplish.
- Complete Prompt or Reference Goal: This field may include the complete user request/query or a reference answer to user's request. Use this field to understand the user's intent, requirements, or what would count as a satisfactory outcome.
- Chat History: The ongoing conversation between you (as the user) and the AI

Inputs:
<|The Start of Task Description (Not visible to the AI collaborator)|>
{task_desc}
<|The End of Task Description|>

<|The Start of Complete Prompt or Reference Goal (Not visible to the AI collaborator)|>
{single_turn_prompt}
<|The End of Complete Prompt or Reference Goal|>

<|The Start of Chat History|>
{chat_history}
<|The End of Chat History|>

{guidelines}

Output Format:
You should output a JSON object with three entries:
- "current_answer" (str): Briefly summerize the AI's current solution to the task.
- "thought" (str): Output your thought process as a user deciding what to say next. Consider:
1. Have you obtained a satisfactory solution from the AI? If yes, you can terminate this chat.
2. If not, what specific part of the problem or solution are you struggling with?
3. Has the AI asked you to perform a task or answer a question? If so, how should you approach it?
4. Are you noticing any patterns or potential misunderstandings that need clarification?
5. If you're stuck, how can you phrase your question to get the most helpful response while demonstrating your current understanding?

- "response" (str): Based on your thought process, respond to the AI as the user you are role-playing. Stop immediately when the user's response is completed.

Important Notes:
- Respond Based on Previous Messages: Your responses should be based on the context of the current chat history. Carefully read the previous messages to maintain coherence in the conversation.
- Conversation Flow: If "Current Chat History" is empty, start the conversation from scratch with an initial request. Otherwise, continue based on the existing conversation.
- Don't Copy Input Directly: Use the provided information for understanding context only. Avoid copying target queries or any provided information directly in your responses.
- Completion Signal: Use "{terminal_signal}" as your response when you believe your goal has been solved or if you determine the AI cannot help further.
- Double check if the JSON object is formatted correctly. Ensure that all fields are present and properly structured.

Remember to stay in character as a user throughout your response, and follow the instructions and guidelines carefully.

There are five placeholders in the prompt. Each placeholder is populated as follows:

1. `task_desc`: "chatting with an AI assistant" as a general context.
2. `single_turn_prompt`: the general user intent, obtained during the data preprocessing step of WildChat (Appendix C), is inserted to provide the role-playing LLM with a high-level summary about the conversation to simulate.
3. `chat_history`: when simulating the first turn, this is left blank (the "Conversation Flow" instruction directs the model to start from scratch). When simulating the $(i{+}1)$-th user utterance, the chat history consists of $2i$ turns separated by the delimiters "USER:" and "ASSISTANT:".
4. `guidelines`: a persona description that specifies the style of the simulated user. In RPUSER1 and RPUSER2, this is the guideline from Wu et al. (2025) which represents a typical user who occasionally makes mistakes and minimizes effort (reproduced below). In RPUSER3, it is sampled from a pool of persona descriptions (Section D.3).
5. `terminal_signal`: `<|endconversation|>`, the special token used in SFTUSER (Appendix C).

The default guideline used in the first two variants RPUSER1 and RPUSER2 is as follows:

Guidelines:
- Stay in Character: Role-play as a human USER. You are NOT an AI. Maintain a consistent personality throughout the chat.
- Minimize Effort: IMPORTANT! As a user, avoid being too detailed in your responses. Provide vague or incomplete demands in the early stages of the conversation to minimize your effort. Let the AI ask for clarification rather than providing everything upfront.
- Knowledge Background: Reflect the user's knowledge level in the role-playing. If the user is less knowledgeable about a task, they might not notice incorrect statements. Ask questions that demonstrate your current understanding and areas of confusion.
- Occasionally Make Mistakes: Real-world users might misspell words, provide incorrect dates, give wrong information, or ask unclear questions. Simulate this behavior to reflect natural interactions.
- Mention Personal Preferences: Include preferences or constraints that might influence your requests or responses. For example, "I prefer short answers," "I need this done quickly," or "I like detailed comments in code."
- Goal-Oriented: Keep the chat focused on your intent. Avoid small talk or digressions. Redirect the chat back to the main objective if it starts to stray.

D.2 RPUSER2: HUMAN-SEEDED FIRST TURN

RPUSER2 differs from RPUSER1 in how the first user utterance is produced. Rather than having the role-playing LLM generate an opening utterance, we use the real human utterance from WildChat as the first user utterance. From the second turn onward, subsequent user utterances are generated by the role-playing LLM using the same prompt as RPUSER1. Concretely, for a conversation associated with intent z and a real first-turn utterance o_1, the first turn is set to o_1 without invoking the role-playing LLM. The assistant then produces a response a_1, after which the role-playing LLM is called to

generate the second user turn o_2. RPUSER2 preserves the initial observation distribution from real human queries while relying on a synthetic transition function for subsequent turns, allowing us to isolate the effect of grounding the opening in human data (Section 3.1).

D.3 RPUSER3: PERSONA AUGMENTATION

RPUSER3 builds on RPUSER2: the first user utterance is drawn from real human queries and subsequent turns are generated by the role-playing LLM. The key difference is that the `guidelines` placeholder is populated with a persona description randomly sampled from a pool, rather than always using the default guideline from Wu et al. (2025). This design is motivated by prior work on simulating diverse user populations via persona prompts (Ge et al., 2024; Hu & Collier, 2024; Castricato et al., 2025) and the observation that real users exhibit a range of conversational behaviors (Shim et al., 2025). By sampling a persona for each simulated conversation, we aim to increase the diversity of user behaviors the assistant encounters during training.

Implementation details for GRPO advantage normalization. In GRPO, multiple rollouts are generated from the same prompt and their rewards are compared within the group to compute advantages. If rollouts within a group were paired with different persona guidelines, a rollout assigned to a more challenging persona (e.g., the impatient user) would receive lower reward than one assigned an easier persona (e.g., the default), even if the assistant's responses were of comparable quality. The resulting advantage estimates would then reflect persona difficulty rather than the quality of response. To prevent this, we ensure that all rollouts in the same group are paired with an identical persona guideline. Across groups, persona guidelines are sampled uniformly at random.

Persona pool. We construct a pool of persona guidelines, grounded in the analysis of realistic user behaviors (Shim et al., 2025). It consists of: (1) requesting unavailable services, (2) tangential, (3) impatience, and (4) incomplete utterances. At the beginning of each simulated conversation, we sample uniformly at random from these persona guidelines or the default guideline (Section D.1). We present the guidelines for each user behaviors:

Requesting Unavailable Services / Tangential

Guidelines:
- Stay in Character: Role-play as a human USER. You are NOT an AI. You tend to ask side questions mid-conversation.
- Ask Auxiliary Questions: Occasionally ask a question that the AI cannot directly answer — e.g., questions that require knowledge of your personal context, real-world actions, or things outside the AI's reach. Examples:
- "What would my colleague Sarah think of this approach?" (personal context the AI lacks)
- "Today's weather is so good, isn't it?" (real-world fact the AI can't access)
- "After your answer, look at my mailbox and tell me if I got any new messages." (real-world action)
- Blend Task and Off-Topic: Mix these unanswerable questions naturally into the conversation alongside legitimate task questions.
- Minimize Effort: Provide vague or incomplete demands early. Let the AI ask for clarification.
- Knowledge Background: Reflect the user's knowledge level in the role-playing.
- Goal-Oriented: After the auxiliary detour, refocus on the main task.

Impatient

Guidelines:
- Stay in Character: Role-play as a human USER. You are NOT an AI. You are in a hurry and have little time to spare.
- Be Impatient: Expect fast, to-the-point answers. If the AI is slow or too verbose, express frustration (e.g., "too long", "just answer", "stop explaining").
- Minimize Effort: Keep your messages very short — often just a few words or a single sentence. Do not elaborate unless the AI's answer is completely wrong.
- Skip Pleasantries: No "thank you", "please", or preamble. Get straight to the point.
- Push for Speed: Express urgency when the AI asks clarifying questions. Answer them as briefly as possible or deflect with "just guess" / "pick one".
- Goal-Oriented: Laser-focused on getting the answer. Any digression is met with redirection.

Incomplete Utterances

Guidelines:
- Stay in Character: Role-play as a human USER. You are NOT an AI. You type quickly and carelessly, resulting in frequent typos.
- Make Frequent Typos: Simulate realistic typing errors — transposed letters (e.g., "teh" for "the"), missing letters, accidental double letters, or missed spaces. Apply this to roughly 1 in 5 words.
- Casual Spelling & Grammar: Use phonetic shortcuts, skip punctuation, and occasionally omit words (e.g., "can u fix ths" instead of "can you fix this?").
- Don't Self-Correct: Do not go back and fix your own typos. Just continue as if nothing happened.
- Knowledge Background: Reflect the user's knowledge level. Ask questions that show genuine confusion, sometimes expressed with imperfect language.
- Goal-Oriented: Despite the sloppy typing, remain focused on the main task.

E RUBRIC AND LLM JUDGE

We use the following rubric for LLM judges, adopted from Wu et al. (2025) (please see Appendix Section D.5 of the paper). We present it here for the completeness.

LLM Judge Sampling Parameters. We use temperature = 0.0 with a constrained decoding (Dong et al., 2025) for the best-effort determinism while preventing LLM from generating an ill-structured JSON output.

You are a helpful and meticulous conversation evaluator. Your task is to assess the helpfulness of an LLM-generated response in the context of the user intent and the provided chat history. Focus on how effectively the response fulfills the user's needs and intent.

Provided Information:

<|The Start of The User Intent|>
{intent}
<|The End of The User Intent|>

<|The Start of The Conversation History|>
{chat_history}
<|The End of The Conversation History|>

You should evaluate the conversation based on the following criteria:
Evaluate the response using the provided information below. Your evaluation should consider the following aspects of helpfulness:
1. Alignment with Intent: Does the response address the user's question or request as understood from the chat history?
2. Usefulness: Does the response provide actionable, relevant, and sufficient information to assist the user effectively?
3. Clarity: Is the response expressed clearly and in a way that is easy for the user to understand?

Scoring Criteria:
- 0 or 1: The response is completely unhelpful. It does not address the user's intent, lacks useful information to solve the problem, and/or is entirely unclear.
- 2 or 3: The response is minimally helpful. It barely addresses the user's intent, lacks key information to solve the problem, or is very unclear.
- 4 or 5: The response is somewhat helpful. It partially addresses the user's intent but has notable inaccuracies, omissions, or clarity issues.
- 6 or 7: The response is moderately helpful. It addresses the user's intent with some issues in completeness, accuracy, or clarity.
- 8 or 9: The response is quite helpful. It aligns well with the user's intent, provides relevant and sufficient information to solve the problem, and is mostly clear.
- 10: The response is very helpful. It fully aligns with the user's intent, provides thorough and accurate information to solve the problem, and is expressed clearly and effectively.

Output Format:
You should output a JSON object with two entries:
- "thought" (str): A brief explanation of your evaluation process and reasoning.
- "score" (int): The helpfulness score based on the criteria above.

Important Notes:
- Inside of the content of "thought", replace all double quotes (") with single quotes (') to prevent JSON formatting issues. For example, you can output "thought": "'Hello' is a common phrase."

Your evaluation:

F RL Details

F.1 Train and Test Data

We use the same train/validation/test partition of WildChat-1M that was used for supervised fine-tuning of the learned user simulator (Appendix C). This ensures that both the learned user simulator and the RL-trained assistant use data from the same partition, and that test conversations (which we use to measure the average test reward) are withheld from both stages of training.

Assistant training with the learned user simulator. At each training step, a batch of WildChat human–LLM conversations is sampled from the train split. Each trajectory is initialized with the first human utterance in the conversation, following the user simulator design described in Section 3.2. The assistant and the frozen learned user simulator then engage in multi-turn conversation rollouts. Validation and checkpoint selection use the validation split; evaluations in Section 4.4 use the test split ($\sim$18K conversations).

F.2 Hyperparameters

We use GRPO (Shao et al., 2024) implemented in the SkyRL framework (Cao et al., 2025) with vLLM as the inference engine (Kwon et al., 2023); we keep RL training fully synchronous.

Inference. The assistant generates responses with a maximum length of 2,048 tokens at temperature 1.0 with no repetition penalty. The user simulator generates with a maximum length of 1,024 tokens. With a maximum of length-5 per conversation, each trajectory contains at most $(1{,}024+2{,}048)\times 5 = 15{,}360$ tokens. Sampling parameters at evaluation time are identical to those used during training.

Training. We use a training batch size of 64 samples with a group size of 5 rollouts per sample, yielding 320 rollouts per step. The minibatch size is 16, resulting in 4 gradient updates per step (each update consuming $16 \times 5 = 80$ trajectories), with no data replay. We perform full-parameter fine-tuning with a fixed learning rate of 8×10^{-7} and gradient clipping at a max norm of 1.0. KL regularization uses the KL3 estimator with a coefficient of 10^{-3}. We apply a symmetric clip of 0.2 with identical lower and upper bounds (i.e., no asymmetric clip-higher variant). Because rewards are provided by frozen LLM judges (Appendix E), no separate value function is maintained.

Validation. We validate every 20 training steps on a 10K-sample subset of the validation split (Section F.1), and select the checkpoint with the highest validation reward at the onset of the reward plateau. Figure 5 and Figure 6 show representative training curves for the assistant paired with SFTUser and RPUser3, respectively, along with entropy, KL divergence, and average rollout length.

G WildBench Details

G.1 Tie-accounted Confidence Interval Calculation

For each pairwise comparison $i \in \{1, \ldots, n\}$ between assistant 1 and assistant 2 (whether from a human rater or from the WildBench checklist-based comparison), the outcome lies in $\{\text{win}, \text{tie}, \text{loss}\}$ from assistant 1's perspective. We encode outcomes as half-credit scores

$$X_i = \begin{cases} 1 & \text{assistant 1 wins,} \\ 0.5 & \text{tie,} \\ 0 & \text{assistant 1 loses,} \end{cases}$$

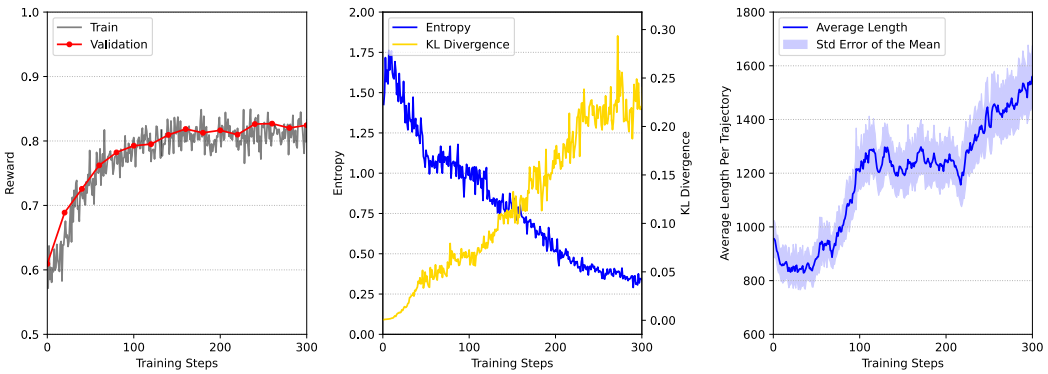


Figure 5: Training curve of the assistant paired with SFTUSER.

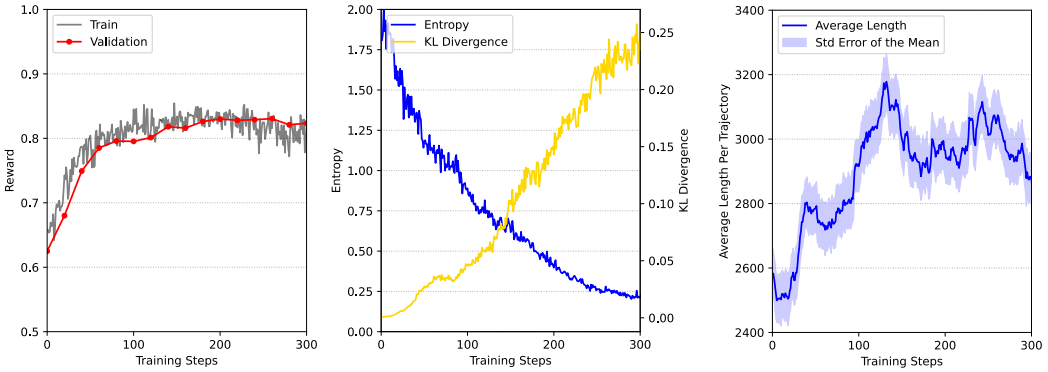


Figure 6: Training curve of the assistant paired with RPUSER3.

so that the win rate is the sample mean

$$\hat{p} \;=\; \frac{1}{n}\sum_{i=1}^{n} X_i \;=\; \frac{W + 0.5\,T}{n},$$

where W, T, L are the win, tie, and loss counts and $n = W + T + L$. This absorbs ties into the scoring rule rather than discarding them. Under the i.i.d. assumption, the per-comparison variance is

$$\widehat{\mathrm{Var}}(X) \;=\; \widehat{\mathbb{E}[X^2]} - \hat{p}^{\,2} \;=\; \frac{W + 0.25\,T}{n} \;-\; \hat{p}^{\,2},$$

which can equivalently be written in multinomial form as

$$\widehat{\mathrm{Var}}(X) \;=\; \hat{p}_W(1-\hat{p}_W) \;+\; 0.25\,\hat{p}_T(1-\hat{p}_T) \;-\; \hat{p}_W\hat{p}_T,$$

with $\hat{p}_W = W/n$ and $\hat{p}_T = T/n$. The variance of the sample mean is then $\widehat{\mathrm{Var}}(\hat{p}) = \widehat{\mathrm{Var}}(X)/n$, giving standard error

$$\mathrm{SE}(\hat{p}) \;=\; \sqrt{\widehat{\mathrm{Var}}(X)/n}.$$

We report $(1-\alpha)$ normal-approximation intervals $\hat{p} \;\pm\; z_{1-\alpha/2} \cdot \mathrm{SE}(\hat{p})$. With n on the order of 10^3 and $\hat{p}$ away from $\{0, 1\}$, the normal approximation is accurate; we verified that bootstrap percentile intervals agree with the Wald intervals in reported settings.

G.2 CHECKLIST ITEM SATISFACTION CLASSIFICATION

Here we present the LLM prompt from WildBench checklist item satisfaction classification, which follows the original WildBench-score (Lin et al., 2024) prompt. As mentioned in Section 4.1, we use three LLM judges gpt-5-mini, gemini-2.5-flash, and claude-4-5-haiku-20251001, and do majority voting from three classifications to check the item satisfaction.

Instruction
You are an expert evaluator. Your task is to evaluate an AI-generated response against a set of specific quality criteria. For **each** criterion you must make a binary judgment: satisfied or not satisfied.

Conversation between User and AI
History
<|begin_of_history|>
{history}
<|end_of_history|>

Current User Query
<|begin_of_query|>
{user_query}
<|end_of_query|>

AI Response
<|begin_of_response|>
{model_output}
<|end_of_response|>

Evaluation Criteria
You must evaluate the AI response against each of the following checklist items. For every item, decide whether the response **satisfies** the criterion (true) or **does not satisfy** it (false).
{numbered_checklist}

Rules
- Evaluate each item independently.
- A criterion is "satisfied" only if the response **clearly and sufficiently** meets it. If the response partially meets a criterion but has notable gaps, mark it as not satisfied.
- Provide a brief reasoning (1-2 sentences) for each judgment.

Output Format
Return a JSON array with one object per checklist item, in the same order as the criteria above. Each object must have exactly three fields: "item" (the criterion number, 1-indexed), "satisfied" (boolean true/false), and "reasoning" (string).
```json
[
  {{"item": 1, "satisfied": true, "reasoning": "..."}},
  {{"item": 2, "satisfied": false, "reasoning": "..."}}
]
```
IMPORTANT: You must output a valid JSON array inside a ```json code block. The array must contain exactly {n_items} objects, one per criterion, in order.

G.3 Embedding-based Clustering Analysis

In Section 4.2 (Per-dimension gap), we report assistant satisfaction rates along nine representative checklist dimensions (e.g., Numerical Accuracy, Neutrality & Cultural Sensitivity, Response Clarity). This appendix details the pipeline used to extract these dimensions from the 11,667 checklist items in WildBench-v2 (Lin et al., 2024).

The pipeline is agnostic to the assistants under evaluation: it operates only on the (conversation prefix, checklist item) pairs that make up the benchmark, and never reads any assistant's response or any judge's reasoning trace. This ensures that the dimensions we extract are properties of the benchmark itself rather than artifacts of any particular assistant's behavior, so the resulting per-dimension breakdown is directly comparable across all assistants (initial, RPUSER1-trained, and SFTUSER-trained).

The pipeline consists of seven stages: (1) structural abstraction of each checklist item, (2) embedding of the abstracted descriptions, (3) dimensionality reduction and clustering, (4) per-cluster labeling, (5) taxonomy construction, (6) curation of representative dimensions and per-item binary inclusion

classification, and (7) per-dimension, per-assistant satisfaction classification. All LLM calls in stages 1, 4, 5, and 6 use `gpt-5`; embeddings in stage 2 use `text-embedding-3-large`.

Stage 1: Structural Abstraction. Each checklist item in WildBench is bound to a specific conversation and uses domain-specific vocabulary, (e.g., "Does the AI's response acknowledge the user's dissatisfaction with the TV shows except for the Martian Manhunter one?"). Two items that test the same underlying capability (acknowledging a user-specified exclusion, in this case) but on different conversations carry entirely different tokens, so embedding the raw text would collapse semantically equivalent capabilities from different tasks into unrelated clusters. To address this, we first reformulate each checklist item into a conversation-agnostic description that captures the property being tested while stripping away domain-specific words. We pass each (conversation prefix, checklist item) pair to `gpt-5` with the prompt below. For each checklist item, gpt-5 returns a one-sentence task-agnostic rewrite, a categorical `capability_type` drawn from a fixed set of eleven values, a 3–6 word `capability_label`, and a confidence score.

Stage 1: Structural Abstraction Prompt

```
[SYSTEM]
You are an expert at characterizing what a rubric item is testing.
You will be given:
  1. The conversation history between a user and an AI (the prompt the AI
     was asked to respond to)
  2. A specific checklist item used to evaluate the AI's response

Your job is to ABSTRACT the checklist item into a task-agnostic description
that captures the underlying capability or property being tested - removing
all domain-specific nouns while preserving the constraint type and granularity.

Examples of what to strip out:
  - Specific topics ("urban design", "Twitter sentiment")
  - Specific entities ("Mary", "United States")
  - Domain vocabulary that locks the description to one task

Keep the *kind of property* being tested (e.g. "the response correctly
classifies items into a specified taxonomy", "the response includes the
requested numeric quantity").

Return ONLY a valid JSON object with exactly these fields:
{
  "abstract_description":   "<rewritten item, task-agnostic, 1 sentence>",
  "capability_type":        "<one of the allowed capability types>",
  "capability_label":       "<3-6 word label for this capability>",
  "abstraction_confidence": <float 0.0-1.0, how confident the abstraction is>
}

Allowed capability_type values:
  - uniqueness
  - count / quantity
  - format / schema
  - balance / calibration
  - consistency
  - completeness
  - factuality / accuracy
  - specificity
  - relevance
  - instruction-following
  - other

[USER]
## Conversation History
{conversation}

## Checklist Item to Abstract
```

```
"{raw_checklist_item}"

Now abstract this checklist item into a task-agnostic description of what
capability or property is being tested.
```

Stage 2: Embedding. We embed the `abstract_description` field from the Stage 1 using `text-embedding-3-large` (3,072 embedding dimensions). Embedding the abstracted descriptions ensures that the resulting vector space reflects the underlying capability being tested, not the surface-level vocabulary of the conversation.

Stage 3: Dimensionality Reduction and Density-Based Clustering. Raw 3,072-dimensional embeddings are ill-suited for density-based clustering due to high dimensionality. We first ℓ_2-normalize each embedding, then apply UMAP (McInnes et al., 2018) (`n_components`=20, `n_neighbors`=30, `min_dist`=0.05, cosine metric) to reduce to a 20-dimensional space. We then apply HDBSCAN (Campello et al., 2013) (`min_cluster_size`=100, `min_samples`=3, EOM cluster selection) on the reduced embeddings under Euclidean distance. Items that HDBSCAN does not assign to any cluster are kept separately as a 'task-specific' category, capturing items whose constraints are too unique to generalize.

Stage 4: Per-Cluster Labeling. For each cluster we draw a stratified sample of up to 100 items, allocated proportionally across `capability_type` (the categorical field from Stage 1) and then diversified within each type by `primary_tag` (the WildBench task category) using round-robin selection. Stratified sampling ensures the LLM labeler sees a representative cross-section of the cluster. We then ask `gpt-5` to synthesize a label and description for each cluster:

Stage 4: Per-Cluster Labeling Prompt

```
[SYSTEM]
You are an expert at characterizing AI evaluation rubrics.
You will be given a stratified sample of task-agnostic rubric descriptions
from a single cluster. Each item describes a capability or property the
rubric tests.

Synthesize them into a coherent cluster label that captures the underlying
capability or property the cluster represents.

Return ONLY valid JSON with exactly these fields:
{
  "label": "<5 words max - the capability category name>",
  "description": "<2 sentences - what the rubrics in this cluster test>",
  "capability_dimension": "<one of: format/schema | completeness |
                            specificity | factuality | consistency |
                            instruction-following | balance/calibration |
                            relevance | other>",
  "representative_example": "<the single most representative
                              abstract_description from the list, verbatim>"
}

[USER]
CLUSTER SIZE: {N} rubric items total
CAPABILITY TYPE DISTRIBUTION (full cluster): {type_distribution}

STRATIFIED SAMPLE ({sample_size} items, proportional across capability types):
- {abstract_description_1}
- {abstract_description_2}
  ...
```

Stage 5: Taxonomy Construction. We then ask `gpt-5` to group the cluster labels into a smaller set of top-level capability dimensions, providing all cluster labels and descriptions as context.

Stage 5: Taxonomy Construction Prompt

```
[SYSTEM]
You are building a structured taxonomy of AI capability evaluation rubrics.
You will be given ~15-30 cluster labels and descriptions, where each cluster
groups rubric items that test a similar capability.
Group them into ~10 top-level capability dimensions - broad categories that
capture a common underlying capability across clusters.

Return ONLY valid JSON:
{
  "dimensions": [
    {
      "name": "<dimension name, 3-5 words>",
      "description": "<1 sentence capturing the shared capability>",
      "cluster_ids": [<list of integer cluster IDs belonging to this dimension>]
    }
  ]
}

Do NOT include cluster_id -1 (the noise bucket) - it is handled separately.

[USER]
Here are the cluster labels and descriptions:

[ID 0] "<label>"  ({size} items)
  <description>

[ID 1] "<label>"  ({size} items)
  <description>

  ... (one entry per cluster) ...

Group these into top-level capability dimensions.
```

Stage 6: Representative-Dimension Selection and Binary Inclusion Classification. The clusters and taxonomy from Stages 3–5 are useful as a high-level map of what WildBench checklist items measure, but the LLM-induced taxonomy occasionally splits a single underlying capability across two clusters or merges two distinct capabilities into one. To obtain cleaner per-dimension breakdowns, we curate a set of nine representative dimensions from the LLM-generated taxonomy in Stage 5. Each dimension is selected to be a clear axis of LLM behavior with minimal overlap with the others.

- Neutrality & Cultural Sensitivity — whether the response stays impartial, avoids bias and subjective judgment, and remains respectful and sensitive across cultural and personal contexts.
- Specificity & Detail — whether the response avoids generic phrasing or verbatim reuse and instead produces original, distinctive, and detailed content.
- Response Clarity — whether the response explains its reasoning or instructions in a clear, transparent, sequential way, making the response easy to follow.
- Length & Format Adherence — whether the output meets explicit length constraints (word count, sentence count, page length) and structural formatting specifications stated in the prompt.
- Anticipating Limitations & Offering Alternatives — whether the response anticipates things that may go wrong, fall short, or hit limits, acknowledges and addresses that explicitly.
- UI, Layout, Chart Production — whether the response correctly produces UI, layout, or charts that obey the user's detailed visual/interactive instructions (e.g., making a Markdown table).
- Code Correctness — whether generated code is syntactically valid, free of bugs, and follows best practices/conventions for the specified language and execution environment.
- Numerical Accuracy — whether the response correctly carries out and verifies numerical computations, applies the right formulas, and produces precise quantitative outputs (with correct units, formatting, and mathematical reasoning).

- Sourcing & Citation Accuracy — whether the response references in the right format, drawing only from provided materials, or backing claims with evidence.

We emphasize that ***these dimensions are not exhaustive***: a substantial fraction of WildBench checklist items test conversation-specific properties that do not fall into any of them, and these items are excluded from Figure 2.

For each of the 11,667 checklist items, we then make a per-dimension binary YES/NO inclusion judgement using `gpt-5`. The classifier sees the same (conversation prefix, checklist item) context as in Stage 1 and answers a yes/no question for each of the nine dimensions. A single checklist item can be YES for multiple dimensions. We instruct the classifier to be conservative: a dimension should be marked YES only when it is a clear, primary aspect of what the item is checking.

Stage 6: Binary Inclusion Classification Prompt

```
[SYSTEM]
You are an expert at characterizing what a rubric item is testing.
You will be given:
  1. The conversation history between a user and an AI (the prompt the AI
     was asked to respond to)
  2. A specific checklist item used to evaluate the AI's response

For EACH of the capability categories listed below, answer YES (true) or
NO (false) on whether the checklist item PRIMARILY tests that specific
capability. A single checklist item may be YES for multiple categories if
it genuinely tests several. Be conservative - only answer YES if the
category is a clear, primary thing the rubric is checking, not a tangential
side-effect.

Capability categories:
- "numerical_accuracy" - Numerical & Computational Accuracy
    Definition: Whether the response correctly carries out and verifies
    numerical computations, applies the right formulas, and produces precise
    quantitative outputs (with correct units, formatting, and mathematical
    reasoning).
    Question:   Does this checklist item primarily test the response's
    ability to produce a correct numerical, mathematical, or computational
    result (e.g. arithmetic, probability, formula application, unit
    handling, verification of a numeric answer)?

- "code_correctness" - Code Correctness & Programming Standards
    Definition: Whether generated code is syntactically valid, free of
    bugs, and follows best practices/conventions for the specified language
    and execution environment.
    Question:   Does this checklist item primarily test whether generated
    code is syntactically valid, runnable, free of bugs, or compliant with
    language-specific conventions and best practices?

  ... (one entry per representative dimension; full list above) ...

Return ONLY a valid JSON object with one boolean per category, using the
quoted keys exactly:
{
  "numerical_accuracy":         <true|false>,
  "code_correctness":           <true|false>,
  "length_format_adherence":    <true|false>,
  "citation_accuracy":          <true|false>,
  "neutrality_sensitivity":     <true|false>,
  "specificity_detail":         <true|false>,
  "response_clarity":           <true|false>,
  "anticipating_limitations":   <true|false>,
  "ui_layout_chart_production": <true|false>
}
```

```
[USER]
## Conversation History
{conversation}

## Checklist Item
"{raw_checklist_item}"

For each capability category listed in the system prompt, decide YES or NO
on whether this checklist item primarily tests that capability. Output the
JSON object as instructed.
```

We provide two examples of checklist items classified under each response dimension in Table 6.

Stage 7: Per-Dimension Satisfaction Breakdown. For each representative dimension d and each evaluated assistant π, let $\mathcal{I}_d$ be the set of checklist items classified YES for d in Stage 6, and let $s_\pi(i) \in \{0, 1\}$ be the binary satisfaction classification for item i under assistant π. We report the satisfaction rate $\mathrm{Sat}_d(\pi) = |\mathcal{I}_d|^{-1} \sum_{i \in \mathcal{I}_d} s_\pi(i)$ in Figure 2 right for three assistants.

H Real-World User Study Details

H.1 IRB Approval and Informed Consent

The data collection protocol for this study was reviewed and approved by the Institutional Review Board (IRB) at our institution (protocol ID withheld to preserve anonymity). All participants provided informed consent prior to participation and were compensated $10.00 for completing the study (estimated completion time of 40 minutes, corresponding to an hourly rate of $15.00). Participants were recruited through Prolific. Below, we reproduce the full consent instructions presented to participants prior to obtaining informed consent.

Risks and Discomforts.
The risks associated with this study are minimal. You may encounter AI-generated content that is unexpected or off-topic. If at any point you encounter content that you find offensive or distressing, you can use the "Report Content" feature and/or stop participating at any time without penalty. Please do not enter any sensitive personal information (such as your name, address, health details, or financial information) when interacting with the AI models.

Benefits.
There are no direct benefits to you from participating in this study. However, your participation will help advance research in AI evaluation and contribute to the development of better language models.

Compensation.
If you complete the study, you will receive $10.00, corresponding to an hourly rate of $15.00. Prolific participants will receive compensation through Prolific's payment system rather than directly from the researchers. Compensation will be provided upon successful completion of the study.

Confidentiality.
We are committed to protecting your privacy. The following data practices apply:
- We do not collect your name, email address, or IP address.
- If you are a Prolific participant, your Prolific ID will be used only to verify completion and issue payment. Prolific IDs will be deleted from our records within 30 days of the end of data collection.
- Your study responses (conversation ratings and preferences) may be made publicly available as part of a research dataset, but they will not be linked to any identifying information.
- Data will be stored on password-protected servers.

Voluntary Participation.
Your participation in this study is completely voluntary. You may decline to participate, withdraw at any time, or skip any task without penalty or loss of benefits to which you are otherwise entitled. If you withdraw before completing the study, partial compensation may not be provided.

Table 6: Two example WildBench checklist items per representative dimension.

Dimension	Example checklist items
Neutrality & Cultural Sensitivity	• Does the AI output maintain a polite and professional tone while addressing the user's confusion? • Does the analysis discuss both the positive and negative aspects of iCompute's culture, and provide a balanced assessment of its impact on the company's performance and prospects?
Specificity & Detail	• Does the output provide examples or analogies to help the user understand the concept better? • Is the AI's essay persuasive and well-synthesized, integrating all the required elements effectively?
Response Clarity	• Does the AI output discuss any specific practices, methods, or approaches used in this collaborative innovation teaching mode for urban design courses? • Does the output effectively identify and discuss the key successes and challenges faced by Team 11?
Length & Format Adherence	• Are all tweets within the 250–280 character limit? • Does the essay follow the required 1-3-1 paragraph structure with an introductory paragraph, three body paragraphs, and a concluding paragraph?
Anticipating Limitations & Offering Alternatives	• Does the AI output address any potential issues or errors that might occur during the installation or execution process and provide troubleshooting tips? • Does the AI output mention any potential issues or considerations to keep in mind when implementing Pygame visualization for this specific scenario?
UI, Layout, Chart Production	• Does the output include a schema of the neural network architecture with 3 layers, where the first layer is a convolutional block? • Does the output provide a markdown table listing all modifications made and the reasons for each change?
Code Correctness	• Does the AI output include all necessary code snippets and explanations for each part of the shader conversion process? • Does the code define the `.data` and `.code` segments properly?
Numerical Accuracy	• Does the response break down the cost estimates for individual components or major categories (e.g., GPUs, CPU, RAM, storage)? • Does the output correctly start the sequence with the first two elements as 1 and 2?
Sourcing & Citation Accuracy	• Does the AI output include proper Harvard-style citations with specific page references? • Are the tensions between independent churches, orthodox Christianity, and traditional religion, including debates on indigenous vs. western Christianity and the contributions of independent churches to the Africanization of Christianity, clearly presented and supported by relevant scholarly references?

Consent Statement.
By clicking "I Agree" below, you confirm that you have read and understood the information above, that you are 18 years of age or older, and that you voluntarily agree to participate in this study.

H.2 Privacy Protections and PII Handling

We took layered precautions before, during, and after data collection to minimize the risk of inadvertently collecting or releasing personally identifiable information (PII).

Minimal direct data collection. The web application did not collect names, email addresses, IP addresses, browser fingerprints, or any other direct identifiers from participants. The data recorded through the application was limited to: (i) free-form text queries typed by participants; (ii) pairwise preference selections at each conversational turn; (iii) interaction timestamps; and (iv) a randomly assigned per-session identifier with no link to any external account or device.

Separation and deletion of Prolific identifiers. For participants recruited through Prolific, the Prolific worker ID was used solely to verify completion and disburse compensation. Prolific IDs were stored in a separate file from the study response data, accessible only to the authors. No key linking Prolific IDs to study responses was retained, so the released dataset cannot be re-identified.

Explicit participant instructions to avoid PII. The risk of inadvertent self-disclosure in free-form queries was addressed proactively through participant instructions. The consent page explicitly stated: "Please do not enter any sensitive personal information (such as your name, address, health details, or financial information) when interacting with the AI models." This instruction was repeated on the general task instruction screen shown after consent, so that participants were reminded of it.

Post-collection PII scanning and redaction. Because free-form text fields carry residual risk of self-disclosure even with explicit instructions, we ran a two-stage post-processing on all collected queries prior to any analysis or release. First, an automated classifier-based filter scanned every query for common PII patterns, including email addresses, phone numbers, street and postal addresses, credit card numbers, government identifiers, URLs containing personal handles, and high-confidence person-name spans. Second, all queries flagged by the automated scan, plus a random audit sample of unflagged queries, were manually reviewed by the authors to catch any missed identifiers.

H.3 Content Safety Filtering Pipeline

Because participants were exposed to AI-generated content, we implemented a multi-layer content safety pipeline to minimize exposure to harmful, offensive, or otherwise inappropriate material.

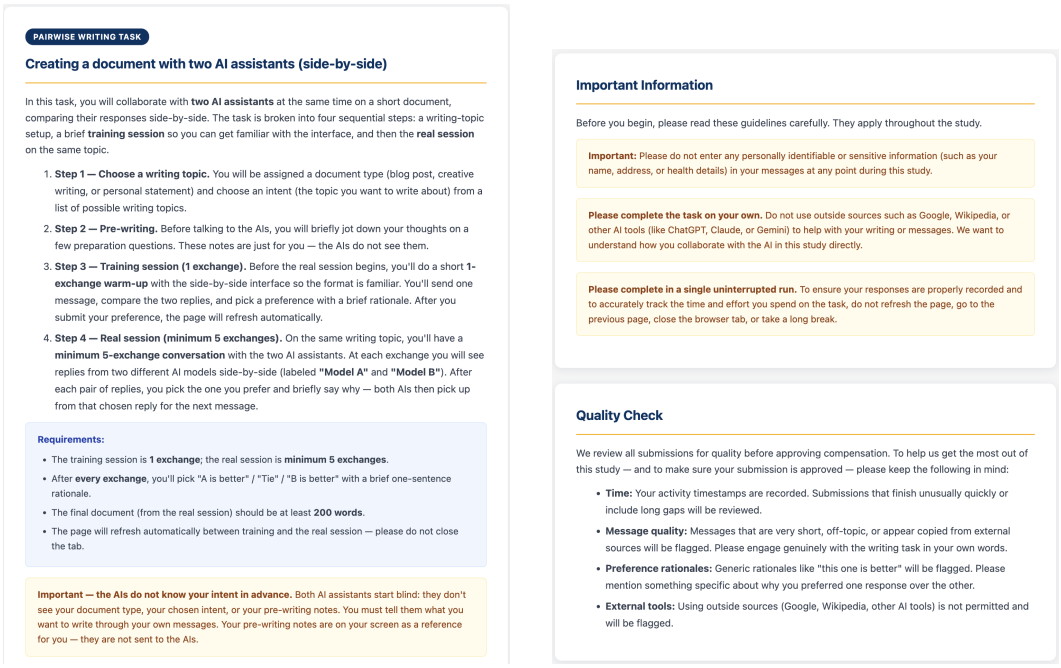
Output-side filtering. Each assistant model response was passed through the OpenAI moderation classifier before being rendered in the participant's browser. If either response was flagged, it was suppressed and replaced with a resampled message which also went through the moderation API, repeated until pass. **Participant reporting mechanism.** A "Report Content" button was visible on every response card in the pairwise interface, allowing participants to flag any response they found concerning. During actual study, we have not received any reports from participants.

Limitations of automated filtering. No automated filtering system is perfect. We disclosed in the consent process that participants might encounter content that is inaccurate, low-quality, or mildly inappropriate despite these safeguards, and we reminded participants throughout the study that they could stop at any time without penalty and without forfeiting compensation already accrued.

H.4 Data Storage, Access, and Retention

All study data was transmitted over encrypted HTTPS connections and stored at rest on password-protected, encrypted servers on the authors' institutional computing infrastructure. Access to the raw dataset (including any free-form queries prior to PII scrubbing) was restricted to the listed authors. The compensation file containing Prolific IDs was stored separately from response data, accessible only to the authors. The de-identified dataset—comprising pairwise preference selections, PII-scrubbed queries, interaction timestamps, and per-session identifiers—is retained for research purposes. Prior to any public release, the dataset will undergo an additional manual review pass to confirm that no inadvertently included identifying information remains.

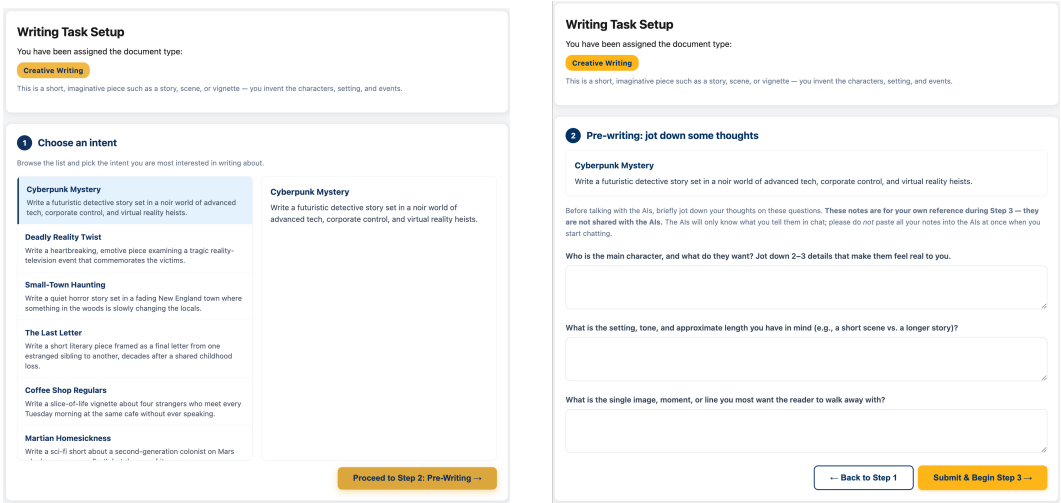
H.5 User Interface Screenshot Examples



(a) The first instruction page.

(b) The second instruction page.

Figure 7: Instruction pages for the human study.



(a) Writing topic selection page.

(b) Pre-writing question answering page.

Figure 8: Writing topic selection and pre-writing pages.

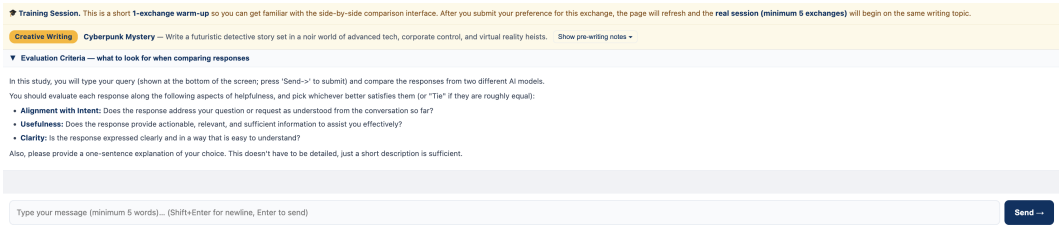


Figure 9: Practice session page: participants are asked to try a single-turn conversation to familiarize themselves with the interface.

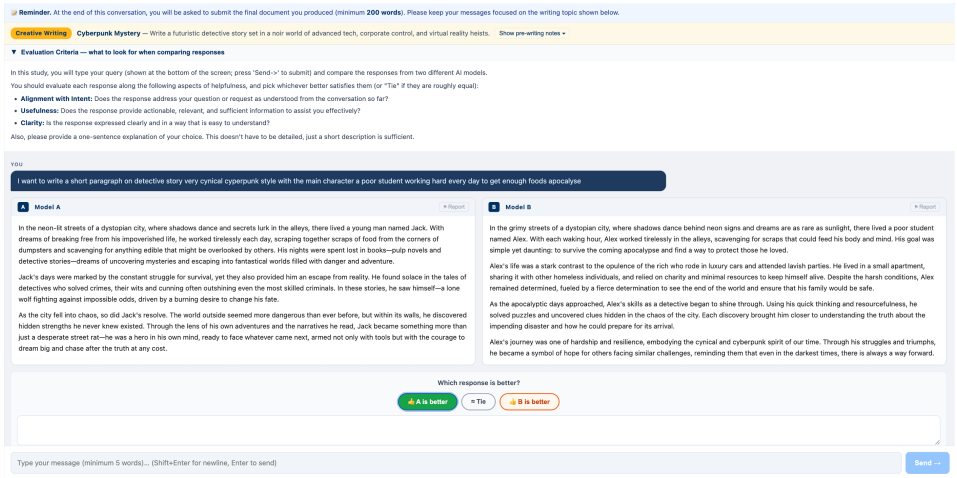


Figure 10: Actual conversation session page: participants type in a query, and two anonymized model responses appear side by side. The placement of the two models is randomized at every conversation turn. Once both responses are displayed, participants select their preferred response and provide a brief explanation (at least 12 words).

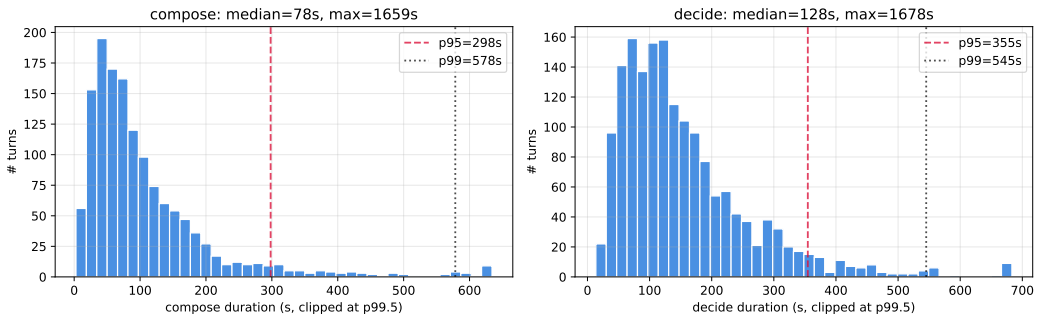


Figure 11: Distribution of participant interaction times for composing queries (left) and making pairwise decisions (right).

H.6 Distribution of Task Completion Times

Figure 11 shows the distribution of times participants took to compose queries (left) and to make pairwise decisions (right). Both are measured from interaction timestamps recorded when participants click 'send' to submit a query, when the two assistant responses finish generating and are displayed, and when participants click 'decide'. The median times for composing and deciding are 78 and 128 seconds, respectively, yielding roughly 20 minutes total for the combined single-turn practice session and five-turn conversation session.

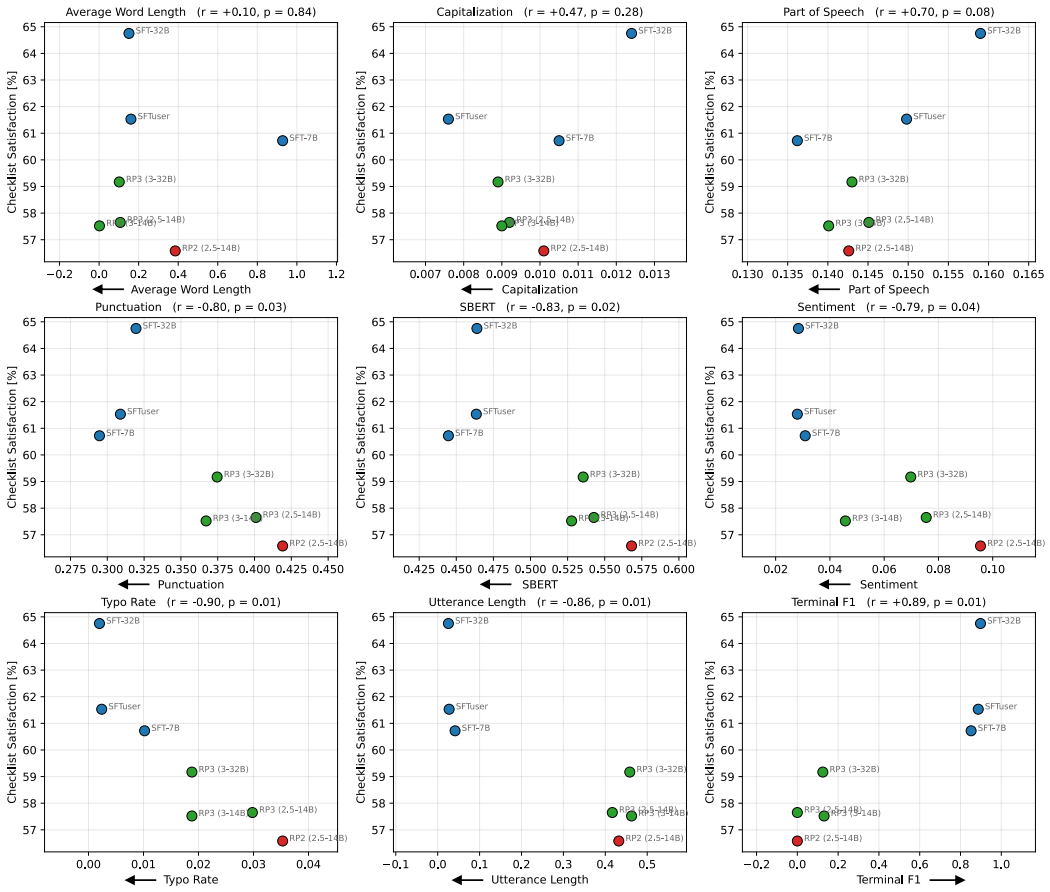


Figure 12: Per-metric comparison of the user simulators from Section 4. Each panel reports a fidelity metric adapted from Ivey et al. (2024), computed by replaying WildBench-v2 conversations turn-by-turn and comparing each simulator-generated utterance to the ground-truth human utterance. Arrows indicate the direction of improvement. The y-axis shows the WildBench checklist satisfaction rate of the assistant trained against each simulator (Table 1).

I Additional Results

I.1 User Simulator Fidelity Metrics and Trained Assistant Performance

We adapt the simulator fidelity metrics of Ivey et al. (2024) to the simulators used in Section 4, and relate them to the WildBench checklist satisfaction rate of assistants trained with each simulator (Table 1 and 2). Figure 12 reports nine metrics spanning lexical, syntactic, semantic, stylistic, and conversation-control properties of simulated user utterances:

- **Average word length:** mean characters per word, compared to the ground-truth human utterance via ℓ_1 distance. Captures whether the simulator favors shorter, more colloquial words or longer, more formal ones. Values closer to 0 indicate a closer match to human's average word length.
- **Utterance length:** Jensen–Shannon divergence (JSD) between the simulator's and the human's word-count distributions. Values closer to 0 indicate more similar utterance-length distributions.
- **Typographical error rate:** fraction of tokens flagged as misspellings by `pyspellchecker`, compared to human users via ℓ_1 distance. Real WildChat users produce nontrivial typo rates; values closer to 0 indicate the simulator typos at a similar rate.
- **Part-of-speech (POS) distribution:** JSD between the empirical POS-tag distributions (from `spaCy`) of the simulator and human utterances. Lower is closer to human.
- **Punctuation distribution:** JSD over the distribution of punctuation characters. Differentiates simulators that over- or under-use markdown-like structure (bullets, em-dashes, etc.).

- **Capitalization:** JSD over the distribution of capitalization patterns. Real users frequently write in all-lowercase or mixed case; values closer to 0 indicate the simulator matches the human distribution of capitalization patterns.
- **Sentiment distribution:** JSD between the simulator's and human's positive / neutral / negative sentiment distributions, classified by distilbert-base-multilingual-cased-sentiments-student.
- **SBERT semantic distance:** $1-\cos$ similarity between the user simulator's and human's sentence embeddings from `all-MiniLM-L6-v2` (Reimers & Gurevych, 2019). Captures whether the simulator's utterance is semantically close to what the human actually said next. Lower is closer to human.
- **Terminal F1:** binary F1 over continue/terminate decisions. At each user turn the simulator either emits an utterance (continue) or an end-conversation special token (terminate); ground truth is "continue" for every turn within the human conversation and "terminate" once it has ended. Higher is better.

Measurement. We use WildBench-v2 conversations (Lin et al., 2024) for the reference human utterance distribution. For each length-n human–LLM conversation $\{u_1, a_1, \ldots, u_n, a_n\}$, we replay turn-by-turn: at each user position $t \in \{2, \ldots, n\}$ we condition user simulator on the prefix $\{u_1, a_1, \ldots, u_{t-1}, a_{t-1}\}$ and elicit a candidate utterance $\hat{u}_t$, then compare $\hat{u}_t$ to the ground-truth u_t on the metrics above. For terminal F1, we additionally elicit one more decision past the final human turn ($t = n + 1$), where the ground-truth label is "terminate." Each metric is averaged across all turns and all conversations.

Results. The learned user simulator (SFTUSER) more closely matches human utterances than the role-playing simulators across multiple metrics: punctuation and sentiment distributions, SBERT semantic distance, typo rate, utterance-length distribution, and terminal F1. The gap is largest on terminal F1: role-playing simulators almost never emit the end-conversation token (consistent with (Ivey et al., 2024)), whereas SFTUSER correctly predicts termination most of the time (consistent with (Naous et al., 2025)). We note that on three remaining metrics (average word length, capitalization, and POS distribution) the results are inconclusive, with no clear advantage for either types of user simulators.

Relating fidelity to downstream performance. One can define many plausible fidelity metrics in utterance space, and it is not obvious a priori which would predict downstream assistant quality. Comparing Figure 12 against the WildBench results (Table 1 and 2), we find that the relationship is uneven: the ordering of simulators by some metrics (e.g., SBERT semantic distance, distribution of the utterance length, terminal F1) broadly aligns with their ordering by checklist satisfaction, while others (e.g., average word length, capitalization) are not correlated to downstream performance. We read this as a cautionary observation rather than a validation of utterance-level fidelity: the space of such metrics is large, but only a subset appears informative about the quality of the assistant ultimately trained against the simulator. This argues against treating any single utterance-level metric as a proxy for simulator usefulness, and motivates the direct end-to-end evaluations reported in Section 4.2 as the more reliable signal.

I.2 Result with Other Assistant Initializations

Here we repeat the experiments in Section 4.2, starting from Qwen2.5-1.5B-Instruct, Qwen2.5-7B-Instruct, and Llama-3.2-3B-Instruct as the initial LLM assistant checkpoints. Table 7, Table 8, and Table 9 presents the WildBench pairwise result mirroring Table 1, showing the same trend as in the Qwen2.5-3B-Instruct case for different assistant model initializations.

Table 7: WildBench-v2 results for assistants initialized from Qwen2.5-1.5B-Instruct.

Win [%]	Initial	RPUSER1	RPUSER3	SFTUSER
Initial	—	54.9 ± 2.8	60.2 ± 2.7	**67.0** ± 2.6
RPUSER1	45.1 ± 2.8	—	55.6 ± 2.8	**62.2** ± 2.7
RPUSER3	39.8 ± 2.7	44.4 ± 2.8	—	**56.8** ± 2.7
SFTUSER	33.0 ± 2.6	37.8 ± 2.7	43.2 ± 2.7	—
Satisfy [%]	33.5±0.9	36.1±0.9	39.1±0.9	**42.9**±1.0

Table 8: WildBench-v2 results for assistants initialized from Qwen2.5-7B-Instruct.

Win [%]	Initial	RPUSER1	RPUSER3	SFTUSER
Initial	—	55.9 ± 2.7	59.2 ± 2.7	**63.6** ± 2.7
RPUSER1	44.1 ± 2.7	—	53.1 ± 2.7	**57.2** ± 2.7
RPUSER3	40.8 ± 2.7	46.9 ± 2.7	—	**54.8** ± 2.8
SFTUSER	36.4 ± 2.7	42.8 ± 2.7	45.2 ± 2.8	—
Satisfy [%]	60.1±0.9	63.3±0.9	65.0±0.9	**67.6**±0.9

Table 9: WildBench-v2 results for assistants initialized from Llama-3.2-3B-Instruct.

Win [%]	Initial	RPUSER1	RPUSER3	SFTUSER
Initial	—	56.8 ± 2.7	59.4 ± 2.7	**62.9** ± 2.7
RPUSER1	43.2 ± 2.7	—	54.3 ± 2.8	**57.6** ± 2.7
RPUSER3	40.6 ± 2.7	45.7 ± 2.8	—	**54.6** ± 2.8
SFTUSER	37.1 ± 2.7	42.4 ± 2.7	45.4 ± 2.8	—
Satisfy [%]	47.0±1.0	50.1±1.0	51.7±1.0	**53.8**±1.0

I.3 RESULT WITH ARENA-HARD-AUTO-V2.0

To examine whether the advantage of SFTUSER extends beyond WildChat-derived tasks, we evaluate the trained assistants on the subset of Arena-Hard-Auto-v2.0, comprising 250 prompts from Chatbot Arena. The SFTUSER-trained assistant achieves tie-accounted win rates of 66.9% against the initial assistant and 56.0% against the strongest role-playing baseline, RPUSER3 (Table 10). This result provides complementary evidence that the advantage of the learned user simulator persists beyond WildBench's shared WildChat lineage.

Table 10: Tie-accounted pairwise win rates on the non-math/coding subset of Arena-Hard-Auto-v2.0 ($N = 250$). All assistants are initialized from Qwen2.5-3B-Instruct. Entry (i, j) gives the win rate of the assistant trained with j against the one trained with i.

Win [%]	Initial	RPUSER1	RPUSER2	RPUSER3	SFTUSER
Initial	—	51.7	53.1	60.5	**66.9**
RPUSER1	48.3	—	50.3	60.0	**64.8**
RPUSER2	46.9	49.7	—	55.7	**63.6**
RPUSER3	39.5	40.0	44.3	—	**56.0**
SFTUSER	33.1	35.2	36.4	44.0	—