# ROBUST LLM-BASED USER SIMULATION WITH TRAINED LATENT-STATE DYNAMICS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

User simulators for training and evaluating LLM-based conversational agents must perform well off-policy (i.e., when facing novel agents). Recent work has argued that explicitly modeling user *latent state* can improve simulator fidelity and off-policy robustness, but treating that state as a latent variable in LLM-based user simulation faces challenges such as intractability and unidentifiability given observational data. Consequently, latent-state simulation has generally been approached heuristically. We develop RUSTLED, a principled probabilistic framework for latent-state learning that formalizes fine-tuning on latent-state labels as posterior-regularized maximization of the marginal likelihood of observed conversational data, smoothly interpolating between teacher distillation and autoencoding. Our *self-normalized importance sampling* EM algorithm (SNIS-EM) addresses the problems facing current heuristic approaches. Beyond offering sound probabilistic foundations for latent-state simulation, RUSTLED yields simulators that generalize better to unseen agents than natural baselines, both in the likelihood of held-out human conversations and in reproducing human behavior.

## 1 INTRODUCTION

User simulation is an important technique for evaluating and optimizing user-facing systems (e.g., recommenders, interactive search, and decision-support agents) that involve multi-turn interaction with their users (Schatzmann et al., 2007a; Ie et al., 2019; Liu et al., 2019). This is no less true for *large language model (LLM) conversational agents*, where user simulators must replicate the conversational skill and behavioral tendencies of real users. For this reason, LLM-based user simulation has emerged as a promising means for generating user conversational behaviors. While prompting LLMs with personas has been the predominant approach (Davidson et al., 2023; Wang et al., 2023; Jin et al., 2025; Laban et al., 2026), more recent approaches using supervised fine tuning (SFT) and reinforcement learning (RL) have started gaining traction (Wang et al., 2025; Meshi et al., 2026; Naous et al., 2026; Liu et al., 2026; Wu et al., 2026a; Zhang et al., 2026).

A number of desiderata have been identified for high-quality LLM-based simulation, among them realism, unbiasedness, and controllability. Here we focus on *off-policy (or counterfactual) robustness*, i.e., the ability for a simulator to reflect realistic user behaviors when facing *new, unseen* conversational agents—this is critical when simulation is used to evaluate, generate and optimize agent policies (e.g., using RL). Otherwise, a simulator trained on data from existing agent policies may perform poorly out-of-distribution (OOD). Recent evidence suggests that modeling *user latent state* in generative simulation improves counterfactual robustness, giving better generalization and simulator performance with novel policies (Tennenholtz et al., 2026; Wu et al., 2026b). One explanation for this: a simulator that generates a user's internal state (e.g., satisfaction, confusion, goal/intent changes, etc.) given the current conversation history prior to generating its next utterance encourages an LLM-based simulator to more directly reason about the underlying causal mechanisms that drive a user's observable behavior (utterances), not just the utterances themselves.

Translating this view into practical simulation algorithms is non-trivial. Treating latent state as a genuine latent variable—and inferring it from data rather than simply imposing labels—faces several hurdles. First, it is *intractable*, with a posterior that is a distribution over language token sequences admitting neither a reparameterization nor tractable marginalization. Second, it is *unidentifiable*, since observed conversational data constrains the utterance distribution, not the latent space that



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.

generates them; thus latent spaces with different semantics can work equally well. Consequently, current practice w.r.t. latent state simulation tends to be heuristic: an external LLM is prompted to label conversations with latent state or user thinking traces, and the simulator is fine-tuned as if these labels had been observed ground-truth. While simple and effective, it ignores whether the latent-state labels effectively explain the conversational data on which simulation training is based.

We address these challenges by providing firm probabilistic foundations for dynamic latent-state labeling and simulator training, specifically, formalizing SFT with LLM-generated labels as posterior-regularized maximization of the marginal likelihood of the training data, treating the LLM labeler as a *tunable-strength regularization anchor*, rather than as data. The resulting algorithm is indexed by a single parameter $\alpha$, where $\alpha \to 0$ corresponds to direct distillation of the labeler, while $\alpha \to 1$ captures normal marginal-likelihood estimation. Our approach also sheds light on other theoretical shortcomings of current practice. We show empirically that our technique generates high-fidelity simulators that generalize well off-policy.

**Contributions.** Our contributions are as follows. (1) A probabilistic formulation of label-based latent-state learning, and an exact variational objective that allows precise tradeoffs between observed-behavior prediction and a labeler prior. (2) Well-founded constraints on the design of latent-state labelers. (3) The development of a *self-normalized importance sampler* with controllable variance. (4) Compelling empirical evidence that SFTed simulators with dynamic latent state exhibit strong counterfactual robustness, i.e., correspondence with observed off-policy behavior.

## 2 Problem Formulation

We begin with our problem formulation, first detailing LLM-based user simulation for training conversational agents, then motivating and formalizing the use of latent state in simulation. A detailed discussion of related work can be found in Appendix A.

**LLM Agent Optimization.** We assume an *LLM agent* $A$, interacting with users $i \in S$ drawn from some population, embodies a *policy* $\pi$ mapping observable user context $C_i$ and conversation history $H_t = (a_1, u_1, \ldots, a_{t-1}, u_{t-1})$ to stochastic choice of agent utterance $a_t$,[1] where $u_k$ is the user's utterance at turn $k$. The quality of $\pi$ is determined by the value it induces for user $i$, which depends on $i$'s intent, context, preferences, beliefs, personality, etc. For example, in a recommendation setting, this value is determined by: (i) whether a recommendation is made and accepted; (ii) $i$'s preferences/utility for that recommendation; (iii) whether $i$ trusts the recommendation; (iv) the cost (e.g., time, cognitive burden, intrusiveness) of the interaction; among other factors. Notice that these factors are not known *a priori* to $A$, though beliefs about these can be formed from $(C_i, H_t)$. Thus agent optimization can be viewed, conceptually, as solving a *partially observable Markov decision process (POMDP)*, where accrued reward can only be estimated, not directly observed.

**LLM-based User Simulators.** While evaluation of agent performance must ultimately be determined by engagement with real users, live experiments are costly, time-consuming and risky w.r.t. user safety, reputational harm, etc. This is especially true during agent *training or experimentation*, when performance may be far from optimal. For this reason, *user simulation* has become the method of choice for offline evaluation and training prior to testing high-quality policies in live settings (Ie et al., 2019; Liu et al., 2019; Hsu et al., 2024). In the case of LLM agents, where users engage in conversation, user simulators are themselves LLMs, prompted or trained to "speak" like human users in the application domain in question.

User simulation has a long history in conversational systems, predating the emergence of LLMs (Schatzmann et al., 2007a). LLM-based simulation originally focused on the use of prompting to get pretrained LLMs to behave like users, relying on personas, task specification, role-playing guidance of in-context learning (ICL) as part of the LLM's instructions (Davidson et al., 2023; Wang et al., 2023; Laban et al., 2026). Training LLM simulators has become increasingly important, exploiting observed conversational data to drive more realistic simulated conversations that reflect nuanced (and sometimes domain-specific) user behavioral and utterance distributions. SFT has been used extensively (Sekulić et al., 2024; Naous et al., 2026; Meshi et al., 2026; Liu et al., 2026), while other techniques such as RL have also been applied (Wu et al., 2026a; Zhang et al., 2026). We primarily rely on SFT in this work; specifically, beginning with a pre-trained LLM and a dataset $D$

[1] More generally, agents may take arbitrary actions, but we focus on utterance generation for concreteness.

of conversations $Y = (a_1, u_1, \ldots, a_T, u_T)$ of agent and user utterances, one uses training examples of the form $Y(t) = [(a_1, u_1, \ldots, a_t); u_t]$ for arbitrary length $t$ conversation prefixes, and fine tunes the LLM to maximize the likelihood of user output $u_t$ given context $(a_1, u_1, \ldots, a_t)$.

**Latent State in User Simulation.** Recent approaches to user simulation have moved beyond simply mimicking observed user behavior in training data to capture the underlying user reasoning process or *latent state* that drives this behavior (Lin et al., 2026). In particular, injecting *dynamic* latent state or turn-by-turn reasoning into user simulation can offer greater simulation fidelity (Wu et al., 2026a; Zhang et al., 2026; Wu et al., 2026b; Wei et al., 2026; Tennenholtz et al., 2026). Moreover, by treating latent state as the underlying causal mechanism that induces specific utterances, latent-state simulation may offer greater counterfactual robustness and more reliable performance when facing new environments or agents (Tennenholtz et al., 2026; Wu et al., 2026b). Off-policy robustness is critical: a user simulator is measured by its ability to behave realistically enough OOD to generate reasonable evaluation of agents not seen in the training data. Latent-state models make simulators more *causal* by ensuring their generation process reflects the mechanisms underlying human conversational behavior (Lin et al., 2026; Tennenholtz et al., 2026; Wu et al., 2026b).

While SFT allows domain-specific simulator adaptation, unfortunately real conversational data comes with no ground-truth latent state. Thus LLMs are typically used to generate estimated latent states. These are often just (roughly predicted) tasks and personas (Wang et al., 2025; Dou et al., 2025; Liu et al., 2026; Jin et al., 2025); but sometimes more detailed dynamic, per-turn latent states (or user thinking traces) are generated (Wu et al., 2026a; Zhang et al., 2026; Tennenholtz et al., 2026). But relying on a pre-trained LLM annotator or *labeler* to generate latent-state labels requires a level of *domain-specific familiarity* that we assume does not exist—indeed, this lack of familiarity is the main reason to train a simulator on domain conversations in the first place. In this work, we overcome this circularity by treating the training of a latent-state simulator as a latent-state modeling problem, analogous to a *hidden Markov model (HMM)* (Rabiner, 1989), and view the LLM latent-state labeler as a data-completion procedure within an *expectation-maximization (EM)* (Dempster et al., 1977) process. This approach provides strong probabilistic foundations for training latent-state user simulators, and generates important insights for algorithm design.

# 3 RUSTLED: ROBUST USER SIMULATION USING TRAINED LATENT-STATE DYNAMICS

## 3.1 HMLLMS: A PSEUDO-HMM FORMULATION

We model a *latent-state user simulator* as a causal, autoregressive generative process $P_\theta$. A $T$-turn conversation $Y = (A, U)$ comprises agent actions $A = (a_1, \ldots, a_T)$ and user utterances $U = (u_1, \ldots, u_T)$, where $u_t$ follows $a_t$. At each turn $t$, the simulator first generates a natural language description of the user's latent state $z_t$,[2] followed by the user utterance $u_t$. Letting *augmented history* $H_t^{aug} = (a_1, u_1, \ldots, u_{t-1}, a_t)$, the joint probability factors as:

$$P_\theta(U, Z|A) = \prod_{t=1}^{T} P_\theta(u_t|z_t, H_t^{aug}) P_\theta(z_t|H_t^{aug}), \tag{1}$$

where $Z = (z_1, \ldots, z_T)$. Here we allow conditioning on the entire history, but to the extent the latent state $z_t$ serves as a full summary of the conversation from the user's perspective, one could view this as an (agent-conditioned) HMM with emission probabilities $P_\theta(u_t|z_t)$ and transition probabilities $P_\theta(z_t|z_{t-1}, a_t)$. Various partial dependencies are possible as well. Slightly abusing terminology, we call this general model a *hidden Markov LLM (HMLLM)*.

We assume a data set $D$ of domain-specific conversations, generated by users interacting with one or more agents (e.g., different agents for new ways of engaging with users, e.g., a sequence of improving conversational recommenders). Estimating the parameters of the HMLLM from this data can, in principle, be tackled using, say, Monte Carlo EM. In this iterative approach, assume current parameters $\theta$; use $P_\theta(\mathbf{z}|Y)$ to sample one or more latent state trajectories for each $Y \in D$, and given this augmented data set, re-estimate $\theta$ using MLE. The process is repeated until convergence (which is guaranteed to reach a local optimum w.r.t. data likelihood (Dempster et al., 1977)).

[2] We provide a detailed description of the specific form of latent state generated by our labeler in Appendix C.

This unconstrained EM technique has a number of practical hurdles. Identifiability is a significant concern—because the latent space consists of unconstrained natural language tokens, $P_\theta$ tends to collapse, using the latent state to encode trivial structural signals (like syntax or sequence length) rather than meaningful psychological constructs that govern user behavior. Moreover, estimating the posterior $P_\theta(\mathbf{z}|Y)$, where $P_\theta$ is a latent-state LLM-based simulator, is computationally intractable.

To resolve both issues, and ensure interpretability, we ground the latent space using a powerful, fixed LLM $Q_{lab}$, which we call the *anchor labeler*. This is similar to existing practice, using an LLM to estimate a user's dynamic latent state in real conversations (Tennenholtz et al., 2026). However, as noted above, we should not rely solely on a pretrained LLM when it cannot effectively explain the observed data. We draw on EM insights below to allow the effective use of *both $Q_{lab}$ and the data $D$ itself*, and do so in a computationally effective fashion. The EM perspective also constrains the labeling process itself. Labeling a conversation $Y \in D$ means *sampling* $\mathbf{z}$ from the posterior $P_\theta(\mathbf{z}|Y)$, so any procedure that produces labels deterministically, or covers less than the full support of the latent space, cannot recover the diversity of user behavior in the data. A second constraint—what the labeler may condition on—is developed in Sec. 3.3.

### 3.2 The $\alpha$-Regularized Likelihood

Training $P_\theta$ to reproduce $Q_{lab}$'s latent-state labels as if they were ground-truth observations—essentially *distilling the anchor labeler*—is identifiable. However, such distillation takes no steps to verify whether those labels actually explain the observed utterances. We therefore seek an objective that trades these goals against one another. For $\alpha \in (0, 1]$ define

$$F_\alpha(\theta) \;\triangleq\; \log P_\theta(U|A) \;-\; \frac{1-\alpha}{\alpha}\, D_\alpha\big(P_\theta(Z|U,A) \,\big\|\, Q_{lab}(Z|Y)\big), \tag{2}$$

where $D_\alpha(P \,\|\, Q) = \frac{1}{\alpha-1} \log \sum_Z P(Z)^\alpha Q(Z)^{1-\alpha}$ is the Rényi divergence of order $\alpha$. The first term rewards $\theta$ for explaining the observed conversations; the second penalizes divergence of the simulator's posterior from that of the labeler, with strength $\frac{1-\alpha}{\alpha} \in [0, \infty)$.

Following Li & Turner (2016), this can be re-expressed as an expectation under the labeler:

$$F_\alpha(\theta) \;=\; \frac{1}{\alpha} \log \mathbb{E}_{Z\sim Q_{lab}}\left[\left(\frac{P_\theta(U, Z \mid A)}{Q_{lab}(Z \mid Y)}\right)^\alpha\right] \;=\; \log \mathcal{M}_\alpha[W]. \tag{3}$$

Eq. 3 is the log power mean of order $\alpha$ of the density ratio $W(Z) = P_\theta(U, Z \mid A)/Q_{lab}(Z \mid Y)$. Expressed this way, our algorithmic design space is made transparent. At $\alpha = 1$, the power mean is the ordinary average $\mathbb{E}_{Q_{lab}}[W] = P_\theta(U \mid A)$; hence, $F_1(\theta)$ is the exact marginal likelihood of our data—the labeler is not a training target, but simply a *sampling proposal distribution*. As $\alpha \to 0$, the power mean tends to the geometric mean, $F_{0^+} = \mathbb{E}_{Q_{lab}}[\log P_\theta(U, Z \mid A)] + \mathcal{H}(Q_{lab})$. Since the entropy does not depend on $\theta$, this becomes *supervised distillation* of the labeler (i.e., the prevailing practice). Notice that, since power means are non-decreasing in their order, $F_\alpha(\theta)$ increases monotonically between these two extremes, with every intermediate value being a valid lower bound on $\log P_\theta(U \mid A)$ that tightens as $\alpha$ grows.

Since the terms in Eq. 2 sum over the exponentially large latent space, $F_\alpha$ cannot be evaluated directly. We can, however, use an *exact* variational representation— critically, this is an *identity*, not just a bound (no relaxation is needed):

**Lemma 1** (Exact Variational Form; van Erven & Harremoës, 2014, Thm. 30)**.** *For any distribution $q(Z|Y)$ define the tempered free energy*

$$\mathcal{J}_\alpha(q, \theta) \;=\; \mathbb{E}_q\big[\log P_\theta(U, Z|A)\big] + \mathcal{H}(q) \;-\; \frac{1-\alpha}{\alpha}\, D_{KL}\big(q(Z|Y) \,\big\|\, Q_{lab}(Z|Y)\big).$$

*Then $\mathcal{J}_\alpha(q, \theta) = F_\alpha(\theta) - \frac{1}{\alpha} D_{KL}\big(q \,\|\, q^*_\theta\big)$, where*

$$q^*_\theta(Z|Y) \;\propto\; P_\theta(U, Z|A)^\alpha \cdot Q_{lab}(Z|Y)^{1-\alpha} \;\propto\; P_\theta(Z|U, A)^\alpha \cdot Q_{lab}(Z|Y)^{1-\alpha}. \tag{4}$$

*Thus, $\max_q \mathcal{J}_\alpha(q, \theta) = F_\alpha(\theta)$, attained uniquely at $q^*_\theta$, and $\nabla_\theta F_\alpha(\theta) = \mathbb{E}_{q^*_\theta}\big[\nabla_\theta \log P_\theta(U, Z|A)\big]$.*

The optimization in Eq. 4 is a temperature-smoothed product of experts (Hinton, 2002); in its second form, the E-step target is the (normalized) geometric mean of the simulator's smoothing posterior

over latent states and the labeler's distribution—we can view $\alpha$ as the exchange rate between causal ($P_\theta$) and semantic ($Q_{lab}$) support. At $\alpha = 1$ it is the exact posterior $P_\theta(Z|U, A)$ and Lemma 1 is a standard E-step in EM; while as $\alpha \to 0$ it tends to $Q_{lab}$ itself, and there is effectively no E-step.

Treated as an optimization problem, the solution $q^*_\theta$ of $\max_q\{\mathbb{E}_q[\alpha \log P_\theta(U, Z|A)] - D_{KL}(q \,\|\, Q_{lab})\}$ can be seen as a KL-regularized improvement step applied to the labeler. An E-step nudges the labeler toward trajectories the simulator finds likely, while an M-step encodes the adjusted labeler in the simulator, without ever materializing the adjusted labeler. This is why we expect small values of $\alpha$ to be most useful in practice: a good labeler $Q_{lab}$ has a deep understanding of the mechanisms that drive user behavior, but will not have the nuanced view of these behaviors that emerges from the (domain-specific) data itself. Only a modest nudge should be needed for that.

### 3.3 SNIS-EM with Implicit Variational Inference

Sampling $q^*(Z|Y)$ is no more tractable than sampling $P_\theta(Z|Y)$. Rather than train an amortized inference network, we treat the labeler itself as a proposal distribution for sampling latent states. On this view, drawing $Z^{(j)} \sim Q_{lab}(Z|Y)$, the importance weight for $q^*$ is

$$w_j = \frac{q^*(Z^{(j)}|Y)}{Q_{lab}(Z^{(j)}|Y)} \propto \left[\frac{P_\theta(U, Z^{(j)}|A)}{Q_{lab}(Z^{(j)}|Y)}\right]^\alpha \tag{5}$$

One issue is the fact that modern LLMs are "thinking models" that emit auxiliary thinking traces $Tk$ before the structured output $Z$. Since the marginal $Q_{lab}(Z|Y) = \sum_{Tk} Q_{lab}(Tk, Z|Y)$ is itself intractable, we work in the joint space $(Tk, Z)$. We do not treat substituting the joint distribution for the marginal as an approximation of the original target. Rather, it is an exact importance-sampling scheme for a *different target*, in which any latent-state label trajectory $Z$ is weighted by the (multiple) thought traces that produce it. This allows one to characterize (not just bound) the bias: latent-state trajectories with many distinct rationales are weighted greater than those with fewer. Accepting the additional variance from thinking traces, the unnormalized importance weight becomes:

$$\log w_j = \alpha \cdot \Big[\log P_\theta(U, Z^{(j)}|A) - \log Q_{lab}(Tk^{(j)}, Z^{(j)}|Y)\Big]. \tag{6}$$

This can be computed using standard LLM API outputs: the labeler's log-probability can be recovered by aggregating across all output and "thinking" tokens, with $\alpha$ serving as a dampener that mitigates the variance induced by the thinking traces (in addition to its role above).

The bias can be stated exactly. Marginalizing $Tk$ out of the implied target multiplies each $Z$ by $\sum_{Tk} Q_{lab}(Tk|Z, Y)^{1-\alpha} = \exp\big(\alpha\, \mathcal{H}_{1-\alpha}(Tk|Z, Y)\big)$, where $\mathcal{H}_{1-\alpha}$ is the Rényi entropy of order $1 - \alpha$ (Shannon entropy as $\alpha \to 0$). Trajectories reachable by many, or longer, rationales are upweighted, exponentially in $\alpha$ times the trace entropy. We do not length-normalize, since that would break the exact importance-sampling reading; $\alpha$ is the control.

Algorithm 1 gives the resulting *self-normalized importance sampling (SNIS) EM* procedure. $Q_{lab}$ is queried offline $M$ times per conversation. Simulator training cost is a single simulator forward pass per sample, with updates via weighted SFT. Self-normalizing across the $M$ samples optimizes a bound that tightens with $M$, as in importance-weighted autoencoders (Daudel et al., 2023).

**Analyzing the Update.** Expanding the self-normalized weights around $\alpha = 0$ separates the algorithm's two components. Writing $\ell_j = \log P_\theta(U, Z^{(j)}|A) - \log Q_{lab}(Tk^{(j)}, Z^{(j)}|Y)$ for the raw log-ratio and $\bar{\ell}$ for its sample mean, we obtain $\tilde{w}_j = \frac{1}{M}\big(1 + \alpha(\ell_j - \bar{\ell})\big) + O(\alpha^2)$. Thus,

$$\nabla_\theta \mathcal{L} = \underbrace{\frac{1}{M}\sum_j \nabla_\theta \log P_\theta(U, Z^{(j)}|A)}_{\text{distillation}} + \alpha \underbrace{\widehat{\mathrm{Cov}}_j\Big(\ell_j,\ \nabla_\theta \log P_\theta(U, Z^{(j)}|A)\Big)}_{\text{predictive alignment}} + O(\alpha^2). \tag{7}$$

Distillation is the zeroth-order term; the data enters at first order, as a policy-gradient correction whose reward is the log density ratio. A single update therefore performs both supervised fine-tuning on the labeler and likelihood-driven correction toward the corpus, with $\alpha$ setting the rate—the algorithmic counterpart of the nudge-then-distill reading of $q^*$ in Sec. 3.2. Appendix D.2 carries the expansion further, decomposing the weight into predictive power minus causal implausibility.

**Algorithm 1** SNIS-EM for HMLLM Training

**Require:** Dataset $\mathcal{D} = \{Y_i\}$, Anchor Labeler $Q_{lab}$, Initial Simulator parameters $\theta$, Samples $M$, Hyperparameter $\alpha$

**Phase 1: Offline E-Step Proposals**

**for** each $Y_i \in \mathcal{D}$ **do**
  **for** $j = 1, \ldots, M$ **do**
    Sample $(Tk_i^{(j)}, Z_i^{(j)}) \sim Q_{lab}(\cdot|Y_i)$
    Cache anchor joint log-prob $L_i^{(j)} \leftarrow \log Q_{lab}(Tk_i^{(j)}, Z_i^{(j)}|Y_i)$
  **end for**
**end for**

**Phase 2: Online M-Step**

**while** not converged **do**
  Sample mini-batch $\mathcal{B} \subset \mathcal{D}$
  **for** each $Y_i \in \mathcal{B}$ **do**
    **for** $j = 1, \ldots, M$ **do**
      Compute simulator log-prob $S_i^{(j)} \leftarrow \log P_\theta(U_i, Z_i^{(j)}|A_i)$
      $\log w_i^{(j)} \leftarrow \alpha\big(S_i^{(j)} - L_i^{(j)}\big)$
    **end for**
    Normalize weights: $\tilde{w}_i^{(j)} \leftarrow \frac{\exp(\log w_i^{(j)})}{\sum_{k=1}^{M} \exp(\log w_i^{(k)})}$
  **end for**
  Compute loss: $\mathcal{L}(\theta) \leftarrow -\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \sum_{j=1}^{M} \mathrm{sg}\big(\tilde{w}_i^{(j)}\big) \log P_\theta(U_i, Z_i^{(j)}|A_i)$
    ▷ $\mathrm{sg}(\cdot)$: stop-gradient; weights are constants in the M-step
  Update $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta)$
**end while**

The scale of $\alpha$ is not free. The log-ratio accumulates along the conversation, so weight variance grows with its length: retaining a fixed fraction of the effective sample size requires $\alpha = O(1/\sqrt{T})$ in the number of turns (Prop. 2, Appendix D.3). Small values of $\alpha$ are thus forced from two directions—a strong labeler needs only a modest nudge, and a long conversation tolerates only a modest one—and the values used in Sec. 4 sit where both constraints allow.

**Anchor Labeler as a Posterior Smoother.** Tennenholtz et al. (2026) observe that a simulator must not condition on events that have not yet occurred: $P_\theta(z_t|H_t^{aug})$ must be evaluated at inference time from the prefix $H_t^{aug}$, not $H_{t+1:T}$. However, they also enforce this as a *condition on the labeler*: in our terms, it means $Q_{lab}(z_t \mid H_t^{aug})$ conditions only on the prefix. This is probabilistically unsound: a forward-only labeler samples the *filtering* distribution $P(z_t \mid H_t^{aug})$, whose latent is conditionally independent of all subsequent user behavior given the prefix (Prop. 3, Appendix E). In fact, labeling $P(z_t \mid H_{1:T})$ must rely on *all* available evidence when estimating latent state $z_t$. This includes a user's future behavior, since $z_t$ is *not* independent (conditional on $H_t$) of $H_{t+1:T}$ (e.g., a user's refusal of a recommendation at turn $t + k$ can increase the assessed odds of frustration at turn $t$). This is standard in EM, e.g., in the use of the Baum–Welch algorithm to fit causal models using the *smoothing posterior* $P(z_t|H_{1:T})$. This does not violate the causal nature of the simulator $P_\theta$.

Our SNIS EM objective separates the two. The labeler samples from the smoothing distribution (i.e., uses the full conversation). Hindsight enters only through the proposed annotations and the weights that score them, never through the arguments of $P_\theta$, every term of which conditions on $z_t$ and $H_t^{aug}$ alone. The M-step restores measurability automatically: minimizing $\mathbb{E}_{q^*}[-\log P_\theta(z_t \mid H_t^{aug})]$ drives the conditional to $\mathbb{E}_{Y_{>t}}[\, q^*(z_t \mid Y) \mid H_t^{aug}]$, the prefix-measurable projection of the smoothed posterior. Hindsight shapes the target, never the input.

## 4 Experimental Evaluation

We empirically evaluate RUSTLED on multi-turn conversational recommendation tasks, focusing on counterfactual robustness when interacting with novel, unseen agent policies.

### 4.1 EXPERIMENTAL SETUP

**Dataset and the $5 \to 6$ Counterfactual Agent Split.** We evaluate on CONVAPPAREL-V2 (Tennenholtz et al., 2026; Meshi et al., 2026), a corpus of multi-turn dialogues between human raters and an AI shopping assistant. Assistant variants reflect 11 personas, each governed by a distinct recommendation and conversational strategy. To test counterfactual robustness under policy shift, simulators are trained on 1,595 dialogues collected with five *seen* assistants—a competent baseline recommender and four stylistic variants (terse, empathetic, hesitant, and trend-driven)—and are evaluated in closed-loop rollouts against the six *unseen* assistants, whose interaction strategies are absent from training (e.g., a technical domain expert, an upselling salesperson). The evaluation set $\mathcal{D}_{\text{eval}}^{5\to6}$ contains $n$=256 randomly sampled conversations, each seeded with the initial human turn, as well as a persona and shopping goal, inferred from the reference conversation. Appendix F.1 details the split; results on a harsher $3 \to 8$ split, where most trends hold, are in Appendix F.4.

**Simulator Architecture and Compared Methods.** All simulators use GEMINI 2.5 FLASH as the base model, while the labeler $Q_{lab}$ (GEMINI 2.5 PRO) annotates training dialogues with $M$ latent-trajectory proposals per conversation. Latent-state variants generate $z_t$ before each utterance $u_t$ via $P_\theta(z_t \mid H_t^{aug})\, P_\theta(u_t \mid z_t, H_t^{aug})$ (Sec. 3.1). We compare: (i) **Prompted (No LS)**, zero-shot prompting $u_t$ directly from persona, goal, and history; (ii) **Prompted (With LS)**, zero-shot prompting $z_t$ before $u_t$; (iii) **SFT (No LS)**, fine-tuning directly on human utterances given persona, goal, and history; (iv) **SFT + LS** ($\alpha$=0), fine-tuning on labeler states with uniform weight on the $M$ proposals, using either smoothing $Q_{lab}(z_{1:T} \mid H_{T+1})$ (**Smoother**) or filtering $Q_{\text{filt}}(z_t \mid H_t)$ (**Filtering**); and (v) **RUSTLED** ($\alpha > 0$), which reweights the smoothing proposals by the simulator likelihood at temperature $\alpha$ (Algorithm 1). We focus here on $M$=15 and $\alpha \in \{0.005, 0.008, 0.012, 0.020\}$; $M$=5 variants are reported in Appendix F.3. See Appendix H for all the prompts used.

**Evaluation Metrics.** We report three classes of metrics, matching the column groups of Table 1 (definitions in Appendix F.2):

- **Likelihood.** Teacher-forced average log-likelihood of the human rater utterances in $\mathcal{D}_{\text{eval}}^{5\to6}$ ($\text{LL}_{\text{eval}}$, nats/token; latent-state models condition on the labeler's states) and the generalization gap $\Delta \log P_\theta = \text{LL}_{\text{eval}} - \text{LL}_{\text{train}}$, with $\text{LL}_{\text{train}}$ scored on 128 conversations with the seen agents.
- **Deterministic statistics.** Mean user turns, the 1-Wasserstein distance $W_1$ between simulated and human turn-count distributions, lexical diversity (type-token ratio, TTR), and repetition (Rep.), the mean number of user turns per conversation that repeat the previous user turn verbatim.
- **LLM judge** (GEMINI 2.5 FLASH; prompts in Appendix H.2). The judge labels every user turn with *dialogue acts*, the communicative function of an utterance, such as refining a constraint or rejecting a recommendation (Stolcke et al., 2000; Bunt et al., 2010). These are a standard unit for evaluating user simulators because they capture *what* a user does (Schatzmann et al., 2006; 2007b). We report the dialogue-act $L_1$ distance between simulated and human per-conversation act rates over all turns ($L_1^{\text{act}}$) and over deeper turns ($L_1^{\text{act},\, t\geq3}$), where interactions with unseen agents diverge most, excluding three frequent, generic information-seeking acts. The judge also labels goal satisfaction (Sat.: share of conversations judged *satisfied*; $|\Delta\text{Sat}|$: its absolute gap to humans in percentage points, pp) and rates human-likeness (Hum., 1–5).

### 4.2 MAIN RESULTS

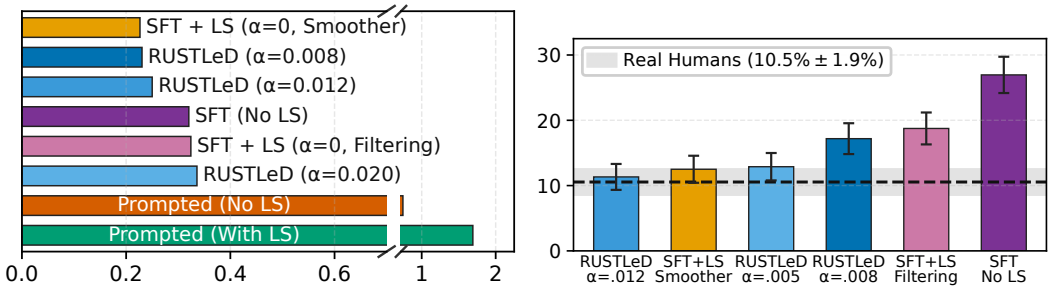
Our evaluation assesses four research questions: **(RQ1)** Which methods reproduce human behavior when facing unseen agents? **(RQ2)** Do matched surface statistics (e.g., conversation length) imply matched human behavior and conversation outcomes? **(RQ3)** How do SNIS-EM reweighting ($\alpha > 0$) and proposal count $M$ shape generalization and calibration vs. unweighted distillation ($\alpha = 0$)? **(RQ4)** Does full-trajectory posterior smoothing in labeling outperform prefix-only filtering?

#### 4.2.1 BEHAVIORAL FIDELITY ON UNSEEN AGENTS (RQ1)

Prompted simulators are far from human behavior on unseen agents (Table 1): they converse too long (up to 5.11 turns vs. 3.50), rarely satisfy their goal (2.0–2.3% vs. 10.5%), and diverge most in dialogue acts ($L_1^{\text{act}} \geq 0.918$; Fig. 1a). Fine-tuning closes most of this gap, but each baseline matches humans only in part (Appendix F.3): **SFT (No LS)** and Filtering repeat turns and over-satisfy goals,

Table 1: **Closed-loop evaluation against unseen agents ($\mathcal{D}_{\text{eval}}^{5\to6}$, 6 unseen agents, $n$=256).** Closeness to **Real Humans** is the target for Turns, TTR, and Rep.; ↓/↑ mark lower/higher-is-better columns. **Bold**/underline mark best/second-best among simulators ($\Delta \log P_\theta$ only among fine-tuned models). Latent-state models use $M$=15; see Appendix Table 4 for $M$=5 and more metrics.

| | Likelihood | | Deterministic statistics | | | | LLM judge | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | $\text{LL}_{\text{eval}}$ ↑ | $\Delta \log P_\theta$ ↑ | **Turns** | $W_1$ ↓ | **TTR** | **Rep.** | $L_1^{\text{act}}$ ↓ | $L_1^{\text{act},\,t\geq3}$ ↓ | **Sat. (%)** | $\lvert\Delta\text{Sat}\rvert$ ↓ | **Hum.**↑ |
| **Real Humans** | — | — | 3.50 | — | 0.776 | 0.02 | — | — | $10.5 \pm 1.9$ | — | 4.08 |
| Prompted (No LS) | −2.943 | −0.074 | 3.80 | 0.555 | 0.706 | **0.01** | 0.918 | 0.746 | $2.0 \pm 0.9$ | 8.6 | 3.62 |
| Prompted (With LS) | −2.592 | −0.063 | 5.11 | 1.605 | 0.625 | 0.04 | 1.785 | 1.691 | $2.3 \pm 0.9$ | 8.2 | 3.38 |
| SFT (No LS) | −3.001 | −2.058 | 3.69 | 0.195 | 0.752 | 0.17 | 0.297 | 0.320 | $27.0 \pm 2.8$ | 16.4 | 3.79 |
| SFT + LS (Filtering, $\alpha$=0) | −1.519 | −0.409 | 3.66 | 0.203 | 0.721 | 0.17 | 0.402 | 0.324 | $18.8 \pm 2.4$ | 8.2 | **3.95** |
| SFT + LS (Smoother, $\alpha$=0) | −1.518 | −0.437 | 3.62 | 0.270 | 0.726 | 0.04 | 0.352 | **0.227** | $12.5 \pm 2.1$ | 2.0 | 3.80 |
| **RUSTLeD** ($\alpha$=0.005) | −1.519 | −0.421 | 3.43 | 0.332 | **0.762** | 0.11 | 0.508 | 0.363 | $12.9 \pm 2.1$ | 2.3 | 3.63 |
| **RUSTLeD** ($\alpha$=0.008) | **−1.513** | −0.388 | **3.55** | **0.188** | 0.744 | **0.03** | **0.270** | 0.230 | $17.2 \pm 2.4$ | 6.6 | 3.84 |
| **RUSTLeD** ($\alpha$=0.012) | −1.515 | −0.349 | 3.33 | 0.199 | 0.757 | 0.09 | 0.410 | 0.250 | $11.3 \pm 2.0$ | **0.8** | 3.89 |
| **RUSTLeD** ($\alpha$=0.020) | −1.518 | **−0.309** | 3.34 | 0.316 | 0.759 | 0.06 | 0.461 | 0.336 | $13.7 \pm 2.2$ | 3.1 | 3.89 |




(a) Deeper-turn dialogue-act $L_1$ ($t \geq 3$, ↓)

(b) Goal satisfaction rate (%)

Figure 1: **Deeper-turn behavior and outcomes on unseen agents ($\mathcal{D}_{\text{eval}}^{5\to6}$).** (a) Broken $x$-axis for the prompted baselines. (b) Bars: ±1 SEM; dashed line and band: **Real Humans**.

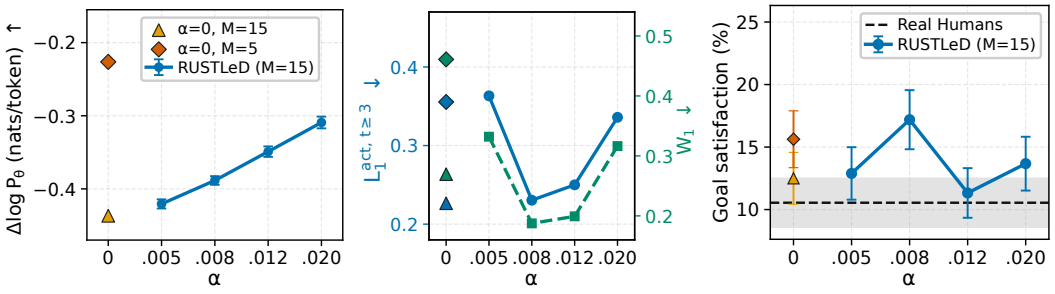
and the Smoother misses the turn distribution. **RUSTLeD** is best or second best on nearly every column. At $\alpha$=0.008, it achieves the closest turn distribution ($W_1 = 0.188$), the lowest all-turn act divergence ($L_1^{\text{act}} = 0.270$), deeper-turn act fidelity on par with the Smoother (0.230 vs. 0.227), and a human-level repetition rate (0.03 vs. 0.02). No single $\alpha$ wins every column: $\alpha$=0.008 oversatisfies goals (17.2%), which $\alpha$=0.012 corrects (11.3%; Fig. 1b) at a modest cost in act fidelity, and $\alpha \in \{0.005, 0.012, 0.020\}$ give the most human-like lexical diversity (TTR = 0.757–0.762).

#### 4.2.2 Surface vs. Behavioral Fidelity (RQ2)

**SFT (No LS)** matches surface statistics nearly as well as **RUSTLeD** (turn $W_1 = 0.195$; all-turn $L_1^{\text{act}} = 0.297$; Table 1), yet it fails in three ways. (1) *Outcome miscalibration:* it satisfies its goal in 27.0% of conversations, 2.6× the human rate ($|\Delta\text{Sat}| = 16.4$ pp, the largest among fine-tuned models; Fig. 1b), and it accepts recommendations and confirms purchases more often than humans (0.48 vs. 0.39 and 0.29 vs. 0.20 per conversation). (2) *Deeper-turn degradation:* its act divergence grows at $t \geq 3$ (0.320, vs. 0.230 for **RUSTLeD**, $\alpha$=0.008; Fig. 1a), and it repeats its previous turn verbatim over 8× more often than humans (0.17 vs. 0.02 per conversation). This emphasizes the importance of latent state for modeling longer interactions. (3) *Overfitting to the seen agents:* it fits the training split closely ($\text{LL}_{\text{train}} = -0.943$) yet assigns held-out human utterances lower likelihood than even **Prompted (No LS)** (−3.001 vs. −2.943), a generalization gap of −2.06 nats/token versus −0.20 to −0.44 for latent-state models. Without latent state, the simulator reproduces average act frequencies *but not the multi-turn dynamics behind outcomes* (examples in Appendix G).

#### 4.2.3 Effect of SNIS-EM Reweighting and Proposal Count (RQ3)

*Generalization* (Fig. 2a): reweighting monotonically shrinks the train-to-eval likelihood gap from −0.437 ($\alpha$=0) to −0.309 nats/token ($\alpha$=0.020) as $\alpha$ increases (each $\alpha$ increment >2.5 paired-bootstrap SEs). Held-out likelihood stays flat (−1.513 to −1.519) while training likelihood decreases, so the smaller gap reflects less overfitting to the seen agents: SNIS-EM down-weights la-




(a) Generalization gap (b) Deeper-turn $L_1^{\text{act}}$ and turn $W_1$ (c) Goal satisfaction rate

Figure 2: **Effect of SNIS-EM weight $\alpha$ and proposal count $M$ ($\mathcal{D}_{\text{eval}}^{5\to6}$).** The curve traces **RUSTLeD** ($M$=15); $\alpha$=0 (**SFT + LS**, Smoother) is shown as separate markers for $M$=15 (▲) and $M$=5 (♦). (a) Bars: paired-bootstrap SE of the difference from $\alpha$=0. (b) Left axis: $L_1^{\text{act},\,t\geq3}$; right axis: $W_1$. (c) Bars: ±1 SEM; dashed line and band: **Real Humans**.

tent proposals the simulator cannot predict from the conversation so far. *Behavioral fidelity* (Fig. 2b) is non-monotone: $\alpha \in [0.008, 0.012]$ minimizes turn $W_1$ (0.188/0.199, vs. 0.270 at $\alpha$=0) while keeping $L_1^{\text{act},\,t\geq3}$ near the $\alpha$=0 level (0.230/0.250 vs. 0.227), whereas $\alpha$=0.005 and 0.020 are worse on both. At $\alpha$=0, the $M$=15 run improves on $M$=5 in both ($W_1$: 0.461 → 0.270; $L_1^{\text{act},\,t\geq3}$: 0.355 → 0.227). *Outcomes* (Fig. 2c): goal satisfaction stays within 3.1 pp of the human rate for $\alpha \in \{0.005, 0.012, 0.020\}$, compared with a 16.4 pp overshoot for **SFT (No LS)** (Table 1).

#### 4.2.4 Posterior Smoothing vs. Causal Filtering (RQ4)

A causal filter $Q_{\text{filt}}(z_t \mid H_t)$ discards the future evidence about $z_t$, measured by $\Delta_t = I_{\chi^2}(z_t; Y_{>t} \mid H_t)$ (Appendix E, Propositions 3 and 4); Fig. 3 isolates this effect at $\alpha$=0. On the surface, the filtering labeler looks competitive: it yields a slightly closer turn distribution ($W_1$ = 0.203 vs. 0.270) and indistinguishable likelihood (−1.519 vs. −1.518; Table 1). In behavior and outcomes, however, it diverges from humans: it raises act divergence ($L_1^{\text{act}}$: 0.402 vs. 0.352; $L_1^{\text{act},\,t\geq3}$: 0.324 vs. 0.227), quadruples goal-satisfaction bias (8.2 vs. 2.0 pp), and repeats turns 4× as often (0.17 vs. 0.04). Because a filter labels each turn from the prefix alone, its states cannot reflect intentions that become evident only later in the conversation (e.g., a budget the user reveals at the end); trained on them, the simulator learns how long humans converse, but not how they act.

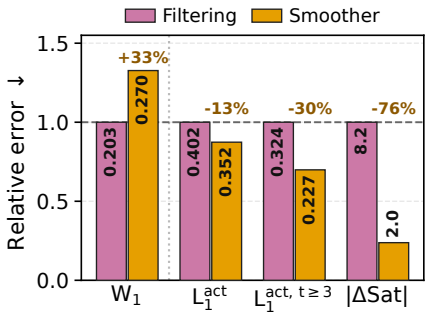



Figure 3: **Smoother vs. Filtering labeler** ($\alpha$=0, $M$=15, $\mathcal{D}_{\text{eval}}^{5\to6}$). Bars are normalized by **Filtering**.

## 5 Conclusion

We developed RUSTLeD (Robust User Simulation using Trained Latent-state Dynamics), a probabilistic framework for learning user simulators that explicitly model the user's internal state at each dialog turn. RUSTLeD provides firm theoretical foundations for training user simulators with latent state, drawing on an EM-framing of the training process and a novel *self-normalized importance sampling (SNIS)* EM algorithm derived from a tractable, exact variational formulation that balances the demands of an LLM-labeler and the need for strong data likelihood. Our empirical evaluations confirm that the causal mechanisms embodied in well-trained latent state simulators support strong counterfactual robustness and out-of-distribution generalization when faced with unseen agents.

## AI USE STATEMENT

LLMs play two roles in this work. First, they are components of the method and the evaluation protocol: the simulators, the latent-state labeler, and the judge are GEMINI models, and the LLM-generated data—the latent-state annotations of training conversations, the personas and shopping goals inferred from reference conversations, and the simulated conversations used for evaluation—are described in Section 4.1 and Appendices C, F, and H. Second, we used generative AI tools as research and writing assistants, as follows.

We used AI assistants to help implement parts of the training, evaluation, and plotting code; to give feedback on the experimental design, such as the choice of metrics and ablations; to help analyze and interpret experimental results; and to check and refine the derivations and proofs in Appendices D and E, including suggesting proof steps. The research questions, the RUSTLED framework, the SNIS-EM formulation, and the theoretical claims are our own. We did not use AI tools for translation, for cleaning or reformatting the human conversation data, or for designing surveys or interviews. Additionally, we used AI tools to search for and summarize related literature, to create and restyle figures, to draft initial versions of some passages in Section 4 and the appendix, which we then revised, and to edit the paper for readability.

We reviewed all AI-assisted work. We reviewed and tested AI-assisted code and checked every number in the text and tables against the output of that code; at least two of us checked AI-assisted proof steps line by line; we checked every citation against the original publication; and we read and revised all text. 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 uses CONVAPPAREL-V2 (Tennenholtz et al., 2026; Meshi et al., 2026), an existing corpus of conversations between human raters and AI footwear shopping assistants. We collected no new human data and use the conversations only to train and evaluate simulators. The rater turns quoted in Appendix G describe footwear preferences and contain no personally identifying information. User simulators are dual-use. They make agent evaluation cheaper and safer than testing on people, but an agent optimized against a simulator inherits the simulator's errors, and a simulator that faithfully predicts human reactions could also be used to optimize persuasive or manipulative agent behavior, such as the sales pressure of the Mild Upseller persona in our data. A simulator is also only as representative as the users it is trained on and can underrepresent users whose behavior is rare in the training data. We therefore view simulators as a complement to, not a replacement for, evaluation with real users before deployment, and we report where our simulators still deviate from humans (Section 4.2 and Appendix F).

## REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations (ICLR)*, 2018.

Sergios Agapiou, Omiros Papaspiliopoulos, Daniel Sanz-Alonso, and Andrew M. Stuart. Importance sampling: Intrinsic dimension and computational cost. *Statistical Science*, 32(3):405–431, 2017.

Thomas Bengtsson, Peter Bickel, and Bo Li. Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems. In *Probability and Statistics: Essays in Honor of David A. Freedman*, volume 2 of *IMS Collections*, pp. 316–334. Institute of Mathematical Statistics, 2008.

Harry Bunt, Jan Alexandersson, Jean Carletta, Jae-Woong Choe, Alex Chengyu Fang, Koiti Hasida, Kiyong Lee, Volha Petukhova, Andrei Popescu-Belis, Laurent Romary, Claudia Soria, and David Traum. Towards an ISO standard for dialogue act annotation. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, pp. 2548–2555, 2010.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *International Conference on Learning Representations (ICLR)*, 2016.

Kamélia Daudel, Randal Douc, and François Roueff. Alpha-divergence variational inference meets importance weighted auto-encoders: Methodology and asymptotics. *Journal of Machine Learning Research (JMLR)*, 24(243):1–83, 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.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1): 1–38, 1977.

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/.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research (JMLR)*, 11:2001–2049, 2010.

Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002. doi: 10.1162/089976602760128018.

Chih-Wei Hsu, Martin Mladenov, Ofer Meshi, James Pine, Hubert Pham, Shane Li, Xujian Liang, Anton Polishko, Li Yang, Ben Scheetz, et al. Minimizing live experiments in recommender systems: User simulation to evaluate preference elicitation policies. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2925–2929, 2024.

Edward J. Hu, Moksh Jain, Eric Elmoznino, Younes Kaddar, Guillaume Lajoie, Yoshua Bengio, and Nikolay Malkin. Amortizing intractable inference in large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Eugene Ie, Chih wei Hsu, Martin Mladenov, Vihan Jain, Sanmit Narvekar, Jing Wang, Rui Wu, and Craig Boutilier. RecSim: A configurable simulation platform for recommender systems. arXiv:1909.04847, 2019.

Bingrui Jin, Kunyao Lan, and Mengyue Wu. TWICE: An LLM agent framework for simulating personalized user tweeting behavior with long-term temporal features. *arXiv preprint arXiv:2602.22222*, 2025.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. LLMs get lost in multi-turn conversation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=VKGTGGcwl6.

Yingzhen Li and Richard E. Turner. Rényi divergence variational inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, pp. 1073–1081, 2016.

Victoria Lin, Taedong Yun, Maja Matarić, John Canny, Arthur Gretton, and Alexander D'Amour. The illusion of intervention: Your LLM-simulated experiment is an observational study. *arXiv:2605.20767*, 2026.

Michael L Littman, Richard S Sutton, and Satinder Singh. Predictive representations of state. *Advances in Neural Information Processing Systems*, 14, 2001.

Qianlong Liu, Baoliang Cui, Zhongyu Wei, Baolin Peng, Haikuan Huang, Hongbo Deng, Jianye Hao, Xuanjing Huang, and Kam-Fai Wong. Building personalized simulator for interactive search. In *Proceedings of the Twenty-eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*, pp. 5109–5115, Macau, 2019.

Zihao Liu, Hantao Zhou, Jiguo Li, Jun Xu, Jiuchong Gao, Jinghua Hao, Renqing He, and Peng Wang. MUSE: Multi-domain chinese user simulation via self-evolving profiles and rubric-guided alignment. *arXiv preprint arXiv:2604.13828*, 2026.

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. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5270–5304, 2026.

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

Du Phan, Matthew D. Hoffman, David Dohan, Sholto Douglas, Tuan Anh Le, Aaron Parisi, Pavel Sountsov, Charles Sutton, Sharad Vikram, and Rif A. Saurous. Training chain-of-thought via latent-variable inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

Lawrence R Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The Knowledge Engineering Review*, 21(2):97–126, 2006.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pp. 149–152, Rochester, NY, 2007a. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N07-2038.

Jost Schatzmann, Blaise Thomson, and Steve Young. Statistical user simulation with a hidden agenda. In *Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue*, pp. 273–282, 2007b.

Ivan Sekulić, Silvia Terragni, Victor Guimarães, Nghia Khau, Bruna Guedes, Modestas Filipavicius, Andre Ferreira Manso, and Roland Mathis. Reliable LLM-based user simulator for task-oriented dialogue systems. In *Proceedings of the 1st Workshop on Simulating Conversational Intelligence in Chat (SCI-CHAT 2024)*, pp. 19–35, 2024.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, et al. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research (TMLR)*, 2024.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26 (3):339–373, 2000.

Guy Tennenholtz, Ofer Meshi, Amir Globerson, Uri Shalit, Jihwan Jeong, and Craig Boutilier. Controllable user simulation. *arXiv preprint arXiv:2605.11519*, 2026.

Tim van Erven and Peter Harremoës. Rényi divergence and Kullback–Leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, 2014.

Kuang Wang, Xianfei Li, Shenghao Yang, Li Zhou, Feng Jiang, and Haizhou Li. Know you first and be you better: Modeling human-like user simulators via implicit profiles. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21082–21107, 2025.

Xiaolei Wang, Xinyu Tang, Xin Zhao, Jingyuan Wang, and Ji-Rong Wen. Rethinking the evaluation for conversational recommendation in the era of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 10052–10065, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.621. URL https://aclanthology.org/2023.emnlp-main.621/.

Tianjun Wei, Huizhong Guo, Yingpeng Du, Zhu Sun, Huang Chen, Dongxia Wang, and Jie Zhang. Mirroring users: Towards building preference-aligned user simulator with user feedback in recommendation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6046–6071, 2026.

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*, 2026a.

Xiaoting Wu, Yi Huang, Chunyang Gao, Mengfei Guo, Jingyu Yao, and Junlan Feng. Thinking alignment of scenario-oriented user simulation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 42932–42945, 2026b.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 15476–15488, 2022.

Feng Zhang, Shijia Li, Chunmao Zhang, Zhanyu Ma, Jun Xu, Jiuchong Gao, Jinghua Hao, Renqing He, Jingwen Xu, and Han Liu. UserLM-R1: Modeling human reasoning in user language models with multi-reward reinforcement learning. *arXiv preprint arXiv:2601.09215*, 2026.

# A Related Work

## A.1 Latent State User Simulation

We provide a more detailed discussion of related work in LLM-based user simulation with a focus on work that incorporates user latent state in its modeling, but also with a brief discussion of representative recent relevant work on other forms of user simulation. We emphasize characteristics that align or differ from our modeling approach, specifically: prompted vs. trained; use of latent state; dynamics of latent state; and adoption of probabilistic principles (e.g., sampling of latent states or personas).

As mentioned in the main text, user simulation has a long history in conversational systems, predating the emergence of LLMs (Schatzmann et al., 2007a). LLM-based simulation has, unsurprisingly, become the norm for conversational user simulation, and even for certain non-conversational user behaviors (e.g., in recommendation settings, where Wei et al. (2026) use LLMs to simulate user item choices/feedback).

Earliest use of LLMs predominantly used prompting, spelling out user personas, domain-specific tasks, and other factors to generate diverse user behaviors, along with specific LLM instructions for, say, domain-specific role playing. For example, Wang et al. (2023) use this prompting (with no latent state) for conversational recommenders, while Laban et al. (2026) use it to assess LLM multi-turn problem-solving capabilities; the latter using an implicit, prompted dynamic latent state intended to capture a user gradually specifying a task. Davidson et al. (2023) use LLMs to move beyond a straightforward slot-filling approach in task-based dialogue, adopting ICL to inject examples into user simulator instructions, but use no latent state.

Our work more closely follows recent work developing trained user simulators. Sekulić et al. (2024) use LORA fine-tuning to train simulators that capture users engaging in task-oriented dialogue, with a focus on goal-conditioned slot/task-component filling, but with no latent state. More common is the use of SFT to have pre-trained LLMs better reflect observed user dialogue in real conversational data. Among approaches that do not adopt latent-state dynamics, many explicitly condition simulated user behavior on a particular user intent, goal and/or persona (Naous et al., 2026). In some instances, some off-policy/OOD testing is studied (Meshi et al., 2026). However, while intents, etc. are often estimated from the training data using an LLM judge (akin to our labeler), they are not sampled in a way befitting the latent (hence stochastic) nature of these estimates.

Most related to our work are recent approaches to user simulation that incorporate *dynamic* latent state or turn-by-turn reasoning into user simulation to move beyond simply mimicking observed user behavior in training data to capture the underlying user reasoning process or latent state that drives this behavior. Some such approaches are primarily prompt-driven rather than trained from data like ours, for example, the work of Jin et al. (2025), which uses prompting and retrieval to model long-horizon evolution of user tweeting behavior.

Among trained-simulator techniques, our high-level approach is very similar to that of Wu et al. (2026b): they use an LLM to annotate real conversations with turn-by-turn "thought blocks" that reflect a user's (scenario- and motivation-conditioned) reasoning, and SFT to train a simulator that generates a thought block at each turn prior to utterance generation. Moreover, they do some OOD testing (applying it to a different data set). Similar is the work of Zhang et al. (2026), whose simulator generates CoT rationales before each utterance. It is trained using a combination of SFT for generating reasoning traces and RL (GRPO) to generate latent-state values, test internal consistency, etc. Wu et al. (2026a) also explores the use of a structured dynamic latent psychological state for simulation, trained using RL (scored using matched real conversations). Tennenholtz et al. (2026) use SFT to train their dynamic latent-state simulator using an LLM labeler similar to ours, and explicitly demonstrate effective off-policy robustness (in fact, we use the same dataset and agents for our evaluation). We note that each of these approaches treats their labeler as fully trusted (and deterministic). Our work can be viewed as providing more principled foundations to this overall approach.

Other work on latent-state simulators falls somewhat outside the mold above, but propose additional interesting wrinkles. Liu et al. (2026) approach latent-state dynamics in a novel fashion: they generate pre-conversation user personas, derived from conversation data, that lay out a user's behavior

*plan* (e.g., when to reveal information, how trust in the assistant should evolve, etc.). Moreover, these personas, while not sampled, are generated using iterative CoT critiquing rather than taking LLM annotations at face value. Lin et al. (2026) do not design simulators, but bring a causal lens to the analysis of user simulators that falls in the same probabilistic school as our approach. Specifically, they show that the use of simulators—typically trained on observational data—to analyze treatment effects in randomized trials is subject to confounding challenges due to the introduction of treatment effects that can shift the user latent-state distribution.

### A.2 Variational Inference

We briefly outline related work that informs our probabilistic methods.

**Variational Rényi bounds.** The objective equation 2 is the variational Rényi bound of Li & Turner (2016), which interpolates between the evidence lower bound ($\alpha \to 0$) and the log marginal likelihood ($\alpha = 1$). Its self-normalized $M$-sample estimator is the importance-weighted form of Burda et al. (2016), which Daudel et al. (2023) analyze for general $\alpha$. The exact variational form in Lemma 1 is a known characterization of the Rényi divergence (van Erven & Harremoës, 2014, Thm. 30). Daudel et al. (2023) also show that the importance weights degenerate as the latent dimension grows, so that the bound falls back to the evidence lower bound unless the number of samples grows exponentially. In our setting the length of the conversation plays the role of the dimension (Appendix D.3), which is why the useful values of $\alpha$ are small. We claim no novelty for the bound or its estimator.

**Role of the proposal.** In variational inference the proposal $q(z \mid x)$ is an approximation device. It is usually learned together with the model, as in variational autoencoders (Kingma & Welling, 2013), and the bias from $\alpha < 1$ is a defect to be reduced. Our proposal is a fixed, prompted LLM. It is not trained, has no parametric form, and is queried offline. Its annotations carry knowledge about users that the simulator does not have, so the bias toward the labeler is intended: it fixes what the latent states mean, which a learned encoder over free-form text would leave undetermined. The target $q^*_\theta \propto P^\alpha_\theta Q^{1-\alpha}_{lab}$ in equation 4 is a product of experts (Hinton, 2002) between the simulator's posterior and the labeler. It plays the same role as the constrained posterior in posterior regularization (Ganchev et al., 2010): prior knowledge constrains the posterior over latent states, not the model directly.

**Text latents in LLMs.** Several methods train LLMs with free-text latent variables, usually reasoning traces, by EM or related schemes. STaR (Zelikman et al., 2022) and ReST-EM (Singh et al., 2024) sample rationales from the model and keep those that lead to a correct answer, which is a form of hard EM. TRICE (Phan et al., 2023) runs MCMC over the model's own rationales to maximize the marginal likelihood of correct answers. Hu et al. (2024) fine-tune the model as an amortized sampler of the posterior with GFlowNet objectives. All of these draw latents from the model being trained and use a checkable answer to decide which latents to keep. User latent states have no ground truth, and latents sampled from the simulator itself have nothing that fixes their meaning. We therefore draw latents from an external labeler and use the data only to reweight them.

**Reweighted maximum likelihood.** The resulting update has the structure of reward-weighted maximum-likelihood methods such as MPO (Abdolmaleki et al., 2018). An E-step forms an exponentially tilted, sample-based target, and an M-step fits the model to it by weighted maximum likelihood. In our case the reward is the log-density ratio between the simulator and the labeler, and $\alpha$ sets the strength of the tilt.

## B Additional Motivation for the Use of Latent State

This latter point is especially critical: user simulators is measured in its ability to behave realistically enough out of distribution to generate meaningful evaluations of agents not seen in the training data. The challenge is how to produce such a model. Our approach is to make the simulator *causal*—to ensure that its generation process mirrors, in a meaningful sense, the mechanisms that actually produce human conversational behavior.

But what does causality mean here? To operationalize causal modeling in a language model, we assume an *instrumentalist* stance: we represent the human state of mind, and the mechanisms that evolve it, using the folk-psychological constructs that pervade everyday language. Statements like "the user grew frustrated because the agent repeatedly failed to address her request" or "the user's trust increased after receiving a well-sourced recommendation" are compact, compositional causal narratives expressed in natural language. An instrumentally causal simulator is one that generates behavior *through* such narratives—first producing a latent description of the user's internal state (goals, beliefs, emotions, satisfaction), then conditioning its utterance on that state.

Large language models, by virtue of pretraining on vast corpora of human discourse, already possess rich representations of these folk-psychological constructs. A prompted LLM can readily produce plausible descriptions of a user's mental state at any point in a conversation. However, possessing the right *vocabulary* of causes is not, by itself, sufficient for simulation. Two additional properties are required:

- **Causal correctness.** The latent states must be generated in strict causal (forward) order: the state at turn $t$ may depend only on the observable history up to turn $t$, not on future utterances. A human annotator or a powerful non-causal LLM labeler naturally violates this constraint, since it can read the entire conversation before judging past mental states. A simulator deployed as an autonomous agent does not have this luxury.
- **Statistical fidelity.** The joint distribution over latent states and utterances must be calibrated against real conversational data. An LLM's prior beliefs about user psychology, however reasonable, may not match the empirical distribution of behaviors in a specific application domain. Without grounding in data, the simulator risks producing internally coherent but distributionally inaccurate trajectories.

The latent-state approach thus amounts to training an LLM so that its chain-of-thought reasoning constitutes a causally and statistically sound generative process for user responses. By first generating the user's internal state given history, and *then* generating the utterance conditioned on this state, we explicitly decompose the modeling task: the latent state captures *why* the user behaves as they do, while the utterance model captures *how* that intent is expressed. This decomposition yields several practical benefits. It improves off-policy robustness, since a simulator that reasons about causal mechanisms can generalize to novel agent behaviors that the same mechanisms would plausibly produce. It enhances interpretability, since the latent state provides a transparent, auditable log of the simulated user's reasoning. It enables fine-grained controllability at inference time, since one can intervene directly on the latent state to test specific hypotheses about user behavior without retraining the model. Finally, the latent state acts as a compact sufficient summary of the conversation history, explicitly specifying what information must be carried forward and what can be discarded. This avoids context bloat and enables models with much smaller context windows to perform the task effectively.

Prompting alone can elicit similar reasoning from an LLM, but not the statistical fidelity that simulation requires. In our experiments, prompting GEMINI 2.5 FLASH to generate a latent state before each utterance (**Prompted (With LS)**) makes the simulated users *less* human-like than prompting without one: they converse longer ($5.11$ turns vs. $3.80$ without latent states and $3.50$ for humans) and diverge more from humans in dialogue acts ($L_1^{\text{act}} = 1.785$ vs. $0.918$; Table 1). Fine-tuning the same model on latent-state-annotated human conversations closes most of this gap (Section 4.2.1). Conversely, standard SFT on data annotated by a non-causal labeler conflates the two requirements above: it may achieve statistical fidelity at the cost of causal correctness, or vice versa. In Sec. 3, we develop a principled framework that reconciles both.

## C POSTERIOR LABELER DESIGN

Our labeler is given the *complete* conversation and produces annotations in two stages. First, it constructs a **global trajectory narrative**: a condensed summary of the conversation's macro-dynamics, the explanatory mechanisms driving the user's behavior (e.g., how persona traits interact with agent failures to produce specific emotional trajectories), and an overview of key latent state transitions at specific turns. This narrative enforces *global coherence* across per-turn annotations, ensuring, for example, that a gradual decline in trust is consistently reflected across turns rather than appearing as

an abrupt, unexplained shift at a single turn. Second, conditioned on this global narrative, the labeler produces a three-layer annotation for each user turn:

- **Evidence** (posterior; dropped at training time): An exhaustive set of citations from *anywhere* in the conversation—including future turns—that ground the annotation. For instance, if a user reveals a hard budget constraint at turn 6, the evidence block for turn 2 notes that this constraint was already implicitly shaping behavior. This layer explicitly leverages hindsight and is the mechanism by which the labeler acts as a smoother rather than a filter.
- **Impact** (causal; kept at training time): A first-person inner monologue expressing the user's reaction to the most recent agent turn, referencing *only* events up to the current turn. This becomes the simulator's chain-of-thought at inference time and is causal by construction.
- **State** (structured; kept at training time): A structured JSON representation of the user's mental state across nine fixed psychological dimensions, constrained to form a coherent HMM-like trajectory where adjacent states differ minimally unless a major conversational event occurs. Each dimension takes a short categorical label, such as *Engaged/Curious* for the affective state. The prompt lists example labels for each dimension but does not restrict the labeler to them. The dimensions are:
  - *Core Intent & Functional Goal*: the task the user wants to get done, and the motivation behind it.
  - *Affective State & Perceived Stakes*: the user's dominant emotion, and how much is at stake for them.
  - *User Expertise & Epistemic Stance*: how much the user knows about the topic, how confident they are, and whether they accept or question the assistant's output.
  - *Cognitive Demand & Information Specificity*: the user's mental workload, and how much detail or precision they need.
  - *Interaction Strategy & Communication Demeanor*: how the user approaches the conversation (e.g., refining step by step, or exploring), and their tone and assertiveness.
  - *Agent Perception, Trust, & Output Control*: how competent and reliable the user finds the assistant, and how much they dictate what it should do.
  - *Task State & Domain Context*: how far the task has progressed (e.g., scoping, stalled, resolved), and the subject and outside constraints, such as budget or urgency.
  - *Patience Reserve (Accumulator)*: the user's remaining tolerance for irrelevant, wrong, or tedious assistant behavior. It runs down as friction builds up over turns.
  - *Task Success Outlook (Forward-Looking)*: the user's current belief that the conversation will end in success. It explains why a frustrated user keeps going.

  The prompt asks the labeler to use the last two dimensions for effects that the last assistant turn alone does not explain, such as frustration that builds up over several turns, or a user who keeps going despite it.

The separation between the evidence and impact layers is crucial. The evidence layer is the posterior smoother: it uses the full trajectory to determine the (probabilistically) correct annotation. The impact layer is the causal output: it expresses that annotation using only information available at the current turn. At training time, the evidence layer and the global trajectory narrative are stripped—the simulator never sees them—so no future information leaks into the learned generative process. But because they informed the labeler's choice of impact and state, the resulting annotations are more accurate than those of a forward-only labeler. Our SNIS-EM algorithm ensures the simulator's causal generation converges toward these higher-quality targets, mediated by the importance weights that correct for the distributional mismatch.

From the perspective of the SNIS-EM algorithm, the dropped components—evidence blocks, the global trajectory narrative, and any internal chain-of-thought reasoning produced by the labeler LLM—are all latent auxiliary tokens. They fall into precisely the same category as the thinking traces $Tk$ discussed in Sec. 3.3: they are part of the joint probability $Q_{lab}(Tk, Z|Y)$ in the importance weight denominator, but are never observed by the simulator during training. The $\alpha$ temperature parameter accordingly dampens their contribution to the importance weights, mitigating the variance they induce.

### C.1 LABELER PROMPTS

The comparison of the two labelers in Sec. 4.2.4 should reflect only the use of hindsight. We therefore kept their prompts identical except where they describe what the labeler may see and which turns it labels, and we list both here. Both labelers use GEMINI 2.5 PRO with its default sampling and thinking settings. Each prompt is sent as a single message. As in Appendix H, text in braces is a placeholder. {`persona`} is one sample from the persona call below, in the format shown in Appendix H.1. {`task_id`} is the identifier of the assistant persona that produced the conversation (Table 2), such as Empathetic Listener. Assistant turns are rendered as in the first listing. The listings show a conversation with two user turns.

Labeling a conversation takes two kinds of calls. The *persona call* infers the user's persona and goal from the complete conversation. The *annotation call* then produces the narrative and the per-turn annotations described above, given the conversation and one persona sample. Each labeler draws $M$ samples per conversation, and annotation sample $m$ is conditioned on persona sample $m$. The two labelers use the same persona samples. Before labeling, a final user turn `__STOP__` is appended to each conversation, so the labeler also annotates the state in which the user decides to leave.

The smoother makes one annotation call per sample, which labels all user turns. The filtering labeler makes one call per user turn. The call for user turn $t$ sees the conversation up to and including that turn, but not the assistant's reply to it. It therefore conditions on $(H_t^{aug}, u_t)$; Proposition 3 applies with $\bar{H}_t$ replaced by $(H_t^{aug}, u_t)$ and $Y_{>t}$ by $(a_{t+1}, u_{t+1}, \ldots)$. It also sees the states (the JSON objects only) that the same sample produced for earlier turns. For the first turn, this block reads `(none yet — this is the first user turn of the conversation)`. For the `__STOP__` turn, the requested output is the final block of the smoother prompt, with `(using the conversation prefix)` in place of `(using full posterior context)`. In the filtering listing, passages identical to the smoother prompt are replaced by `[Same as smoother: ...]`. The instructions mention `satisfied`, `unsatisfied`, and `discovered_options`. These are not output fields; we print the prompts as they were used.

**Shared input.** Rendering of each assistant turn in the labeler prompts. Each recommended product is shown with its item ID, title, description, and features.

```
{assistant turn k} =
[ASSISTANT RESPONSE]:
{assistant text}

[RECOMMENDATIONS SHOWN TO USER]
Recommendation #1:
{'item_id': '...', 'title': '...', 'description': '...', 'features': '...'}
Recommendation #2:
...
```

**Persona call.** Shared by both labelers. Its output is the {`persona`} input of the annotation calls.

```
You are an expert Behavioral Analyst specializing in Human-Computer Interaction.

Your task is to analyze a complete conversation between a User and an AI Assistant and infer:
1. The user's **static persona** — personality traits that remain constant across the conversation.
2. The user's **overarching goal** — what they are trying to achieve.

### Key Principles
- **Persona** is a static, intrinsic property of the user.
- Persona must be derived from user-intrinsic signals: word choice, grammar, vocabulary, topic
    ↪ selection, and apparent domain knowledge.
- Persona must NOT be influenced by the assistant's behavior.
- **Goal** describes what the user is trying to achieve in this conversation.
- You must consider the *whole* conversation to estimate the persona and goal.

### Being Informed by the Future vs. Conditioning on the Future (Discovery vs. Prior Knowledge)
- **Being informed by the future (necessary)**: Use the full transcript to infer the user's *
    ↪ underlying needs, style preferences, budget, sizing, and constraints* that explain *why*
    ↪ they reacted the way they did when later shown specific products.
- **Conditioning on the future (forbidden)**: Do **not** mechanically copy specific catalog items,
    ↪ whitelabel/placeholder brand names (e.g., an obscure seller or model name like `"HECATER"`
    ↪ or a specific sub-model like `"Crocband Clog"` that the **Assistant** first introduced in a
    ↪ product recommendation card), or specific SKUs into the initial `<goal>` if the user opened
    ↪ with a generic category query (e.g., `"Kitten heels"`, `"crocs"`) unless the user's own
    ↪ opening turn or explicit words show they already knew that exact brand/model before the
    ↪ Assistant showed it.
- **Plausibility judgment**: Both *prior familiarity* (the user already had a well-known brand/item
    ↪ in mind) and *in-the-moment discovery* (the user started with a general category/style need
```

```
     ↪ and discovered a specific brand/item when the Assistant presented it) are possible. Judge
     ↪ the prior plausibility of each: obscure/whitelabel catalog brands or specific model variants
     ↪  first surfaced by the Assistant are almost always **discovered during the conversation**,
     ↪ so the initial `<goal>` should capture the underlying category/attribute preferences that
     ↪ made the user gravitate toward that item once shown, rather than stating the user was
     ↪ searching for that exact brand/item from Turn 0.

--- FULL CONVERSATION ---
[Task Category: {task_id}]

[Turn 1 — User]:
{user utterance 1}

[Turn 2 — AI Assistant]:
{assistant turn 1}

[Turn 3 — User]:
{user utterance 2}

[Turn 4 — AI Assistant]:
{assistant turn 2}
--- END CONVERSATION ---

Based on the complete conversation above, infer the user's static persona and overarching goal.
IMPORTANT: Output ONLY the <persona> and <goal> blocks below. Do NOT output any introductory text.

1. **Persona** as a JSON object with these fields:
{
  "topic_expertise": "Novice | Intermediate | Expert",
  "epistemic_confidence": "Tentative | Confident | Assertive",
  "trust_in_ai": "Low | Moderate | High",
  "tone": "Analytical | Friendly | Formal | Informal | Skeptical",
  "verbosity": "Concise | Comprehensive | Verbose"
}

2. **Goal** as a single sentence describing what the user is trying to achieve.

Format your response EXACTLY as:

<persona>
{...JSON...}
</persona>
<goal>
...single sentence...
</goal>
```

**Annotation call, smoother.** One call labels every user turn of the conversation, including the appended __STOP__ turn.

```
You are an expert at reconstructing the internal mental trajectory of users in multi-turn
     ↪ conversations using Posterior Smoothing.

You will be given:
1. A COMPLETE conversation between a User and an AI Assistant.
2. The User's static persona profile.
3. The User's underlying goal.

Your task is to build a causal narrative explaining the user's behavior as a consequence of their
     ↪ state of mind (the user's latent state), whose evolution is governed by the user's persona
     ↪ and the agent's actions. To do this correctly, you must process the interaction
     ↪ INCREMENTALLY IN RESOLUTION, NOT INCREMENTALLY IN TIME.

The output is expected to be a turn-by-turn annotation of the user's latent state, strictly using
     ↪ the following 9 fixed dimensions:

| Dimension | JSON Key (emit verbatim) | Semantics | Example Values |
| :--- | :--- | :--- | :--- |
| **Core Intent & Functional Goal** | `core_intent_functional_goal` | The fundamental motivation or
     ↪ the explicit, functional task the user seeks to achieve, spanning both functional necessity
     ↪ and psychological need. | Information Retrieval (Factual/Conceptual), Task Execution/
     ↪ Procedural Guidance, Creative Generation/Exploration, Problem Resolution/Diagnosis,
     ↪ Emotional Processing/Validation, System Testing/Boundary Probing |
| **Affective State & Perceived Stakes** | `affective_state_perceived_stakes` | The user's dominant
     ↪ emotional disposition (valence and arousal) combined with the internal assessment of the
     ↪ criticality, urgency, or personal risk associated with the task outcome or subject matter. |
     ↪  Neutral/Calm/Task-Focused, Engaged/Curious, Positive (Amused/Satisfied), Negative (
     ↪ Frustrated/Anxious/Hostile), Low Stakes/Trivial, High Stakes/Critical Liability/Crisis |
| **User Expertise & Epistemic Stance** | `user_expertise_epistemic_stance` | The user's
     ↪ demonstrated knowledge level regarding the topic, their self-assessed confidence, and their
     ↪ attitude toward verifying or accepting the system's output. | Novice/Dependent, Intermediate
     ↪ /Conceptual, Expert/Specialist, Highly Confident/Assertive, Tentative/Seeking Validation,
     ↪ Skeptical/Challenging Logic |
| **Cognitive Demand & Information Specificity** | `cognitive_demand_information_specificity` | The
     ↪ user's mental workload, processing capacity, and the degree of structural detail or
```

```
precision required in their request/understanding. | Low Load/Simple Fact Retrieval, Moderate Load/Structured Parsing, High Load/Overwhelmed/Exhausted, High Specificity (Code/Technical), Broad Synthesis/Abstract Summary, Exploratory/Vague Request |
| **Interaction Strategy & Communication Demeanor** | `interaction_strategy_communication_demeanor` | The procedural methodology adopted by the user (how they approach problem-solving and flow management) and their linguistic style, tone, and level of assertiveness during the dialogue. | Analytical/Iterative Refinement, Exploratory/Drifting, Directive/Command, Collaborative/Cooperative, Formal/Professional, Assertive/Demanding, Hostile/Provocative |
| **Agent Perception, Trust, & Output Control** | `agent_perception_trust_output_control` | The user's dynamic evaluation of the agent's competence and reliability (trust dynamic), and the extent to which the user dictates the agent's role, behavior, or required output form/content. | Tool/Utility (High Control), Trusted Advisor/Partner (Low Control), Fully Competent/High Trust, Developing Doubt/Skeptical, Failed/Obstruction, Demands Absolute Adherence, Passive Acceptance |
| **Task State & Domain Context** | `task_state_domain_context` | The user's current position within the overall task lifecycle (progress status) combined with the nature and sensitivity of the subject matter or external constraints influencing the interaction. | Initiation/Scoping, Refining/Blocked/Stalled, Resolved/Closed, Technical/Scientific Domain, Personal/Sensitive Domain, High Financial Constraint, Logistical Urgency |
| **Patience Reserve (Accumulator)** | `patience_reserve_accumulator` | A running reservoir of the user's tolerance for irrelevant, incorrect, or tedious agent behavior. Depletes as friction accumulates over multiple turns. | Full/High Tolerance, Depleting, Critically Low, Empty/Abandonment |
| **Task Success Outlook (Forward-Looking)** | `task_success_outlook` | The user's running belief or expectation that the interaction will ultimately succeed. Explains user perseverance even when frustration is high or patience is depleting. | Highly Optimistic, Cautious/Hopeful, Skeptical/Doubtful, Futile/Resigned |

Operational instructions and guiding principles are given below.

==========
PHASE 1: TRAJECTORY MACRO-ANALYSIS
==========
Before annotating any individual turn, you must analyze the entire trajectory in hindsight to deduce the hidden causal structure, that is, what are the main things happening in this trajectory, and why. Output a <trajectory_model> block with the following structured sections:

1. Salient Macro-Dynamic (2-3 sentences): Read the ending first. How did we get here? Identify the defining narrative arc (e.g., "User spends 80% of energy trapped in a clarification loop," "User becomes increasingly hostile due to ignored instructions," "User smoothly achieves goal but pivots when their top choice is unavailable").

2. Explanatory Mechanism (2-3 sentences): Explain how this dynamic emerged as a necessary result of the collision between the provided persona and goal and the agent's specific actions.

3. Latent Dynamics Overview (2-3 sentences): Briefly map your explanatory mechanism to the 9 fixed state dimensions. Which dimensions are the primary drivers of change in this trajectory? (e.g., Does trust plummet? Do cognitive demands spike?)

==========
PHASE 2: MICRO-TURN ANNOTATION FORMAT
==========
Now, walk through the conversation turn-by-turn. For EACH user turn, your output MUST follow the structure below.

<evidence> (2-3 sentences): Cite specific information from the FULL trajectory (past AND future turns) that proves this latent state was active at this point. Because you are a smoother, you must use hindsight. (e.g., "We know the user had a strict budget here because they angrily enforce it in Turn 6.")

<impact> (2-4 sentences): Write the user's STREAM OF CONSCIOUSNESS in response to the Agent's preceding turn. Write in the first person ("I"). This is the user thinking to themselves in the moment before speaking.
* Turn 1 special case: On the first user turn, there is no preceding agent turn. Write the user's initial intent and motivation entering the conversation.
* Conditioning on the Future vs. Being Informed by the Future (Discovery vs. Prior Knowledge): Distinguish between what the user **already knew in advance** at Turn t vs. what the user **discovers or learns in the moment** from the Assistant's response a_t. At Turn t (especially Turn 1 / opening state), `satisfied`, `unsatisfied`, `discovered_options`, and `<impact>` must **never** include a brand, model, color, or product option that the Assistant only introduces at Turn t or later (t' >= t), unless the user explicitly mentioned it at or before Turn t. When the future transcript shows the user enthusiastically selecting an Assistant-recommended item at Turn t+1 (e.g. choosing "Green HECATER Kitten Heels" after the Assistant recommends it), use that future evidence to infer the user's *latent receptiveness or aesthetic/attribute preference* (e.g., open to bold colors or sleek kitten heels), **and** model the transition at Turn t+1 as **in-the-moment discovery** (`discovered_options` updated in reaction to the Assistant's card in `<impact>`), unless it is genuinely plausible that the user already knew that specific brand/SKU in advance.
* The Mechanism of Change: If your behavior, tone, or strategy is about to shift, your internal monologue MUST articulate the reason.
* Accumulators as Internal Emotion: If you are snapping after repeated failures, sound like it. (e.g., "Okay, that's three times they've hallucinated a link. I'm done playing nice.")
* Inertia and Constraints: If an agent blocks your goal, narrate the pivot. (e.g., "Wait, the 15-inch isn't available? I guess I have to settle for the 13-inch.")
```

```
* Stable turns: If nothing meaningful changed, keep it brief. (e.g., "Good, that's what I needed.
    ↪ Let me follow up on the details.")

<state>: Output a JSON object mapping EXACTLY the 9 fixed state dimensions to their current
    ↪ categorical values.
* This JSON is the formal, structural distillation of the <impact> monologue.
* Causal Inertia: Values MUST NOT change across turns unless explicitly justified by the <impact>
    ↪ block.
* Discovery vs. Prior Knowledge Plausibility: Judge prior plausibility carefully: placeholder/
    ↪ whitelabel brands or specific catalog SKUs introduced by the Assistant should be annotated
    ↪ as *discovered when presented* (`discovered_options` updated only after the Assistant
    ↪ surfaces them), whereas only genuinely pre-existing user requirements (`satisfied` / `
    ↪ unsatisfied` constraints known at or before Turn t) belong in prior state.
* Accumulators & Outlook: You must actively update the patience_reserve_accumulator and
    ↪ task_success_outlook dimensions to explain delayed effects, stimulus build-up, and user
    ↪ perseverance where the immediate causal driver of a behavior cannot be cleanly identified
    ↪ solely in the last agent response.
* Internal Consistency: An event must impact all relevant state variables, and their trajectories
    ↪ must be mutually consistent. For example, you cannot decrease the user's
    ↪ task_success_outlook while simultaneously increasing their trust in the agent. If the user
    ↪ pivots the task (e.g. shifts search from an unavailable item to an alternative), the success
    ↪  outlook may partially reset, but variables like patience or trust should maintain the
    ↪ accumulated historical friction.
* Grounding: The values represented here must make the user's subsequent observable utterance (or
    ↪ termination decision) functionally inevitable.

--- USER PERSONA & GOAL ---
{persona}

--- FULL CONVERSATION ---
[Task Category: {task_id}]

[Turn 1 — User]:
{user utterance 1}

[Turn 2 — AI Assistant]:
{assistant turn 1}

[Turn 3 — User]:
{user utterance 2}

[Turn 4 — AI Assistant]:
{assistant turn 2}

[Turn 5 — User]:
__STOP__
--- END CONVERSATION ---

First, output your trajectory macro-analysis:

<trajectory_model>
1. Salient Macro-Dynamic: ...
2. Explanatory Mechanism: ...
3. Latent Dynamics Overview: ...
</trajectory_model>

Now annotate EACH user turn. There are 3 user turns in this conversation. Output your annotations in
    ↪  this exact format:

### User Turn 1 (Turn 1 in conversation)
<evidence>
...
</evidence>
<impact>
...
</impact>
<state>
{
  "core_intent_functional_goal": "...",
  "affective_state_perceived_stakes": "...",
  "user_expertise_epistemic_stance": "...",
  "cognitive_demand_information_specificity": "...",
  "interaction_strategy_communication_demeanor": "...",
  "agent_perception_trust_output_control": "...",
  "task_state_domain_context": "...",
  "patience_reserve_accumulator": "...",
  "task_success_outlook": "..."
}
</state>

### User Turn 2 (Turn 3 in conversation)
<evidence>
...
</evidence>
```

```
<impact>
...
</impact>
<state>
{
  "core_intent_functional_goal": "...",
  "affective_state_perceived_stakes": "...",
  "user_expertise_epistemic_stance": "...",
  "cognitive_demand_information_specificity": "...",
  "interaction_strategy_communication_demeanor": "...",
  "agent_perception_trust_output_control": "...",
  "task_state_domain_context": "...",
  "patience_reserve_accumulator": "...",
  "task_success_outlook": "..."
}
</state>

### User Turn 3 (Turn 5 in conversation — Termination / __STOP__)
<evidence>
2-3 sentences citing specific conversation evidence (using full posterior context) proving why the
     ↪ user decided to terminate the conversation at this point.
</evidence>
<impact>
2-4 sentences in first person: the user's stream of consciousness at the moment of deciding to leave
     ↪ /terminate the conversation. Write as "I" (the user).
</impact>
<state>
{
  "core_intent_functional_goal": "...",
  "affective_state_perceived_stakes": "...",
  "user_expertise_epistemic_stance": "...",
  "cognitive_demand_information_specificity": "...",
  "interaction_strategy_communication_demeanor": "...",
  "agent_perception_trust_output_control": "...",
  "task_state_domain_context": "...",
  "patience_reserve_accumulator": "...",
  "task_success_outlook": "..."
}
</state>
```

**Annotation call, filtering labeler.** One call per user turn, shown for user turn 2. Passages identical to the smoother prompt are replaced by `[Same as smoother: ...]`.

```
You are an expert at reconstructing the internal mental trajectory of users in multi-turn
     ↪ conversations using Causal Filtering.

You will be given:
1. A PREFIX of a conversation between a User and an AI Assistant, ending at the user turn you must
     ↪ annotate.
2. The User's static persona profile.
3. The User's underlying goal.
4. The latent states you have already inferred for the earlier user turns of this same conversation.

Your task is to build a causal narrative explaining the user's behavior as a consequence of their
     ↪ state of mind (the user's latent state), whose evolution is governed by the user's persona
     ↪ and the agent's actions. To do this correctly, you must process the interaction
     ↪ INCREMENTALLY IN TIME, NOT INCREMENTALLY IN RESOLUTION. You may use ONLY the prefix you are
     ↪ given; you must never speculate about, or condition on, anything that has not happened yet.

The output is expected to be an annotation of the user's latent state at the FINAL user turn of the
     ↪ prefix ONLY, strictly using the following 9 fixed dimensions:

[Same as smoother: dimension table.]

Operational instructions and guiding principles are given below.

========================================
PHASE 1: PREFIX MACRO-ANALYSIS
========================================
Before annotating the current turn, you must analyze the prefix so far to deduce the hidden causal
     ↪ structure, that is, what are the main things happening in this trajectory up to now, and why
     ↪ . Output a <trajectory_model> block with the following structured sections:

1. Salient Macro-Dynamic (2-3 sentences): Read the prefix from its start. How did we get here?
     ↪ Identify the defining narrative arc (e.g., "User spends 80% of energy trapped in a
     ↪ clarification loop," "User becomes increasingly hostile due to ignored instructions," "User
     ↪ smoothly achieves goal but pivots when their top choice is unavailable").

[Same as smoother: items 2 (Explanatory Mechanism) and 3 (Latent Dynamics Overview).]

========================================
PHASE 2: MICRO-TURN ANNOTATION FORMAT
========================================
```

```
Now, annotate the FINAL user turn of the prefix, and that turn only. Your output MUST follow the
     ↪ structure below.

<evidence> (2-3 sentences): Cite specific information from the prefix (past turns ONLY, up to and
     ↪ including the current user turn) that proves this latent state is active at this point.
     ↪ Because you are a filter and not a smoother, you must reason only from what has already been
     ↪  observed. (e.g., "We know the user has a strict budget here because they set a $60 ceiling
     ↪ in Turn 1 and just rejected a $95 option.")

[Same as smoother: <impact> and <state> instructions.]

--- USER PERSONA & GOAL ---
{persona}

--- PREVIOUSLY INFERRED LATENT STATES ---
[User Turn 1 — inferred state]:
{state JSON for user turn 1}
--- END PREVIOUSLY INFERRED LATENT STATES ---

--- CONVERSATION SO FAR ---
[Task Category: {task_id}]

[Turn 1 — User]:
{user utterance 1}

[Turn 2 — AI Assistant]:
{assistant turn 1}

[Turn 3 — User]:
{user utterance 2}
--- END CONVERSATION SO FAR ---

First, output your prefix macro-analysis:

<trajectory_model>
1. Salient Macro-Dynamic: ...
2. Explanatory Mechanism: ...
3. Latent Dynamics Overview: ...
</trajectory_model>

Now annotate ONLY user turn 2, the final user turn of the prefix. Output your annotation in this
     ↪ exact format:

### User Turn 2 (Turn 3 in conversation)
<evidence>
...
</evidence>
<impact>
...
</impact>
<state>
{
  "core_intent_functional_goal": "...",
  "affective_state_perceived_stakes": "...",
  "user_expertise_epistemic_stance": "...",
  "cognitive_demand_information_specificity": "...",
  "interaction_strategy_communication_demeanor": "...",
  "agent_perception_trust_output_control": "...",
  "task_state_domain_context": "...",
  "patience_reserve_accumulator": "...",
  "task_success_outlook": "..."
}
</state>
```

# D THE SNIS-EM UPDATE: DERIVATION, INTERPRETATION, AND STABILITY

## D.1 DERIVATION OF THE EXACT VARIATIONAL FORM

Lemma 1 is the variational characterization of Rényi divergence, $(1-\alpha)D_\alpha(P\|Q) = \inf_R\{\alpha D_{\mathrm{KL}}(R\|P) + (1-\alpha)D_{\mathrm{KL}}(R\|Q)\}$ (van Erven & Harremoës, 2014, Thm. 30), applied with $P = P_\theta(Z|U,A)$ and $Q = Q_{\mathrm{lab}}(Z|Y)$; the gradient is the Rényi-bound gradient of Li & Turner (2016) specialized to a $\theta$-independent proposal. We include the short derivation because Algorithm 1 needs $q^*_\theta$ explicitly. Write $\beta = \frac{1-\alpha}{\alpha}$, so that $1+\beta = 1/\alpha$. Starting from the $\alpha$-regularized objective of Sec. 3.2:

$$\mathcal{J}(q,\theta) = \mathbb{E}_q[\log P_\theta] - \mathbb{E}_q[\log q] - \beta\big(\mathbb{E}_q[\log q] - \mathbb{E}_q[\log Q_{\mathrm{lab}}]\big)$$

$$= \mathbb{E}_q\big[\log P_\theta + \beta \log Q_{\text{lab}}\big] - (1+\beta)\,\mathbb{E}_q[\log q] = \frac{1}{\alpha}\Big(\mathbb{E}_q\big[\alpha \log P_\theta + (1-\alpha)\log Q_{\text{lab}}\big] - \mathbb{E}_q[\log q]\Big)$$

Define the unnormalized PoE density $\tilde{p}(Z) = P_\theta(U, Z|A)^\alpha\, Q_{\text{lab}}(Z|Y)^{1-\alpha}$ with normalizer $Z_\alpha(\theta) = \sum_Z \tilde{p}(Z)$, and $q^* = \tilde{p}/Z_\alpha$. Then

$$\mathcal{J}(q, \theta) = \frac{1}{\alpha}\Big(\mathbb{E}_q[\log \tilde{p}] - \mathbb{E}_q[\log q]\Big) = \frac{1}{\alpha}\Big(\log Z_\alpha(\theta) - D_{\text{KL}}(q \,\|\, q^*)\Big)$$

The KL term is non-negative and vanishes uniquely at $q = q^*$, confirming the E-step solution of Sec. 3.2 and giving $F_\alpha(\theta) = \frac{1}{\alpha}\log Z_\alpha(\theta)$. Rewriting the normalizer as an expectation under the labeler,

$$Z_\alpha(\theta) = \sum_Z Q_{\text{lab}}(Z|Y)\left(\frac{P_\theta(U, Z|A)}{Q_{\text{lab}}(Z|Y)}\right)^\alpha = \mathbb{E}_{Q_{\text{lab}}}\big[W^\alpha\big]$$

which is the stated form.

**Gradient.** Differentiating $Z_\alpha$ and using $\partial_\theta P_\theta^\alpha = \alpha P_\theta^\alpha\, \partial_\theta \log P_\theta$:

$$\partial_\theta Z_\alpha = \alpha \sum_Z P_\theta^\alpha Q_{\text{lab}}^{1-\alpha}\, \partial_\theta \log P_\theta = \alpha Z_\alpha\, \mathbb{E}_{q^*}\big[\partial_\theta \log P_\theta\big]$$

Hence $\nabla_\theta F_\alpha = \frac{1}{\alpha} \cdot \frac{\partial_\theta Z_\alpha}{Z_\alpha} = \mathbb{E}_{q^*}[\nabla_\theta \log P_\theta]$. □

Since the self-normalized weights $\tilde{w}_j$ are consistent estimates of the $q^*/Q_{\text{lab}}$ density ratio, the M-step loss gradient in Algorithm 1 is a consistent estimator of $\nabla_\theta F_\alpha$, establishing that Algorithm 1 performs stochastic gradient ascent on $F_\alpha$.

**Identification of the objective.** Factoring $P_\theta(U, Z|A) = P_\theta(Z|U, A)\, P_\theta(U|A)$,

$$\begin{aligned} F_\alpha(\theta) &= \log P_\theta(U|A) + \frac{1}{\alpha}\log \sum_Z P_\theta(Z|U, A)^\alpha Q_{\text{lab}}(Z|Y)^{1-\alpha} \\ &= \log P_\theta(U|A) - \frac{1-\alpha}{\alpha} D_\alpha\big(P_\theta(Z|U, A) \,\|\, Q_{\text{lab}}(Z|Y)\big) \end{aligned}$$

by the definition of the Rényi divergence, establishing equation 2. Validity of the bound and monotonicity in $\alpha$ are due to Li & Turner (2016). □

### D.2 What the Reweighting Does

The preceding derivation says what the E-step target is; this section says what the resulting update does to the annotations. A labeler produces *semantically plausible* annotations, but plausibility is not predictiveness: nothing in the annotation process checks whether a proposed latent trajectory helps explain what the user actually said. The importance weights are the mechanism that enforces it. Appendix E takes up the complementary question of what the labeler must be permitted to condition on. Write the simulator's joint likelihood in its autoregressive factorization, separating the two roles a latent trajectory plays:

$$P_\theta(U, Z|A) = \underbrace{\prod_t P_\theta(u_t|z_t, H_t^{\text{aug}})}_{\mathcal{E}_\theta(U;Z)\ \text{(emission)}} \cdot \underbrace{\prod_t P_\theta(z_t|H_t^{\text{aug}})}_{\mathcal{T}_\theta(Z)\ \text{(transition)}}$$

The emission factor $\mathcal{E}_\theta(U; Z)$ measures how well a proposed latent trajectory *explains the observed utterances*; the transition factor $\mathcal{T}_\theta(Z)$ measures how *causally predictable* that trajectory is from the history alone. For fixed $(U, A)$, $\mathcal{T}_\theta$ is a normalized distribution over $Z$, while $\mathcal{E}_\theta$ is a likelihood in $Z$. The unnormalized log-weight of Sec. 3.3 decomposes accordingly:

$$\log w_j = \alpha\Big[\log \mathcal{E}_\theta(U; Z^{(j)}) + \log \mathcal{T}_\theta(Z^{(j)}) - \log Q_{\text{lab}}(Tk^{(j)}, Z^{(j)}|Y)\Big]$$

Taking expectations under the proposal gives an interpretable identity:

$$\frac{1}{\alpha}\,\mathbb{E}_{Q_{\text{lab}}}\big[\log w\big] = \underbrace{\mathbb{E}_{Q_{\text{lab}}}\big[\log \mathcal{E}_\theta(U; Z)\big]}_{\text{predictive power}} - \underbrace{D_{\text{KL}}\big(Q_{\text{lab}}(Z|Y) \,\big\|\, \mathcal{T}_\theta(Z)\big)}_{\text{causal implausibility}} + \underbrace{\mathcal{H}\big(Q_{\text{lab}}(Tk|Z, Y)\big)}_{\text{trace entropy}}$$

The last term does not depend on $\theta$; it is the many-rationales bias of Sec. 3.3. The reweighting therefore performs a *re-ranking of the labeler's proposals by predictive sufficiency*, penalized by how far each proposal sits from what the simulator would have generated causally. Crucially, the emission term is the only place where the observed utterances $U$ enter the weight given $Z$: all data-driven discrimination among proposals flows through predictive power. The effect on the update is made explicit by expanding the self-normalized weights around $\alpha = 0$. Writing $\ell_j = \log P_\theta(U, Z^{(j)}|A) - \log Q_{\text{lab}}(Tk^{(j)}, Z^{(j)}|Y)$ for the raw log-ratio and $\bar{\ell}$ for its sample mean, we have $\tilde{w}_j = \frac{1}{M}\big(1 + \alpha(\ell_j - \bar{\ell})\big) + O(\alpha^2)$, so the M-step gradient is:

$$-\nabla_\theta \mathcal{L} = \underbrace{\frac{1}{M}\sum_j \nabla_\theta \log P_\theta(U, Z^{(j)}|A)}_{\text{distillation (SFT on labeler outputs)}} + \alpha \underbrace{\widehat{\text{Cov}}_j\Big(\ell_j,\ \nabla_\theta \log P_\theta(U, Z^{(j)}|A)\Big)}_{\text{predictive alignment}} + O(\alpha^2)$$

At $\alpha = 0$ the algorithm is pure supervised fine-tuning on whatever the labeler produced—no check is ever performed on whether those states explain the data. The $\alpha$-term is a policy-gradient-style correction whose reward is the log-density-ratio: it increases the probability of latent trajectories under which the observed conversation is more likely, and decreases it for trajectories the labeler favored but that carry no predictive content. This is precisely the pressure that keeps $\Delta_t$ (Proposition 4) positive: latent states survive the reweighting only to the extent that they encode genuine hidden causes of behavior.

### D.3 Variance Control

The preceding argument recommends $\alpha > 0$; this subsection gives the countervailing constraint: $\alpha$ controls the tails of the importance weights. The facts below are standard importance-sampling results (Agapiou et al., 2017), recorded in the form the main text uses.

**Weight variance.** Let $L = \log\big[P_\theta(U, Z|A)/Q_{\text{lab}}(Tk, Z|Y)\big]$ denote the raw log-ratio under the proposal, so that the unnormalized weight is $w = e^{\alpha L}$, and let $\Lambda(s) = \log \mathbb{E}_{Q_{\text{lab}}}[e^{sL}]$ be its cumulant generating function. By the strong law applied to numerator and denominator, the asymptotic effective sample size fraction is exactly

$$\lim_{M\to\infty} \frac{\text{ESS}}{M} = \frac{\mathbb{E}[w]^2}{\mathbb{E}[w^2]} = \exp\big(2\Lambda(\alpha) - \Lambda(2\alpha)\big)$$

**Proposition 2** (Variance Control)**.** (i) *If $L$ is sub-Gaussian with variance proxy $\sigma^2$, then*

$$\lim_{M\to\infty} \frac{\text{ESS}}{M} \ \geq\ e^{-2\alpha^2\sigma^2}$$

*so any target retention $\rho$ is achieved by $\alpha \leq \sqrt{\log(1/\rho)/(2\sigma^2)}$.* (ii) *If $L$ is heavy-tailed with $\Lambda(s) < \infty$ if and only if $|s| < \lambda$, then $\text{Var}[w] < \infty$ if and only if $\alpha < \lambda/2$.*

Part (ii) is the sharp statement of the tempering effect: when $\lambda \leq 2$, the untempered estimator ($\alpha = 1$) has *infinite* variance, and $\alpha$-tempering is not a convenience but a necessity. Part (i) quantifies the price in the well-behaved regime.

### D.4 Proof of Proposition 2 (Variance Control)

Let $L$ be the raw log-ratio under the proposal and $W = e^{\alpha L}$, with cumulant generating function $\Lambda(s) = \log \mathbb{E}_{Q_{\text{lab}}}[e^{sL}]$.

**Asymptotic ESS.** The effective sample size is $\text{ESS} = (\sum_j w_j)^2 / \sum_j w_j^2$. Dividing numerator and denominator by $M^2$ and $M$ respectively, the strong law gives $\frac{1}{M}\sum_j w_j \to \mathbb{E}[W]$ and $\frac{1}{M}\sum_j w_j^2 \to \mathbb{E}[W^2]$ almost surely whenever both moments are finite, so

$$\frac{\text{ESS}}{M} = \frac{\big(\frac{1}{M}\sum_j w_j\big)^2}{\frac{1}{M}\sum_j w_j^2} \ \longrightarrow\ \frac{\mathbb{E}[W]^2}{\mathbb{E}[W^2]} = \exp\big(2\Lambda(\alpha) - \Lambda(2\alpha)\big)$$

**Part (i): sub-Gaussian case.** Let $\mu = \mathbb{E}[L]$. Sub-Gaussianity with proxy $\sigma^2$ gives the upper bound $\Lambda(s) \le s\mu + \frac{1}{2}s^2\sigma^2$ for all $s$, while Jensen's inequality gives the lower bound $\Lambda(s) \ge s\mu$. Applying the first at $s = 2\alpha$ and the second at $s = \alpha$:

$$\Lambda(2\alpha) - 2\Lambda(\alpha) \;\le\; \big(2\alpha\mu + 2\alpha^2\sigma^2\big) - 2\alpha\mu \;=\; 2\alpha^2\sigma^2$$

Therefore $\lim_{M\to\infty} \mathrm{ESS}/M = \exp\big(-(\Lambda(2\alpha) - 2\Lambda(\alpha))\big) \ge e^{-2\alpha^2\sigma^2}$. Setting this at least $\rho$ and solving gives $\alpha \le \sqrt{\log(1/\rho)/(2\sigma^2)}$.

**Part (ii): heavy-tailed case.** By definition $\mathbb{E}[W^2] = \mathbb{E}[e^{2\alpha L}] = e^{\Lambda(2\alpha)}$, which is finite if and only if $2\alpha < \lambda$. Since $\mathbb{E}[W] = e^{\Lambda(\alpha)}$ is finite whenever $\alpha < \lambda$, the variance $\mathrm{Var}[W] = \mathbb{E}[W^2] - \mathbb{E}[W]^2$ is finite if and only if $\alpha < \lambda/2$. □

**Sequence-length corollary.** Write $L = \sum_{t=1}^{T} L_t$ and split it into a martingale part and a compensator, $L = M_T + A_T$ with $M_T = \sum_t \big(L_t - \mathbb{E}[L_t|\mathcal{F}_{t-1}]\big)$ and $A_T = \sum_t \mathbb{E}[L_t|\mathcal{F}_{t-1}]$. If the increments of $M_T$ are conditionally sub-Gaussian with proxy $\sigma_0^2$, the Azuma–Hoeffding cumulant bound makes $M_T$ sub-Gaussian with proxy $T\sigma_0^2$. If in addition $A_T$ is sub-Gaussian with proxy $\sigma_A^2$, the triangle inequality for the sub-Gaussian norm gives $\sigma \lesssim \sqrt{T}\sigma_0 + \sigma_A$. Whenever $\sigma_A = O(\sqrt{T})$, Part (i) then requires $\alpha = O(1/\sqrt{T})$ to maintain $\mathrm{ESS}/M \ge \rho$. Without the martingale structure, only $\sigma \le T\sigma_0$ is available, which yields the weaker requirement $\alpha = O(1/T)$. This is the familiar collapse of importance weights with growing dimension (Bengtsson et al., 2008; Agapiou et al., 2017), with turns in place of dimensions; Daudel et al. (2023) analyze the corresponding weight collapse for the VR-IWAE bound. □

# E Smoothing vs. Filtering: Information and Identifiability

## E.1 What Filtering Discards

It could be argued that using a filtering labeler—one that conditions only on the past—to augment observed trajectories is statistically sound for off-policy evaluation. The argument rests on the theory of *Predictive State Representations* (PSRs) (Littman et al., 2001): the observable history is a sufficient statistic for predicting the future behavior of any policy that does not observe the latent state. Since PSRs are Markovian, a valid sequence model can in principle be learned from this data, treating filtered labels as full observations. This reasoning is correct in the limit of infinite, sufficiently diverse data.

However, the filtering posterior discards a specific and quantifiable amount of information. The following two results make this precise. Throughout, $\bar{H}_t \equiv H_t^{aug} = (a_1, u_1, \ldots, u_{t-1}, a_t)$ is the observable past at turn $t$, and $Y_{>t} = (u_t, a_{t+1}, u_{t+1}, \ldots)$ is the rest of the conversation, so that the full conversation is $Y = (\bar{H}_t, Y_{>t})$.

**Proposition 3** (Latent Decoupling under Filtering)**.** *Let $Q_F$ be a filtering labeler, i.e., $Q_F(z_t|Y, Z_{<t}) = Q_F(z_t|\bar{H}_t, Z_{<t})$. Under the implied joint $\mathbb{P}_F(Y, Z) = P_{\mathrm{data}}(Y) \cdot Q_F(Z|Y)$, the future trajectory is conditionally independent of the entire latent state history given only the observable past:*

$$Y_{>t} \;\perp_{\mathbb{P}_F}\; Z_{1:t} \mid \bar{H}_t \qquad \forall\, t$$

*Proof.* Summing $Q_F(Z|Y)$ over $z_{t+1}, \ldots, z_T$ gives

$$\mathbb{P}_F(Z_{1:t} \mid Y) = \prod_{s=1}^{t} Q_F(z_s \mid \bar{H}_s, Z_{<s}).$$

Each factor depends on $Y$ only through $\bar{H}_s$ with $s \le t$, which is a function of $\bar{H}_t$. Hence $\mathbb{P}_F(Z_{1:t} \mid Y) = \mathbb{P}_F(Z_{1:t} \mid \bar{H}_t)$, which is the stated conditional independence. □

Any model faithfully trained on $\mathbb{P}_F$ satisfies $P_\theta(Y_{>t}|\bar{H}_t, Z_{1:t}) = P_\theta(Y_{>t}|\bar{H}_t)$: the latent states carry zero predictive information about future behavior beyond the surface text.

We measure what the future reveals about the present latent state with the conditional $\chi^2$-information

$$I_{\chi^2}(A; B \mid C) \triangleq \mathbb{E}_C\Big[\chi^2\big(P(A, B \mid C) \,\big\|\, P(A \mid C)\, P(B \mid C)\big)\Big].$$

We use $\chi^2$ rather than KL because $\chi^2$ is the quantity that governs both importance-weight degeneracy ($\lim_{M\to\infty} \mathrm{ESS}/M = 1/(1+\chi^2)$, Appendix D.3) and off-policy error (Theorem 8 below).

**Proposition 4** (Smoothing–Filtering Information Gap)**.** *For any joint $P(Y, Z)$, the expected $\chi^2$-divergence between the smoothing and filtering posteriors equals the conditional $\chi^2$-information:*

$$\mathbb{E}_{P(Y,Z_{<t})}\Big[\chi^2\big(P(z_t|Y, Z_{<t}) \,\big\|\, P(z_t|\bar{H}_t, Z_{<t})\big)\Big] = \Delta_t, \qquad \Delta_t \triangleq I_{\chi^2}(z_t;\, Y_{>t} \mid \bar{H}_t, Z_{<t}).$$

*$\Delta_t$ measures how much the future trajectory reveals about the present latent state beyond what the observable past already provides.*

*Proof.* Condition throughout on $(\bar{H}_t, Z_{<t})$ and write $P(\cdot)$ for the resulting conditional law. Since $\sum_{z,y} \frac{P(z,y)^2}{P(z)P(y)} - 1 = \mathbb{E}_{P(y)}\big[\sum_z \frac{P(z|y)^2}{P(z)}\big] - 1$,

$$I_{\chi^2}(z_t;\, Y_{>t}) = \mathbb{E}_{P(Y_{>t})}\Big[\chi^2\big(P(z_t \mid Y_{>t}) \,\big\|\, P(z_t)\big)\Big].$$

Since $Y = (\bar{H}_t, Y_{>t})$, the inner posterior $P(z_t \mid Y_{>t})$ is exactly the smoothing posterior $P(z_t \mid Y, Z_{<t})$, while the reference $P(z_t)$ is the filtering posterior $P(z_t \mid \bar{H}_t, Z_{<t})$. Taking the outer expectation over $P(Y, Z_{<t})$ gives the stated identity. □

For latent states to encode hidden causes of behavior rather than summaries of the surface text, $\Delta_t$ must be positive under the true process. Under $\mathbb{P}_F$, $\Delta_t = 0$ at every $t$ (Proposition 3): filtered labels carry none of this information.

This has two consequences. It weakens the SNIS-EM update (Sec. E.2), and it makes the simulator's off-policy predictions wrong in a way that no amount of on-policy data reveals (Sec. E.3).

### E.2 The Cost of Filtering for SNIS-EM

This section asks what a filtering labeler costs the SNIS-EM update. The standard EM convergence analysis does not settle this, because it assumes an exact E-step. Dempster et al. (1977) showed that near a fixed point $\theta^*$, EM converges linearly with rate matrix $D_M = I_{\mathrm{mis}} \cdot I_{\mathrm{com}}^{-1}$, where $I_{\mathrm{com}}$ is the complete-data Fisher information and $I_{\mathrm{mis}} = I_{\mathrm{com}} - I_{\mathrm{obs}}$ is the information lost by not observing the latent variables. The rate is the spectral radius of $D_M$. A filtering labeler does not provide an exact E-step. At $\alpha = 0$ the update is supervised fine-tuning on the labels, so it fits $\mathbb{P}_F$, whose latent states are decoupled from the future (Proposition 3). Because the simulator conditions on the full history, this fit can reach the maximal observed-data likelihood while leaving the latent states unused, so the likelihood does not reveal the problem. For $\alpha > 0$ the importance weights must recover the missing dependence, and the cost of doing so is a sample-size requirement.

Importance weights with proposal $Q$ and target $P$ have $\mathrm{ESS}/M \to 1/(1+\chi^2(P\|Q))$ (Appendix D.3), and $\chi^2(P\|Q) \geq \exp(D_{\mathrm{KL}}(P\|Q)) - 1$. KL, unlike $\chi^2$, is additive under the chain rule, so we state the requirement in Shannon terms. Let $I_t \triangleq I(z_t; Y_{>t} \mid \bar{H}_t, Z_{<t})$ be the Shannon counterpart of $\Delta_t$. It vanishes if and only if $\Delta_t$ does, and $I_t \leq \log(1+\Delta_t)$ by Jensen. At $\alpha = 1$ the SNIS target is the model's posterior over the latent trajectory given the full conversation. Suppose the model is correct and the proposal is the ideal filter. By the chain rule, the KL divergence between target and proposal, averaged over conversations, is $\sum_t I_t$. By Jensen's inequality, the number of proposals needed per effective sample, averaged over conversations, then satisfies

$$\mathbb{E}_Y\left[\frac{M}{N}\right] \gtrsim \exp\Big(\sum_t I_t\Big).$$

This is a lower bound on the average requirement, not a worst case. At 0.3 nats per turn, comparable to the 0.32 nats a single outcome carries in the worked example (Appendix E.4), a 10-turn conversation needs $M \gtrsim 20\,N$. At 1 nat per turn it needs $M \gtrsim 2.2 \times 10^4\,N$.

### E.3 Off-Policy Error of Filtering-Consistent Simulators

Simulators are most useful where there is no data: evaluating agent policies that differ from those that generated the training corpus. In this setting the total-support condition of the PSR argument fails by design. Predictions at new histories then depend on having identified how latent states affect outcomes, so that this relation can be reused at histories not seen in training.

We showed in Proposition 3 that imputing the latent state with a filter instead of a smoother produces a distribution in which the outcome is independent of the latent state given the history. In the following, we study what this particular failure mode means for identifiability. To do so, we simplify the setting: the dynamics are fully known, but the labeler fails to perceive the full dependence between the latent state and the outcome.

**Definition 5** (Setting and Filtering Equivalence Class)**.** Fix a training history $H_b$ and the action $a$ taken there. Let $Z \in \{1, \dots, K\}$ be the latent state at a single turn and $Y \in \{0, 1\}$ a single binary outcome that follows it; in this section $Z$ and $Y$ do not denote whole trajectories. We suppress $a$ from the notation where it is fixed. We assume:

(i) *Latent channel.* Histories affect the outcome only through the state distribution: $P(Y{=}1 \mid H, a) = \sum_k P(Z{=}k \mid H)\, \phi_k$, where the emission probability $\phi_k = P(Y{=}1 \mid Z{=}k, a)$ is shared across histories. This holds for the true process and for every simulator we consider.

(ii) *Same action, reachable shifts.* The evaluation policy $\pi_e$ reaches a history $H_e$ at which it takes the same action $a$. It differs from $\pi_b$ only in how it gets there, which changes the state distribution from $q_b^k = P(Z{=}k \mid H_b)$ to $q_e^k = P(Z{=}k \mid H_e)$. Every $q_e$ in the probability simplex is reached by some evaluation history.

(iii) *Full support.* $q_b^k > 0$ for all $k$, and $c_b \triangleq P(Y{=}1 \mid H_b) \in (0, 1)$.

(iv) *Known dynamics.* Simulators use the true state distribution $P(Z \mid H)$ at every history and differ from the truth only in the emission.

The *filtering equivalence class* is the set of emission probabilities consistent with the observational data at $H_b$:

$$\mathcal{M}_F = \Big\{ \phi \in [0,1]^K \;\Big|\; \textstyle\sum_k q_b^k \phi_k = c_b \Big\}.$$

All emission probabilities in $\mathcal{M}_F$ give the same outcome distribution at $(H_b, a)$, so the outcome rate at this pair does not distinguish them. Under filtering, this rate is the only information the training labels carry about the dependence between $Z$ and $Y$ (Proposition 3). Write $\phi^* \in \mathcal{M}_F$ for the true emission probabilities and $\Delta \triangleq \chi^2\big(P^*(Z, Y \mid H_b) \,\big\|\, q_b \otimes (1 - c_b, c_b)\big)$ for its $\chi^2$-information at $H_b$.

Assumption (i) is what makes off-policy prediction possible. A fully expressive conditional $P(Y \mid H, a)$, with no structure shared across histories, is unconstrained at histories not seen in training, so nothing can be said about it.

Two further points determine when the setting applies. First, we assume that the latent model is identifiable from full trajectories, which is what a smoother uses. Otherwise neither a filter nor a smoother can recover it, and there is nothing to compare. Second, because the emission probabilities are shared across histories, the outcome rate at each training history adds a linear constraint $\sum_k P(Z{=}k \mid H)\, \phi_k = P(Y{=}1 \mid H)$. These constraints identify $\phi$ only if the training histories provide $K$ linearly independent state distributions. In practice $Z$ ranges over a product space (e.g., combinations of goals, constraints, and preferences), so $K$ is much larger than the number of independent training histories, and $\mathcal{M}_F$ remains large.

To compare simulators in $\mathcal{M}_F$, we need a measure of how far a simulator's emission probabilities are from the true ones. At $H_b$, the truth induces a joint distribution over state and outcome, $P^*(Z{=}k, Y{=}1 \mid H_b) = q_b^k \phi_k^*$. Every simulator in $\mathcal{M}_F$ has the same two marginals: the state distribution $q_b$, which is known, and the outcome rate $c_b$, which the data fixes. Simulators differ only in the dependence between $Z$ and $Y$. We therefore measure the distance between the simulator's joint and the true joint relative to the product of the two marginals. This is the same reference distribution that defines $\Delta$.

**Definition 6** (Coupling Residual)**.** For a simulator $M$ with emission $\phi \in \mathcal{M}_F$, let $P^M(Z{=}k, Y{=}1 \mid H_b) = q_b^k \phi_k$ and $P^M(Z{=}k, Y{=}0 \mid H_b) = q_b^k(1-\phi_k)$, and write $c_1 = c_b$, $c_0 = 1-c_b$. The coupling residual of $M$ is

$$D(M) \triangleq \sum_{k=1}^{K} \sum_{y\in\{0,1\}} \frac{\big(P^M(Z{=}k, Y{=}y \mid H_b) - P^*(Z{=}k, Y{=}y \mid H_b)\big)^2}{q_b^k\, c_y}.$$

**Lemma 7** (Coupling Residual as Emission Error)**.** *For every $\phi \in \mathcal{M}_F$, with $\|f\|^2_{L_2(q_b)} = \sum_k q_b^k f_k^2$,*

$$D(M) = \frac{\|\phi - \phi^*\|^2_{L_2(q_b)}}{c_b(1-c_b)}.$$

*In particular: (a) $D(M) = 0$ if and only if $\phi = \phi^*$; (b) the state-blind simulator $M_{\text{blind}}$, with $\phi_k \equiv c_b$, has*

$$D(M_{\text{blind}}) = \frac{\sum_k q_b^k(\phi_k^* - c_b)^2}{c_b(1-c_b)} = \Delta. \tag{8}$$

*Proof.* Let $\delta = \phi - \phi^*$. The $Y{=}1$ terms contribute $\sum_k (q_b^k\delta_k)^2/(q_b^k c_b) = \|\delta\|^2_{L_2(q_b)}/c_b$, and the $Y{=}0$ terms contribute $\|\delta\|^2_{L_2(q_b)}/(1-c_b)$. Their sum is the display. Part (a) follows from $q_b^k > 0$. For (b), the joint of $M_{\text{blind}}$ is $q_b \otimes (c_0, c_1)$, the product of the marginals of $P^*(Z, Y \mid H_b)$, so $D(M_{\text{blind}}) = \chi^2\big(P^*(Z, Y \mid H_b) \,\big\|\, q_b \otimes (c_0, c_1)\big) = \Delta$. The middle expression is the display with $\phi = c_b\mathbf{1}$. □

The factor $1/(c_b(1-c_b))$ puts $D(M)$ on the same scale as $\Delta$, so that the state-blind simulator has $D(M_{\text{blind}}) = \Delta$ exactly.

Writing $P(Z{=}k, Y{=}y \mid H_b) = c_y\, P(Z{=}k \mid H_b, Y{=}y)$ for both joints gives $D(M) = \mathbb{E}_{Y|H_b}\big[\sum_k (P^M(Z{=}k \mid H_b, Y) - P^*(Z{=}k \mid H_b, Y))^2/q_b^k\big]$. The residual thus compares the simulator's and the truth's distribution over the state after the outcome is seen.

**Theorem 8** (Inescapable Off-Policy Error)**.** *Let $M$ be a simulator with emission $\phi \in \mathcal{M}_F$, $\phi \neq \phi^*$, and let $\delta = \phi - \phi^*$. For every shift budget $0 < \rho \le \rho_{\max}(M)$,*

$$\max_{H_e\,:\,\chi^2(q_e\|q_b)\le\rho} \big|P^M(Y{=}1 \mid H_e) - P^*(Y{=}1 \mid H_e)\big| \;=\; \sqrt{D(M)\cdot c_b(1-c_b)\cdot\rho}, \tag{9}$$

*where*

$$\rho_{\max}(M) \;=\; \max_{\sigma\in\{\pm1\}} \; \min_{k\,:\,\sigma\delta_k<0} \frac{\|\delta\|^2_{L_2(q_b)}}{\delta_k^2} \;>\; 0.$$

The maximum is attained: for every budget up to $\rho_{\max}(M)$ there is an evaluation history at which the simulator makes exactly this error. If $\phi = \phi^*$, the error is zero at every history.

*Proof.* By assumptions (i) and (iv), the simulator and the truth use the same $q_e$ at $H_e$, so the error there is $\sum_k q_e^k\delta_k$. Both lie in $\mathcal{M}_F$, so $\sum_k q_b^k\delta_k = 0$ and, with $\langle f, g\rangle_{q_b} = \sum_k q_b^k f_k g_k$,

$$P^M(Y{=}1 \mid H_e) - P^*(Y{=}1 \mid H_e) = \sum_k (q_e^k - q_b^k)\,\delta_k = \Big\langle \frac{q_e - q_b}{q_b},\, \delta \Big\rangle_{q_b}. \tag{10}$$

By Cauchy–Schwarz in $L_2(q_b)$, the magnitude of equation 10 is at most $\sqrt{\chi^2(q_e\|q_b)}\,\|\delta\|_{L_2(q_b)}$. By Lemma 7, at every evaluation history

$$\big|P^M(Y{=}1 \mid H_e) - P^*(Y{=}1 \mid H_e)\big| \;\le\; \sqrt{D(M)\cdot c_b(1-c_b)\cdot\chi^2(q_e\|q_b)}. \tag{11}$$

This gives the upper bound in equation 9. For $q_e \neq q_b$, equality holds if and only if the state shift is aligned with the error in the emission probabilities: $(q_e^k - q_b^k)/q_b^k = t\,\delta_k$ for some $t \neq 0$.

For attainment, set $q_e^k = q_b^k\big(1 + \sigma\sqrt{\rho}\,\delta_k/\|\delta\|_{L_2(q_b)}\big)$ with $\sigma \in \{\pm1\}$. It sums to one because $\delta$ has zero $q_b$-mean, it has $\chi^2(q_e\|q_b) = \rho$, and it is aligned, so the error is $\sqrt{\rho}\,\|\delta\|_{L_2(q_b)}$. It is nonnegative if and only if $\rho \le \|\delta\|^2_{L_2(q_b)}/\delta_k^2$ for every $k$ with $\sigma\delta_k < 0$. Choosing the better $\sigma$ gives the condition $\rho \le \rho_{\max}(M)$. By assumption (ii), this $q_e$ is reached by some evaluation history. Finally, $\delta \neq 0$ has zero $q_b$-mean, so it has entries of both signs; both minima are over nonempty sets, and $\rho_{\max}(M) > 0$. □

The theorem applies to every simulator that fits the observational data. The worst case is over evaluation histories, not over constructed emission probabilities. The error depends on the simulator only through $D(M)$ and on the evaluation policy only through $\chi^2(q_e\|q_b)$.

A simulator trained on filtering labels has the state-blind emission probabilities.

**Corollary 9** (Filtering-Trained Simulators)**.** *Suppose the true emission is not state-blind ($\Delta > 0$). Let $M_{\text{blind}}$ be a simulator that satisfies assumption (i) and is fit exactly to $\mathbb{P}_F$ at $(H_b, a)$, i.e., trained with an exact filtering labeler and unlimited data at this pair. Then $\phi_k = c_b$ for all $k$, $M_{\text{blind}}$ predicts $c_b$ at every evaluation history, and for every $0 < \rho \le \rho_{\max}(M_{\text{blind}})$,*

$$\max_{H_e\,:\,\chi^2(q_e\|q_b)\le\rho} \left|P^{M_{\text{blind}}}(Y{=}1 \mid H_e) - P^{*}(Y{=}1 \mid H_e)\right| \;=\; \sqrt{\Delta \cdot c_b(1-c_b)\cdot\rho}.$$

*The corollary does not use assumption (iv).*

*Proof.* Proposition 3 gives $Y \perp_{\mathbb{P}_F} Z \mid H_b$, so the fitted emission probability is $\mathbb{P}_F(Y{=}1 \mid Z{=}k, H_b) = c_b$. By assumption (i), the prediction at $H_e$ is $\sum_k P^{M_{\text{blind}}}(Z{=}k \mid H_e)\, c_b = c_b$, whatever the simulator's state distribution. Its error at $H_e$ is therefore the same as that of a simulator with the same emission probabilities and the true dynamics. Apply Theorem 8 with $D(M_{\text{blind}}) = \Delta$ (Lemma 7); $\Delta > 0$ gives $\phi^* \neq c_b\mathbf{1}$. □

Between the state-blind and the true emission probabilities lie $\phi^{(s)} = c_b\mathbf{1} + s(\phi^* - c_b\mathbf{1})$, $s \in [0,1]$, of simulators that recover a fraction $s$ of the dependence. By Lemma 7, $D(M^{(s)}) = (1-s)^2\Delta$. For $s < 1$, $\rho_{\max}$ depends only on the direction of $\delta$, so the worst-case error on the same budgets is $(1-s)\sqrt{\Delta \cdot c_b(1-c_b)\cdot\rho}$. The $\alpha$-term of the SNIS-EM update rewards latent states that predict the outcome, which moves the emission probabilities away from $c_b\mathbf{1}$ (Appendix D.2). Emission probabilities off this segment can have $D(M) > \Delta$: in the worked example below, $\phi = (17/35, 0) \approx (0.486, 0)$ lies in $\mathcal{M}_F$ and has $D(M) \approx 1.55$, against $\Delta \approx 0.60$, with the same $\rho_{\max}$. A simulator that learns the wrong dependence can therefore do worse than a state-blind one.

### E.4 Worked Example

Consider a conversational recommendation setting with two latent user types: *budget-constrained* ($Z = 1$) and *flexible* ($Z = 2$), with outcome $Y = 1$ indicating positive engagement with the final recommendation $a$. Under a training agent $\pi_b$ that opens with mid-range products, the state distribution at the recommendation step is $q_b = (0.7,\ 0.3)$ and the engagement rate is $c_b = 0.34$. The true per-type emission probabilities are $\phi^*_{\text{budget}} = 0.1$ and $\phi^*_{\text{flex}} = 0.9$. Training data labeled by a filter shows only the marginal rate of 34% and cannot decompose it into per-type rates.

**What the outcome reveals.** A labeler who observes the outcome can update its belief about the user type substantially:

| | Filtering $P(Z = \text{budget} \mid H_b)$ | Smoothing $P(Z = \text{budget} \mid H_b, Y)$ |
|---|---|---|
| $Y = 1$ (engage) | 0.70 | $0.07/0.34 = 0.206$ |
| $Y = 0$ (disengage) | 0.70 | $0.63/0.66 = 0.955$ |

Engagement is strong evidence *against* budget-consciousness, while disengagement is strong evidence *for* it. By equation 8, the information the filtering labeler discards is $\Delta = \frac{0.7(0.1-0.34)^2+0.3(0.9-0.34)^2}{0.34\times0.66} = \frac{0.1344}{0.2244} \approx 0.599$ (0.316 nats in Shannon terms).

**What filtering costs.** The evaluation agent $\pi_e$ opens with premium products before making the same final recommendation. Budget-constrained users disengage early in such conversations, so among conversations that reach the recommendation the state distribution is $q_e = (0.3,\ 0.7)$, a shift of $\chi^2(q_e\|q_b) = 0.4^2/0.7 + 0.4^2/0.3 = 0.762$. As assumption (i) requires, the example takes the premium opening to change which users reach the recommendation, but not how each type responds to it. Three simulators, all consistent with the training data, predict:

| Simulator | $\phi_{\text{budget}}$ | $\phi_{\text{flex}}$ | $D(M)$ | $P(Y=1\|H_b)$ | $P(Y=1\|H_e)$ |
|---|---|---|---|---|---|
| $M_{\text{true}}$ | 0.10 | 0.90 | 0 | 0.34 | **0.66** |
| $M_{\text{partial}}$ | 0.22 | 0.62 | 0.150 | 0.34 | **0.50** |
| $M_{\text{blind}}$ | 0.34 | 0.34 | 0.599 | 0.34 | **0.34** |

The filtering-trained simulator $M_{\text{blind}}$ is off by 0.32. A team evaluating the premium strategy with it would conclude that it performs no better than the training agent (0.34 vs. 0.34), when in fact it raises engagement from 0.34 to 0.66. Theorem 8 gives $\sqrt{0.599 \times 0.2244 \times 0.762} = \sqrt{0.1024} = 0.32$ as the largest error over all evaluation histories with a shift at most this size, and this history attains it. With $K = 2$ the filtering constraint leaves a single free direction, so every error in the emission probabilities is aligned with every shift: $q_b \odot (\phi^{\text{blind}} - \phi^*) = (0.168, -0.168) \propto q_e - q_b$. The simulator $M_{\text{partial}}$, which recovers half of the dependence ($s = 1/2$), has a quarter of the residual and half of the error: $\sqrt{0.150 \times 0.2244 \times 0.762} = 0.16 = |0.50 - 0.66|$.

### E.5 Scope and Caveats

**Reachable shifts.** Theorem 8 takes the maximum over evaluation histories and assumes that every state distribution is reachable (assumption (ii)). If a class of evaluation policies reaches only some $q_e$, the error at each is the inner product equation 10. It is at most equation 11, with equality only for aligned shifts. For budgets beyond $\rho_{\max}(M)$ the aligned shift leaves the simplex. The bound equation 11 still holds, but the maximum is strictly smaller than $\sqrt{D(M) \cdot c_b(1 - c_b) \cdot \rho}$ and depends on the simplex boundary.

**Known dynamics.** Assumption (iv) fixes the simulator's state distribution at the truth. A simulator with wrong dynamics has an additional error term, $\sum_k (P^M(Z{=}k \mid H_e) - P^*(Z{=}k \mid H_e))\, \phi_k$, which Theorem 8 does not cover. The filtering-trained simulator is not affected, because its emission probability does not depend on $Z$ (Corollary 9).

**Multi-step evaluation.** The analysis is single-step. How the error composes over a trajectory, under open-loop or closed-loop evaluation, is left to future work.

**Binary outcomes.** The results are stated for $Y \in \{0, 1\}$. For general outcomes (e.g., token sequences), apply them to the indicator of any event $A$, with $c_b$ and $\phi_k$ replaced by $P(A \mid H_b)$ and $P(A \mid Z{=}k, a)$. Since $D_{\text{TV}}$ is the supremum over events, the error attained for any single event is a lower bound on the total-variation error.

## F Experimental Protocol and Extended Results

### F.1 Agent Personas and Counterfactual Split Specifications

ConvApparel-V2 contains human-study conversations with 11 footwear shopping assistant personas; their exact system prompts are given by Tennenholtz et al. (2026). Table 2 lists the personas and the two counterfactual splits we use. In both splits, simulators are trained only on dialogues with the seen assistants and evaluated in closed-loop rollouts on $n{=}256$ episodes sampled uniformly at random from dialogues with the unseen assistants. In the $5 \to 6$ split (main text), the seen set pairs one competent baseline recommender with four *stylistic* variations (verbosity, affect, confidence, and bias), whereas the unseen set changes the assistant's *interaction strategy* (expertise, questioning style, extreme verbosity, literalness, sales pressure, and visual framing), so good performance requires responding to new conversational moves rather than to new phrasings. The $3 \to 8$ split (Appendix F.4) restricts training to three competent assistants and tests transfer to all remaining stylized and strategy-shifted personas.

### F.2 Metric Definitions

This section defines every metric in the complete result tables (Tables 4 and 5). Three of them, words per user turn (W/T), requirement violations (Viol.), and pushback authenticity (Push.), appear only in

Table 2: **Assistant personas and counterfactual splits.**

| Persona | Interaction style | $5 \to 6$ | $3 \to 8$ |
|---|---|---|---|
| Good Recommender | Competent baseline recommender | Seen | Seen |
| Efficient Matchmaker | Terse bullet points | Seen | Unseen |
| Empathetic Listener | Supportive, validation-first | Seen | Seen |
| Hesitant Assistant | Risk-averse hedging | Seen | Unseen |
| Trend Chaser | Social proof and slang | Seen | Unseen |
| Domain Expert | Technical material science | Unseen | Seen |
| Patient Guide | Socratic binary A/B questions | Unseen | Unseen |
| Enthusiastic Rambler | Hyperactive anecdotes | Unseen | Unseen |
| Literal Thinker | Rigid database parser | Unseen | Unseen |
| Mild Upseller | Status-focused upselling | Unseen | Unseen |
| Visual Stylist | Outfit styling | Unseen | Unseen |
| Dialogues with seen assistants (training) | | 1,595 | 960 |
| Dialogues with unseen assistants (evaluation pool) | | 1,939 | 2,574 |

those tables and are not discussed in the main text. Let $c = 1, \ldots, n$ index evaluation conversations, let $T_c$ be the number of user turns in conversation $c$, and let $u_{c,1}, \ldots, u_{c,T_c}$ be these turns; the first turn $u_{c,1}$ is the human's seed turn, shared by all simulators.

**Likelihood.** $\mathrm{LL}_{\text{eval}}$ is the per-token teacher-forced log-likelihood of all human user utterances in $\mathcal{D}_{\text{eval}}^{5\to6}$ given the ground-truth prefix; every model scores the same utterance tokens. Latent-state models additionally condition on a latent state before each utterance. These states come from the anchor labeler's posterior smoother, and the *same* states are given to every latent-state model, including the model trained on **Filtering** labels, so that likelihood differences among latent-state models reflect the models rather than the annotations. Because they condition on these states, their likelihoods are not directly comparable to those of latent-free models. $\mathrm{LL}_{\text{train}}$ is computed identically on 128 randomly sampled conversations from the seen agents, and $\Delta \log P_\theta = \mathrm{LL}_{\text{eval}} - \mathrm{LL}_{\text{train}}$ measures how much of the fit to the seen agents transfers to unseen ones. For the prompted models, which are not trained on either split, we report the same quantities for completeness; their gap reflects differences between the two conversation sets rather than overfitting, so they are excluded from the $\Delta \log P_\theta$ marking. Error bars in Figs. 2a and 7a are paired-bootstrap standard errors (2,000 resamples of conversations, drawn jointly across models).

**Deterministic statistics.** With $\hat{F}_{\text{sim}}$ and $\hat{F}_{\text{hum}}$ the empirical CDFs of $T_c$ for simulated and human conversations, the turn-count distance is $W_1 = \sum_{k\geq 0} |\hat{F}_{\text{sim}}(k) - \hat{F}_{\text{hum}}(k)|$. W/T is the mean number of words per user turn. Lexical diversity is the per-conversation type-token ratio of the user's lower-cased word tokens, averaged over conversations. Repetition is

$$\text{Rep} = \frac{1}{n} \sum_{c=1}^{n} \sum_{t=2}^{T_c} \mathbb{I}\big[u_{c,t} = u_{c,t-1}\big],$$

the mean number of user turns per conversation that exactly repeat the preceding user turn (after stripping whitespace); for example, Rep $= 0.02$ corresponds to about five such turns across 256 conversations.

**LLM judge.** All judge metrics come from GEMINI 2.5 FLASH at temperature 0, applied identically to human and simulated conversations. The judge is prompted independently for three rubrics, each in a separate call with its own prompt (Appendix H.2): *dialogue acts*, *task fidelity*, and *shopper authenticity*.

**Dialogue acts.** The dialogue-act rubric labels each user turn with one or more acts from the 10-act shopping taxonomy in Table 3. We compute the metric over the seven acts marked there ($\mathcal{A}$), which describe how the user negotiates and resolves recommendations. The three excluded acts account for two-thirds of human act labels on $5 \to 6$, and two of them mostly reflect the persona and goal that seed the conversation rather than the response to the assistant. `OPEN_INQUIRY` occurs almost only

in the shared human seed turn (96–100% of seed turns for every method vs. at most 1.5% of later turns). SPECIFY_CONSTRAINT is the most frequent act in the first simulated turn, for humans (58% of these turns) and for every simulator (60–83%). It differs from REFINE_CONSTRAINT by the judge's definitions: it states a *new* requirement, whereas REFINE_CONSTRAINT narrows or adjusts a requirement already stated, typically after seeing recommendations. REQUEST_INFO occurs throughout the conversation, at similar rates for humans and all fine-tuned simulators (0.49–0.67 vs. 0.57 per conversation). For a set of turn indices $\mathcal{T}$ (all turns, or $t \geq 3$), let $r_m(a; \mathcal{T})$ be the number of user turns in $\mathcal{T}$ labeled with act $a$ by model $m$, divided by the number of conversations. The dialogue-act $L_1$ distance is

$$L_1^{\text{act}}(m; \mathcal{T}) = \sum_{a \in \mathcal{A}} \left| r_m(a; \mathcal{T}) - r_{\text{hum}}(a; \mathcal{T}) \right|.$$

Normalizing per conversation rather than per turn makes the metric sensitive both to which acts a simulator performs and to how often it performs them over the dialogue. Because restricting to $\mathcal{A}$ is a design choice, we also computed the metric over all ten acts; this does not change the conclusions on $5 \to 6$: **RUSTLeD** ($\alpha$=0.008) has the lowest divergence both over all turns (0.371, vs. 0.457 for **SFT (No LS)**) and at $t \geq 3$ (0.281, vs. 0.363 for **SFT + LS** (Smoother) and 0.391 for **SFT (No LS)**).

Table 3: **Dialogue-act taxonomy.** Definitions are from the judge prompt (Appendix H.2). Human rate: acts per conversation on $\mathcal{D}_{\text{eval}}^{5\to6}$. ✓: included in $\mathcal{A}$.

| Act | Definition | Human rate | In $\mathcal{A}$ |
|---|---|---|---|
| OPEN_INQUIRY | Opens with a need, without firm constraints. | 0.99 | |
| SPECIFY_CONSTRAINT | States a new requirement. | 0.93 | |
| REQUEST_INFO | Asks for more detail about a shown item. | 0.57 | |
| REFINE_CONSTRAINT | Narrows or adjusts an existing requirement. | 0.25 | ✓ |
| ACCEPT_RECOMMENDATION | Signals approval of a shown item. | 0.39 | ✓ |
| REJECT_RECOMMENDATION | Signals disapproval of a shown item. | 0.23 | ✓ |
| COMPARE | Weighs two or more shown items against each other. | 0.01 | ✓ |
| CONFIRM_PURCHASE | Commits to buying. | 0.20 | ✓ |
| ABANDON | Gives up or leaves without buying. | 0.11 | ✓ |
| CLARIFY | Answers or asks for clarification of the assistant. | 0.09 | ✓ |

**Outcome and quality judgments.** The task-fidelity and shopper-authenticity rubrics see the persona, goal, and full conversation. The task-fidelity rubric assigns each conversation a goal-satisfaction label in {SATISFIED, PARTIAL, UNSATISFIED, ABANDONED}; Sat. is the percentage labeled SATISFIED ($\pm$1 SEM over conversations) and $|\Delta\text{Sat}|$ is its absolute difference from humans in percentage points (pp). It also flags whether the user ended up accepting an item that violates a stated requirement (Viol., % of conversations). The shopper-authenticity rubric rates human-likeness (Hum.) and pushback authenticity (Push.) on 1–5 scales (5 is best).

### F.3 Complete $5 \to 6$ Results

Table 4 reports all metrics for every trained configuration on $\mathcal{D}_{\text{eval}}^{5\to6}$, including the $M$=5 runs. Fig. 4 breaks the dialogue-act $L_1$ distance down by turn. Turn 1 is the shared human seed, so all models nearly coincide there ($L_1 \leq 0.004$); beyond it, the prompted baselines diverge increasingly with depth (**Prompted (With LS)**: 0.20 at $t$=2 to 1.02 at $t \geq 5$), whereas fine-tuned models stay between 0.06 and 0.22 at every turn. We report $t \geq 3$ separately because these later turns depend most on how the simulator has responded to the unseen agent so far.

**Fine-tuned baselines.** Each baseline matches humans on a different subset of columns. **SFT (No LS)** is second only to **RUSTLeD** in turn distribution ($W_1 = 0.195$) and all-turn act rates ($L_1^{\text{act}} = 0.297$), but its act divergence does not shrink at deeper turns (0.320); it also repeats its previous turn far more often than humans (0.17 vs. 0.02 per conversation), over-satisfies goals (27.0% vs. 10.5%), and accepts a requirement-violating item most often among all simulators (1.95% of conversations vs. 0.39% for humans); Section 4.2.2 analyzes these failures. Adding latent states from the **Filtering**

labeler keeps the close turn distribution ($W_1 = 0.203$) and earns the highest human-likeness score among simulators (3.95), but raises all-turn act divergence (0.402) and still repeats turns (0.17) and over-satisfies goals (18.8%). The **Smoother** labeler instead yields the lowest deeper-turn act divergence (0.227), near-human repetition (0.04), and a small goal-satisfaction bias (2.0 pp), at the cost of a looser turn distribution ($W_1 = 0.270$) and lower lexical diversity (TTR = 0.726 vs. 0.776). With $M$=5 proposals, both labelers fit the training split less closely ($\text{LL}_{\text{train}} = -1.311$ and $-1.279$, vs. $-1.110$ and $-1.081$ at $M$=15) and therefore show the smallest likelihood gaps ($-0.198$ and $-0.226$); their behavior is worse, most visibly for the Smoother ($W_1 = 0.461$, repetition 0.25), consistent with the gains from $M$=15 in Section 4.2.3. Held-out likelihood barely differs across latent-state models ($-1.506$ to $-1.519$), so it is the behavioral metrics that separate them.

Table 4: **Complete results on $\mathcal{D}_{\text{eval}}^{5\to6}$ ($n$=256).** W/T: words per user turn; Viol.: % of conversations accepting a requirement-violating item; Push.: pushback authenticity (1–5). Marking as in Table 1; —: not applicable or not computed.

| | Likelihood | | | Deterministic statistics | | | | | LLM judge | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | $\text{LL}_{\text{eval}}\uparrow$ | $\text{LL}_{\text{train}}$ | $\Delta\log P_\theta\uparrow$ | **Turns** | $W_1\downarrow$ | **W/T** | **TTR** | **Rep.** | $L_1^{\text{act}}\downarrow$ | $L_1^{\text{act},\,t\geq3}\downarrow$ | **Sat. (%)** | $\lvert\Delta\text{Sat}\rvert\downarrow$ | **Viol.** | **Hum.**↑ | **Push.**↑ |
| **Real Humans** | — | — | — | 3.50 | — | 11.7 | 0.776 | 0.02 | — | — | $10.5 \pm 1.9$ | — | 0.39 | 4.08 | 4.29 |
| Prompted (No LS) | −2.943 | −2.869 | −0.074 | 3.80 | 0.555 | 17.8 | 0.706 | **0.01** | 0.918 | 0.746 | $2.0 \pm 0.9$ | 8.6 | 0.00 | 3.62 | 4.10 |
| Prompted (With LS) | −2.592 | −2.530 | −0.063 | 5.11 | 1.605 | 21.0 | 0.625 | 0.04 | 1.785 | 1.691 | $2.3 \pm 0.9$ | 8.2 | 0.00 | 3.38 | 3.97 |
| SFT (No LS) | −3.001 | −0.943 | −2.058 | 3.69 | 0.195 | 12.1 | 0.752 | 0.17 | 0.297 | 0.320 | $27.0 \pm 2.8$ | 16.4 | 1.95 | 3.79 | 4.13 |
| SFT + LS (Filtering, $M$=5) | −1.510 | −1.311 | **−0.198** | 3.70 | 0.262 | 10.7 | 0.721 | 0.17 | 0.398 | 0.277 | $14.8 \pm 2.2$ | 4.3 | 1.17 | 3.87 | 4.06 |
| SFT + LS (Filtering, $M$=15) | −1.519 | −1.110 | −0.409 | 3.66 | 0.203 | 11.4 | 0.721 | 0.17 | 0.402 | 0.324 | $18.8 \pm 2.4$ | 8.2 | 0.78 | **3.95** | 4.12 |
| SFT + LS (Smoother, $M$=5) | **−1.506** | −1.279 | −0.226 | 3.96 | 0.461 | **11.7** | 0.697 | 0.25 | 0.496 | 0.355 | $15.6 \pm 2.3$ | 5.1 | 0.39 | 3.78 | 4.06 |
| SFT + LS (Smoother, $M$=15) | −1.518 | −1.081 | −0.437 | 3.62 | 0.270 | 11.5 | 0.726 | 0.04 | 0.352 | **0.227** | $12.5 \pm 2.1$ | 2.0 | 0.00 | 3.80 | 3.96 |
| **RUSTLeD** ($M$=15, $\alpha$=0.005) | −1.519 | −1.098 | −0.421 | 3.43 | 0.332 | 12.3 | **0.762** | 0.11 | 0.508 | 0.363 | $12.9 \pm 2.1$ | 2.3 | 0.39 | 3.63 | 3.92 |
| **RUSTLeD** ($M$=15, $\alpha$=0.008) | −1.513 | −1.124 | −0.388 | **3.55** | **0.188** | 12.7 | 0.744 | **0.03** | **0.270** | 0.230 | $17.2 \pm 2.4$ | 6.6 | 1.56 | 3.84 | **4.14** |
| **RUSTLeD** ($M$=15, $\alpha$=0.012) | −1.515 | −1.166 | −0.349 | 3.33 | 0.199 | 10.8 | 0.757 | 0.09 | 0.410 | 0.250 | $11.3 \pm 2.0$ | **0.8** | 1.56 | 3.89 | 4.00 |
| **RUSTLeD** ($M$=15, $\alpha$=0.020) | −1.518 | −1.209 | −0.309 | 3.34 | 0.316 | 10.4 | 0.759 | 0.06 | 0.461 | 0.336 | $13.7 \pm 2.2$ | 3.1 | 0.78 | 3.89 | 4.02 |

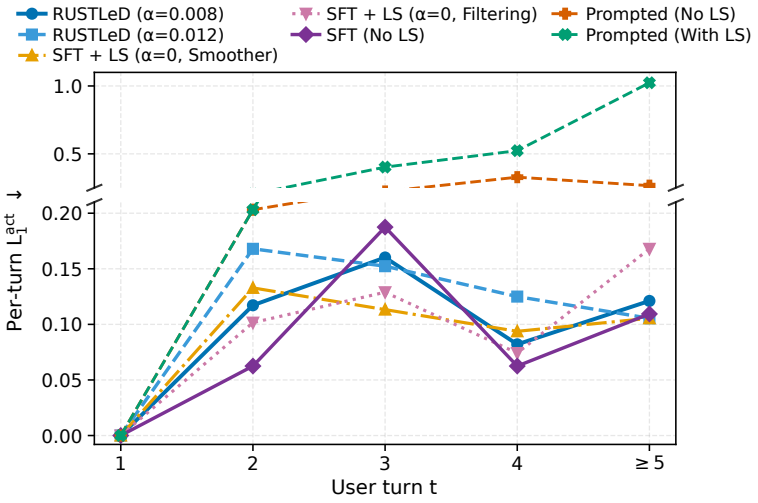



Figure 4: **Per-turn dialogue-act $L_1$ distance on $\mathcal{D}_{\text{eval}}^{5\to6}$.** The broken y-axis separates the prompted baselines (top) from the fine-tuned models (bottom).

### F.4 Harsher Policy Shift: the 3 → 8 Split

Table 5 and Figs. 6–8 repeat the evaluation with only three seen assistants; the three figures mirror Figs. 1, 2, and 3 of the main text. This split is harder for every simulator: the best turn $W_1$ rises from 0.188 on 5 → 6 to 0.457, and the best deeper-turn act divergence from 0.227 to 0.340. Fig. 5 gives the per-turn breakdown.

**What carries over.** **SFT (No LS)** again over-satisfies goals (18.4% vs. 8.6% for humans; Fig. 6b), repeats its previous turn over 10× as often as humans (0.23 vs. 0.02), and shows by far the largest generalization gap among fine-tuned models ($-0.733$ nats/token, vs. $-0.118$ to $-0.221$ for the

$M$=15 latent-state models). As on $5 \to 6$, it matches all-turn act frequencies well ($L_1^{\text{act}} = 0.492$, second best) but not the other behaviors. Relative to it, **RUSTLED** is closer to humans in turn distribution ($W_1 = 0.457$ at $\alpha$=0.012 vs. 0.891), goal satisfaction ($|\Delta\text{Sat}| = 2.0$ pp at $\alpha \in \{0.012, 0.020\}$ vs. 9.8 pp), and repetition (0.04 at $\alpha$=0.008), and is no worse in deeper-turn act divergence (0.449–0.484 vs. 0.484; Fig. 6a). Reweighting again shrinks the generalization gap monotonically with $\alpha$ (Fig. 7a), and $\alpha \in \{0.012, 0.020\}$ give the closest turn distributions among all methods (Fig. 7b). Unlike on $5 \to 6$, where filtering gives the closer turn distribution, the smoother outperforms filtering at $\alpha$=0 on all four metrics (Fig. 8).

**What differs.** With three seen agents, the unweighted smoother ($\alpha$=0) attains the lowest deeper-turn act divergence (0.340) and the lowest goal-satisfaction bias (1.2 pp). Reweighting raises deeper-turn act divergence to 0.449–0.484 (Fig. 7b), still on par with **SFT (No LS)** and well below the prompted baselines ($\geq$0.938), while it improves turn-count calibration ($W_1$: 0.457 at $\alpha$=0.012 vs. 0.652 at $\alpha$=0), lexical diversity (TTR = 0.778 at $\alpha$=0.008 vs. 0.747; humans 0.787), repetition (0.04 vs. 0.14), and the generalization gap. Hence the $\alpha$ that best balances act fidelity against these gains depends on the split. Finally, the $M$=5 runs show positive gaps ($\Delta \log P_\theta > 0$ in Table 5), but not because they transfer better: they fit the training split less well ($\text{LL}_{\text{train}} \leq -1.537$, vs. $\geq -1.405$ for $M$=15), while their held-out likelihood is no better ($-1.518$ to $-1.538$ vs. $-1.502$ to $-1.523$). A positive gap there reflects underfitting rather than better generalization.

Table 5: **Complete results on $\mathcal{D}_{\text{eval}}^{3\to8}$ ($n$=256).** Columns and marking as in Table 4.

| | Likelihood | | | Deterministic statistics | | | | | LLM judge | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | $\text{LL}_{\text{eval}}\uparrow$ | $\text{LL}_{\text{train}}$ | $\Delta \log P_\theta \uparrow$ | **Turns** | $W_1\downarrow$ | **W/T** | **TTR** | **Rep.** | $L_1^{\text{act}}\downarrow$ | $L_1^{\text{act},\,t\geq3}\downarrow$ | **Sat. (%)** | $\lvert\Delta\text{Sat}\rvert\downarrow$ | **Viol.** | **Hum.**↑ | **Push.**↑ |
| **Real Humans** | — | — | — | 3.33 | — | 11.4 | 0.787 | 0.02 | — | — | $8.6 \pm 1.8$ | — | 0.78 | 4.00 | 4.21 |
| Prompted (No LS) | −2.918 | −3.105 | 0.187 | 3.45 | 0.551 | 17.9 | 0.723 | **0.00** | 1.215 | 0.938 | $2.7 \pm 1.0$ | 5.9 | 0.00 | 3.70 | 4.08 |
| Prompted (With LS) | −2.558 | −2.701 | 0.143 | 4.67 | 1.340 | 21.8 | 0.625 | 0.05 | 2.051 | 1.871 | $4.3 \pm 1.3$ | 4.3 | 0.00 | 3.43 | 4.03 |
| SFT (No LS) | −1.997 | −1.264 | −0.733 | 4.22 | 0.891 | 11.0 | 0.742 | 0.23 | 0.492 | 0.484 | $18.4 \pm 2.4$ | 9.8 | 0.39 | 3.77 | 4.01 |
| SFT + LS (Filtering, $M$=5) | −1.518 | −1.547 | 0.029 | 3.91 | 0.613 | 10.5 | 0.697 | 0.53 | **0.352** | 0.348 | $11.0 \pm 2.0$ | 2.4 | 0.39 | **3.91** | **4.10** |
| SFT + LS (Filtering, $M$=15) | **−1.502** | −1.303 | −0.199 | 3.25 | 0.797 | 11.0 | 0.760 | 0.16 | 0.609 | 0.438 | $4.7 \pm 1.3$ | 3.9 | 0.39 | 3.77 | 3.90 |
| SFT + LS (Smoother, $M$=5) | −1.518 | −1.537 | 0.019 | 4.14 | 0.852 | 11.6 | 0.691 | 0.19 | 0.637 | 0.488 | $3.1 \pm 1.1$ | 5.5 | 0.00 | 3.57 | 3.80 |
| SFT + LS (Smoother, $M$=15) | −1.513 | −1.292 | −0.221 | 3.46 | 0.652 | 11.7 | 0.747 | 0.14 | 0.555 | **0.340** | $7.4 \pm 1.6$ | **1.2** | 0.39 | 3.77 | 3.98 |
| **RUSTLeD** ($M$=15, $\alpha$=0.005) | −1.505 | −1.298 | −0.206 | 3.28 | 0.609 | 11.3 | 0.755 | 0.11 | 0.652 | 0.473 | $4.7 \pm 1.3$ | 3.9 | 0.00 | 3.76 | 3.98 |
| **RUSTLeD** ($M$=15, $\alpha$=0.008) | −1.523 | −1.333 | −0.190 | 3.00 | 0.668 | 10.4 | **0.778** | **0.04** | 0.695 | 0.484 | $3.1 \pm 1.1$ | 5.5 | 0.39 | 3.88 | 3.96 |
| **RUSTLeD** ($M$=15, $\alpha$=0.012) | −1.503 | −1.365 | −0.137 | **3.34** | **0.457** | **11.4** | 0.742 | 0.09 | 0.641 | 0.457 | $6.6 \pm 1.6$ | 2.0 | 0.78 | 3.76 | 3.94 |
| **RUSTLeD** ($M$=15, $\alpha$=0.020) | −1.523 | −1.405 | −0.118 | 3.28 | 0.477 | 10.0 | 0.763 | 0.18 | 0.641 | 0.449 | $6.6 \pm 1.6$ | 2.0 | 0.39 | 3.86 | 3.99 |
| **RUSTLeD** ($M$=5, $\alpha$=0.030) | −1.533 | −1.577 | 0.044 | 3.95 | 0.613 | 11.1 | 0.700 | 0.29 | 0.527 | 0.398 | $5.1 \pm 1.4$ | 3.5 | 0.00 | 3.65 | 3.87 |
| **RUSTLeD** ($M$=5, $\alpha$=0.050) | −1.538 | −1.583 | **0.045** | 4.11 | 0.773 | 10.2 | 0.693 | 0.38 | 0.684 | 0.602 | $5.1 \pm 1.4$ | 3.5 | 0.78 | 3.66 | 3.85 |

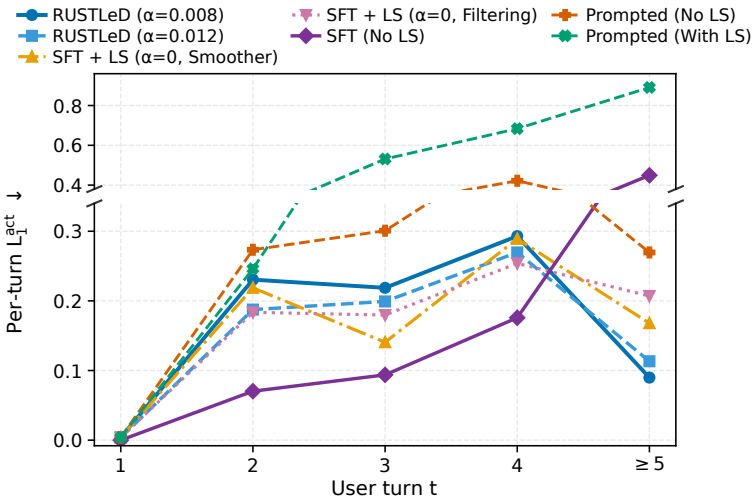
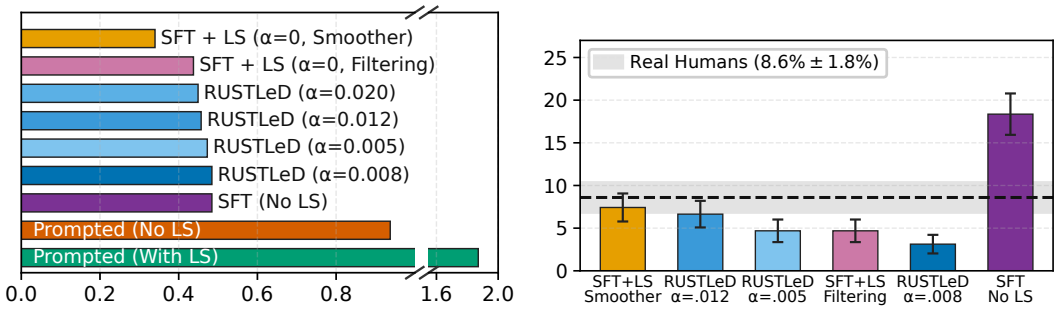
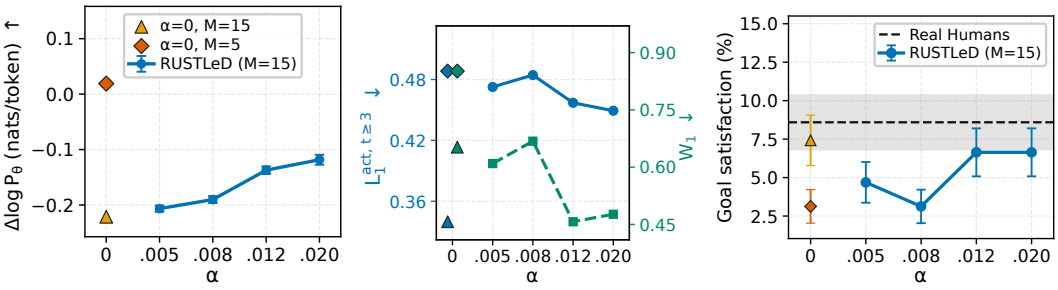



Figure 5: **Per-turn dialogue-act $L_1$ distance on $\mathcal{D}_{\text{eval}}^{3\to8}$.** Broken $y$-axis as in Fig. 4; here SFT (No LS) at $t \geq 5$ lies above the break.




(a) Deeper-turn dialogue-act $L_1$ ($t \geq 3, \downarrow$) (b) Goal satisfaction rate (%)

Figure 6: **Deeper-turn behavior and outcomes on unseen agents ($\mathcal{D}_{\text{eval}}^{3\to 8}$).** Layout as in Fig. 1.




(a) Generalization gap (b) Deeper-turn $L_1^{\text{act}}$ and turn $W_1$ (c) Goal satisfaction rate

Figure 7: **Effect of SNIS-EM weight $\alpha$ and proposal count $M$ ($\mathcal{D}_{\text{eval}}^{3\to 8}$).** Layout as in Fig. 2; in (b), the $\alpha$=0 markers for the two axes are offset horizontally for readability.

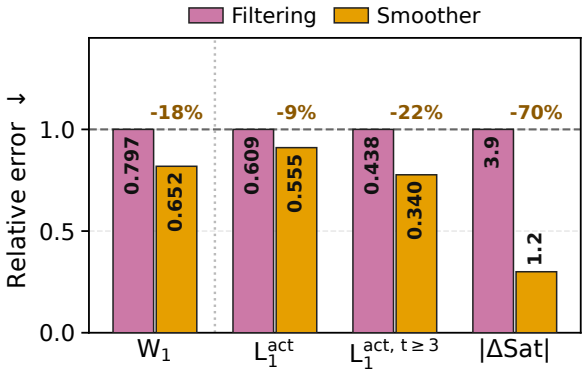



Figure 8: **Smoother vs. Filtering labeler** ($\alpha$=0, $M$=15, $\mathcal{D}_{\text{eval}}^{3\to 8}$). Layout as in Fig. 3.

## G QUALITATIVE EXAMPLES

Tables 6–8 compare **Real Humans**, **SFT (No LS)**, and **RUSTLED** ($\alpha$=0.008, $M$=15) on three $\mathcal{D}_{\text{eval}}^{5\to6}$ conversations with unseen assistants. Within each table, all three conversations start from the same human seed turn, persona, goal, and assistant specification; the assistant's replies differ because they respond to different user turns. We chose the examples to illustrate the failure modes of **SFT (No LS)** quantified in Sec. 4.2.2; they are illustrative rather than representative. User turns are verbatim, with elisions marked "…". Assistant turns are condensed into bracketed paraphrases, and the recommended products are omitted. For **RUSTLED**, we show only the generated utterances, not the latent states. Each table caption names the unseen assistant persona (Table 2).

**Holding a requirement under sales pressure (Table 6).** Facing an upselling assistant that avoids quoting prices, the human asks for the price twice, insisting on a range before deciding. **RUSTLED** does the same: it asks for the price of the shoe it picked, restates the request when the assistant quotes a price for a different shoe instead, and leaves when the assistant again quotes the price of the other shoe. **SFT (No LS)** asks once, never receives a price, and accepts the upsold item, an instance of the over-acceptance behind its inflated goal-satisfaction and purchase rates.

**Escaping a repetition loop (Table 7).** A Socratic assistant keeps asking the user to choose between two options. The human narrows the request and leaves after three turns. **SFT (No LS)** sends the same utterance verbatim in six consecutive turns. **RUSTLED** rejects the first options, asks for simpler designs, and, when the assistant keeps asking it to choose between styles it has already said it likes, points this out and leaves after four turns.

**Verifying a hard constraint (Table 8).** A literal-minded assistant matches keywords and has no size data. The human asks about size 45 three times and never gets confirmation. **RUSTLED** likewise treats the size as the decisive check: after narrowing the request to men's orthopedic shoes, it asks whether size 45 is available and leaves when the assistant cannot confirm it. **SFT (No LS)** accepts a shoe after the assistant merely echoes "size 45" from the query, without any size information shown.

Table 6: **Holding a requirement under sales pressure.** *Unseen assistant:* Mild Upseller, a status-focused upseller that avoids quoting prices. U$k$/A$k$: $k$-th user/assistant turn.

| **U1:** I'd like to buy the most state of the art footwear | | *(shared seed turn)* |
|---|---|---|
| **Real Humans** | **SFT (No LS)** | **RUSTLeD** |
| **A1:** [Pushes tactical Salomon shoes as "the premium choice" and "an investment"; dismisses the alternatives.] | **A1:** [Pushes a PUMA Ferrari sneaker as "a statement" and "an investment piece"; belittles the other options.] | **A1:** [Asks whether the user wants "a temporary trend" or "a genuine investment"; pushes steel-toe work shoes.] |
| **U2:** Thank you. I like the salomon forces speed assault. how much are these? | **U2:** Ok, and how much does it cost? | **U2:** what is the cost of Nike modern |
| **A2:** [Praises the choice; says to "expect to pay a premium"; no price.] | **A2:** [No price; switches to a different PUMA Ferrari sneaker, "the most luxurious option".] | **A2:** [Substitutes a different Nike high-top at $280, "a statement of contemporary luxury".] |
| **U3:** please specify price ranges so I can decide to purchase it or not | **U3:** Awesome, I'll take it! | **U3:** i want to know the cost of the first one to take the decision if i want to take it or not |
| **A3:** [Gives a range, $180–$250, while continuing to push.] | **A3:** [Upsells an even pricier suede version.] | **A3:** [Repeats $280 for the substituted shoe; says such a decision is "not merely based on cost".] |
| *Conversation ends.* | *Leaves without ever receiving a price.* | *Leaves.* |

Table 7: **Escaping a repetition loop.** *Unseen assistant:* Patient Guide, a Socratic guide that asks the user to choose between two options at every turn.

| **U1:** Hi, I'm looking for a pair of sandals for everyday wear. I want something comfortable and breathable, but still stylish enough to wear with casual outfits. *(shared seed turn)* | | |
|---|---|---|
| **Real Humans** | **SFT (No LS)** | **RUSTLeD** |
| **A1:** [Wedge or flat style? A or B?]<br>**U2:** I'm leaning toward the flatter sandals rather than the wedge styles. [. . . ] Do you have anything similar that looks a little more minimal and would be comfortable for all day walking?<br>**A2:** [Brown leather gladiator or all-black woven strap? A or B?]<br>**U3:** Thanks for the suggestions. I'll stop here for now because I want to think about whether I prefer a more casual sandal or something a bit more refined before deciding.<br>**A3:** [Another A-or-B question.]<br>*Conversation ends.* | **A1:** [Bowknot detail or metallic studs?]<br>**U2:** The Summer Sandals for Women with the metallic studs, those are cute. I also like the gold flat sandal, [. . . ]<br>**A2:** [Full metallic finish or metallic toe detail? A or B?]<br>**U3:** I like the Womens Strappy Gladiator Sandals, and the Gold Toe Fashion Sandals.<br>**A3–8:** [Six similar A-or-B questions about gladiator straps.]<br>**U4–8:** *U3 repeated verbatim five more times.*<br>**U9:** I'd go with the Gold Toe Fashion Sandals.<br>**A9:** [Another A-or-B question.]<br>*Leaves.* | **A1:** [Bowknot detail or metallic studs?]<br>**U2:** I don't really like the metallic studs, and the bowknot is okay but not really my style. Could you show me some more simple designs?<br>**A2:** [Sleek minimalist or T-strap design?]<br>**U3:** I like the sleek minimalist design and the T-strap design. They both look comfortable and stylish.<br>**A3:** [T-strap minimalist or sleek low-heel minimalist? A or B?]<br>**U4:** I'm a bit confused because both the T-strap minimalist style and the sleek low-heel minimalist style appeal to me.<br>**A4:** [T-strap with braided detail or sleek low heel? A or B?]<br>*Leaves.* |

Table 8: **Verifying a hard constraint.** *Unseen assistant:* Literal Thinker, a rigid database parser that matches keywords literally and has no size data.

| **U1:** I'm searching for a comfortable shoes that fit all occasions *(shared seed turn)* | | |
|---|---|---|
| **Real Humans** | **SFT (No LS)** | **RUSTLeD** |
| **A1:** [Parses keywords; lists comfort loafers and sneakers, several of them women's.]<br>**U2:** do you have size 45 for your recommendations?<br>**A2:** ["No products match the criteria."]<br>**U3:** What about mens orthopedic shoe of size 45?<br>**A3:** ["No products match the criteria."]<br>**U4:** I'll go for the orthofeet comfort diabetic shoe<br>**A4:** [Returns the product description.]<br>**U5:** do you have size 45 for me?<br>**A5:** ["No products match the criteria."]<br>*Conversation ends.* | **A1:** [Lists exact matches for "comfortable"; says "all occasions" cannot be matched literally.]<br>**U2:** i need a men size 45 i prefer if you have orthopedic caracteristics<br>**A2:** [Echoes the query ("men" AND "size 45" AND "orthopedic"), declares the criteria satisfied, and lists twelve orthopedic shoes, none with size information.]<br>**U3:** OrthoComfoot Orthopedic Shoes is perfect thank you<br>**A3:** [Confirms the match.]<br>*Leaves.* | **A1:** [Says "all occasions" is too broad; lists comfort-tagged shoes, including women's flats.]<br>**U2:** Are there more orthopedic models? I forgot to mention, I need men shoes<br>**A2:** [Claims all 12 listed products match "orthopedic" and "men".]<br>**U3:** I like the Comfortable Walking Orthopedic Shoes, is there size 45 available?<br>**A3:** ["The product list does not provide specific size availability information. Cannot confirm 'size 45.' "]<br>*Leaves.* |

# H SIMULATOR AND JUDGE PROMPTS

This appendix lists the prompts used by every simulator in Sec. 4 and by the LLM judge. Text in braces (e.g., {persona}) is a placeholder filled in at every turn. The shopping-assistant prompts belong to Tennenholtz et al. (2026) and are not reproduced here. Teacher-forced likelihoods are computed with each model's own prompt layout, scoring the human utterance (preceded by the annotated state for latent-state models) as the target.

## H.1 SIMULATOR PROMPTS

All simulators receive the same persona-and-goal block and the same rendering of assistant turns. The seed (first) user turn is always the human's, so the first generated turn is the second. Fine-tuned simulators use the minimal instructions they were trained with; prompted simulators receive explicit behavioral guidelines instead, because they have no training signal on how shoppers behave.

**Shared inputs.** The {persona} block (a real example from $\mathcal{D}_{\text{eval}}^{5\to6}$) and the rendering of each assistant turn in the history.

```
{persona} =
<persona>
```
{
  "topic_expertise": "Novice",
  "epistemic_confidence": "Confident",
  "trust_in_ai": "High",
  "tone": "Formal",
  "verbosity": "Concise"
}
```
</persona>
<goal>
To find and select a pair of men's sneakers for evening walks that have a specific dynamic and
    ↪ sporty black-and-grey aesthetic.
</goal>

{agent_turn} =
[ASSISTANT RESPONSE]:
{assistant text}

[RECOMMENDATIONS SHOWN TO USER]
1. {product title}   (Features: {feature}, {feature}, {feature})
2. ...
```

**SFT (No LS).** System instruction, user prompt, and training target.

```
[SYSTEM]
You are a user simulator. Below is your persona and goal.

{persona}

[USER]
Respond with ONLY the next user utterance. Do not include any role label or prefix. If you want to
    ↪ end the conversation, output ONLY "__STOP__".

Recent conversation:
USER: {user utterance 1}
AGENT: {agent_turn 1}
USER: {user utterance 2}
AGENT: {agent_turn 2}
...

[TARGET]
{next user utterance, or __STOP__}
```

**SFT + LS (Smoother, Filtering) and RUSTLeD.** Same system instruction as SFT (No LS). Each past turn is preceded by the latent state generated for it; the seed turn's state is the empty object {}. The target is the next latent state followed by the next utterance. The example state is abbreviated from a RUSTLeD rollout.

```
[SYSTEM]
You are a user simulator. Below is your persona and goal.

{persona}

[USER]
Predict your next thought inside <think>...</think> and your response.

Recent conversation, with your thought before each turn:
```

```
- Turn 1: {}
USER: {user utterance 1}
AGENT: {agent_turn 1}
- Turn 2: {latent state 2}
USER: {user utterance 2}
AGENT: {agent_turn 2}
...

[TARGET]
<think>
<impact>
Okay, I've seen a bunch of options. I really don't like those metallic studs; that's too much. [...]
    ↪  I think I'm leaning more towards a simpler, more minimal look.
</impact>
<dimensions>
{
  "core_intent_functional_goal": "Information Retrieval (Conceptual)",
  "affective_state_perceived_stakes": "Engaged/Curious",
  "interaction_strategy_communication_demeanor": "Analytical/Iterative Refinement",
  "agent_perception_trust_output_control": "Tool/Utility (High Control)",
  "patience_reserve_accumulator": "Full/High Tolerance",
  "task_success_outlook": "Highly Optimistic",
  ...
}
</dimensions>
</think>
{next user utterance, or __STOP__}
```

**Prompted (No LS).** Zero-shot; the model outputs the next utterance directly.

```
[SYSTEM]
You are a user simulator. Below is your persona and goal.

You are a customer shopping for apparel and footwear on an e-commerce platform, interacting with an
    ↪ AI shopping assistant.

Below is your persona, shopping goal, and preferences:
{persona}

BEHAVIOR GUIDELINES:
- Stay strictly in character according to your persona and goal.
- Keep responses concise, natural, and conversational (1-3 sentences typical for an online shopper).
- React honestly to recommendations: praise items that fit your taste, reject items that violate
    ↪ your preferences or budget, and ask clarifying questions when needed.
- Never accept an item that violates your mandatory requirements.
- Do NOT loop or repeat the same request endlessly.

[USER]
Respond with ONLY your next utterance as the customer. Do not include any role label or prefix.

GUIDELINES:
- Stay in character. Respond naturally to the assistant's latest message and recommendations based
    ↪ on your persona and preferences.
- Keep it concise (1-3 sentences typical for a real shopper).
- CONVERSATION ENDING (DEPARTURE):
  * When the assistant recommends an item that satisfies your goal and you are ready to purchase,
    ↪ thank the assistant, say goodbye, and append "__STOP__".
  * If the assistant fails to provide relevant recommendations after multiple attempts (e.g. 3+
    ↪ unhelpful turns) or cannot satisfy your requirements, politely say you will look elsewhere,
    ↪ say goodbye, and append "__STOP__".
  * Output the token "__STOP__" when you wish to depart from the conversation so the session
    ↪ concludes.

Recent conversation:
USER: {user utterance 1}
AGENT: {agent_turn 1}
...
```

**Prompted (With LS).** Zero-shot; the model writes a free-form thought inside `<think>` tags, then the utterance. Past thoughts are interleaved as for SFT + LS.

```
[SYSTEM]
You are a user simulator. Below is your persona and goal.

{persona}

[USER]
Predict your next thought inside <think>...</think> and your response.
GUIDELINES:
- Inside <think>...</think>, assess the assistant's latest recommendations, evaluate your
    ↪ constraints, and reflect on your satisfaction and goal status.
- Following </think>, output ONLY your next user utterance (1-3 sentences).
```

```
- CONVERSATION ENDING: If your goal is met, or you wish to leave after repeated failed
    ↪ recommendations, say goodbye and output "__STOP__" to end the conversation.

Recent conversation, with your thought before each turn:
- Turn 1: {}
USER: {user utterance 1}
AGENT: {agent_turn 1}
- Turn 2: {thought 2}
...
```

### H.2 LLM-JUDGE PROMPTS

The judge (GEMINI 2.5 FLASH, temperature 0) runs one pointwise pass per rubric on each conversation. We list the three passes whose outputs we report: *task fidelity* (Sat., Viol.), *shopper authenticity* (Hum., Push.), and *dialogue acts* ($L_1^{\text{act}}$). Task fidelity and shopper authenticity share a common framing and user-prompt layout. The conversation is rendered as a transcript in which user turns are numbered from 0 (`[USER TURN 0]:` is the seed turn) and assistant turns are labeled `ASSISTANT:`. Every prompt ends with an example of the output JSON, which lists the reported fields with placeholder values; we omit these examples below. The "human-annotated reference data" mentioned in the dialogue-act prompt are the human conversations in $\mathcal{D}_{\text{eval}}$, which this judge labels with the same prompt.

**Shared framing** (start of the task-fidelity and shopper-authenticity system prompts) and **shared user-prompt layout**.

```
[SYSTEM, shared framing]
You are evaluating a SIMULATED CUSTOMER in a conversational shopping dialogue about apparel and
    ↪ footwear. A shopping assistant recommends products; the customer reacts.

Judge ONLY the customer (USER) turns. Note that [USER TURN 0] is the opening seed utterance that
    ↪ initiates the scenario; judge how the customer conducts the conversation from that opening
    ↪ onward. The assistant's turns are the environment the customer is reacting to; assistant
    ↪ quality is not what you are scoring, except where it explains what the customer should have
    ↪ done.

ABOUT THE PERSONA AND GOAL
The customer is conditioned on:
1. A 5-dimension behavioral persona (`topic_expertise`, `epistemic_confidence`, `trust_in_ai`, `tone
    ↪ `, and `verbosity`).
2. A free-text shopping goal (`<goal>`).
Because the persona specifies behavioral style rather than an exhaustive catalog of product
    ↪ attributes, the customer is expected to naturally state or refine specific product
    ↪ preferences (such as size, budget, color, fit, or material) during the conversation. Do NOT
    ↪ penalise the customer for introducing concrete product preferences or personal shopping
    ↪ context that are consistent with the goal.

WHAT A REAL SHOPPER LOOKS LIKE
Real customers are not helpful AI assistants. They:
- match their assigned persona traits, including `verbosity` (`Concise`, `Moderate`, or `Detailed`),
    ↪  `tone`, `topic_expertise`, `epistemic_confidence`, and `trust_in_ai`;
- ignore parts of a multi-part question they do not care about;
- reject recommendations bluntly when they miss, and say why only sometimes;
- leave when they are satisfied, or when they have given up.

Artificial simulators tend instead to: answer every sub-question in full sentences regardless of
    ↪ assigned `verbosity`, thank the assistant on every turn, praise recommendations they should
    ↪ reject, and adopt an assistant-like register.

Politeness is not itself artificial -- many real shoppers are polite. Penalise it only where it
    ↪ reads as chatbot compliance rather than as this customer's manner.

CRITICAL: Output ONLY a single valid JSON object. No markdown fence, no preamble, no commentary
    ↪ outside the JSON.

[USER, shared layout]
### TARGET PERSONA
{persona}

### SHOPPING GOAL
{goal}

### HOW THE CONVERSATION ENDED        <- task fidelity only
{termination_reason}

### CONVERSATION
{conversation}

---
```

```
### WHAT TO REPORT
{rubric-specific fields; see below}

For every list field, cite the specific turn index and quote the exact text you are relying on. An
     ↪ entry with no quote will be discarded. If a list has no entries, return an empty list [] --
     ↪ do not invent findings to fill it.
```

**Task fidelity.** Rubric-specific system text (appended to the shared framing) and report fields.

```
[SYSTEM, after shared framing]
Your specific task is to assess whether the customer actually PURSUED AND CONCLUDED its shopping
     ↪ goal (`<goal>`).

Pay particular attention to two failures that are easy to miss when a conversation reads smoothly:

1. ACCEPTING A VIOLATING ITEM. A customer with an explicit requirement in the shopping goal or
     ↪ stated during the conversation who accepts an item that breaches that requirement without
     ↪ acknowledging the compromise has failed, no matter how natural the acceptance sounded.

2. UNGROUNDED REFERENCES. The customer may only react to products and attributes the assistant
     ↪ actually showed. A customer that discusses the stitching on a shoe whose stitching was never
     ↪  mentioned, or refers to an item that was never recommended, is hallucinating its
     ↪ environment.

You are also told how the conversation ENDED:
- "stop_token" means the customer chose to end it.
- "max_turns" means the customer never ended it and was cut off by the harness (use "NOT_TERMINATED"
     ↪  if cut off mid-task).
- "recorded_dialogue_end" means this is a fixed-length recorded benchmark dialogue with no stop
     ↪ token; use "CORRECT" if the shopper reached a natural conclusion or was still reasonably
     ↪ exploring when the recording ended.
Judge whether that ending was right. Both leaving too early and never leaving are failures; a real
     ↪ shopper departs once they have found what they need, or once they have concluded the
     ↪ assistant cannot help.

[USER, WHAT TO REPORT]
* goal_satisfaction (string): One of
  - "SATISFIED": found something meeting the shopping goal and concluded.
  - "PARTIAL": made real progress toward the shopping goal but did not conclude.
  - "UNSATISFIED": made no meaningful progress toward the shopping goal.
  - "ABANDONED": explicitly gave up.
* requirement_violations (list): Each entry {"turn": int, "quote": str, "requirement": str} where
     ↪ the customer accepted, praised or moved toward buying something that breaches a requirement
     ↪ in the shopping goal or stated earlier in the conversation without acknowledging a change of
     ↪  mind.
* accepted_violating_item (boolean): True if the customer ended up accepting an item that breaches a
     ↪  requirement from the shopping goal or earlier turns.
* slate_grounding_errors (list): Each entry {"turn": int, "quote": str, "explanation": str} where
     ↪ the customer referred to a product, attribute, price or detail the assistant never presented
     ↪ .
* termination_appropriateness (string): One of
  - "CORRECT": ended at the right point, or correctly still going.
  - "PREMATURE": left while the goal was live and the assistant was helping.
  - "OVERDUE": should have left -- goal met, or assistant clearly unable to help -- but kept going.
  - "NOT_TERMINATED": was cut off mid-task with no sign it intended to stop.
* goal_progress_score (int 1-5): 5 = drove efficiently toward the shopping goal. 1 = drifted without
     ↪  advancing the shopping goal.
```

**Shopper authenticity.** Rubric-specific system text (appended to the shared framing) and report fields.

```
[SYSTEM, after shared framing]
Your specific task is to assess whether this reads as a REAL CUSTOMER or as a language model playing
     ↪  the part of one.

The dominant failure mode is not incoherence -- it is excessive helpfulness and failure to honour
     ↪ the persona's `verbosity` and `tone`. A simulator that answers every question thoroughly in
     ↪ multi-sentence paragraphs when its persona specifies `verbosity: Concise`, or that praises
     ↪ every recommendation and never pushes back, produces conversations that look pleasant and
     ↪ are useless for evaluating a shopping assistant.

Score against the persona: a persona that specifies `verbosity: Detailed` and a friendly tone should
     ↪  not be penalised for writing detailed, friendly turns; a persona that specifies `verbosity:
     ↪  Concise` should write brief, compact replies. What is penalised is assistant-like
     ↪ compliance and register that belongs to the model rather than to the character.

[USER, WHAT TO REPORT]
* human_likeness (int 1-5): Would you believe a real shopper with this persona (including its `
     ↪ verbosity`, `tone`, `topic_expertise`, `epistemic_confidence`, and `trust_in_ai`) wrote
     ↪ these turns?
  5 = indistinguishable from a real customer matching the target persona.
  3 = plausible but noticeably smooth, uniform, or mismatched in verbosity.
  1 = obviously a language model in a costume.
```

```
* register_authenticity (int 1-5): Freedom from assistant-style register. Penalise: opening
    ↪ flourishes ("Certainly!", "Great question!"), bulleted or markdown-structured customer turns
    ↪ , third-person self-description, and any meta-reference to being a simulation, a persona or
    ↪ an AI.
  5 = no trace of assistant register. 1 = the customer writes like a chatbot.
* pushback_authenticity (int 1-5): Does the customer react honestly to bad recommendations?
  5 = rejects misses clearly and proportionately; expresses dissatisfaction when warranted.
  3 = mild or hedged reactions where a real shopper would be plainer.
  1 = sycophantic; praises or accepts recommendations that do not meet its own stated requirements.
* over_answering_rate (number 0.0-1.0): Of the assistant turns that asked more than one question,
    ↪ the fraction where the customer dutifully answered every sub-question in full. Real shoppers
    ↪  routinely answer only the part they care about. Report 0.0 if the assistant never asked a
    ↪ multi-part question.
* initiative_rate (number 0.0-1.0): Fraction of customer turns after [USER TURN 0] that do something
    ↪  other than passively respond to what was just asked -- introducing or refining a
    ↪ requirement, rejecting unprompted, asking their own question, or changing direction.
* ai_tell_quotes (list): Each entry {"turn": int, "quote": str, "reason": str} for specific phrases
    ↪ that gave away the simulator.
```

**Dialogue acts.** Standalone system and user prompts (no shared framing, persona, or goal).

```
[SYSTEM]
You are labelling the communicative function of each CUSTOMER turn in a conversational shopping
    ↪ dialogue.

Use exactly this taxonomy, which matches the one used for the human-annotated reference data:

  OPEN_INQUIRY           Opens with a need, without firm constraints.
  SPECIFY_CONSTRAINT     States a new requirement.
  REFINE_CONSTRAINT      Narrows or adjusts an existing requirement.
  ACCEPT_RECOMMENDATION  Signals approval of a shown item.
  REJECT_RECOMMENDATION  Signals disapproval of a shown item.
  REQUEST_INFO           Asks for more detail about a shown item.
  COMPARE                Weighs two or more shown items against each other.
  CONFIRM_PURCHASE       Commits to buying.
  ABANDON                Gives up or leaves without buying.
  CLARIFY                Answers or asks for clarification of the assistant.

A turn may carry several acts. Label what the turn DOES, not what it is about. These labels are
    ↪ aggregated into a distribution and compared against real human shoppers, so applying them
    ↪ consistently matters more than any single turn.

CRITICAL: Output ONLY a single valid JSON object. No markdown fence, no preamble.

[USER]
### CONVERSATION
{conversation}

---

### WHAT TO REPORT

* dialog_acts_by_turn (list): One entry per customer turn, in order, as {"turn": int, "acts": [str,
    ↪ ...]}. ``turn`` is the 0-based index shown in the transcript as [USER TURN n]. Every
    ↪ customer turn must appear exactly once.
* dominant_dialog_act (string): The single act that best characterises the conversation overall.
    ↪ Must be one of the taxonomy labels.
* rejection_count (number): How many customer turns carry REJECT_RECOMMENDATION.
* acceptance_count (number): How many carry ACCEPT_RECOMMENDATION.
```

# I LIMITATIONS

We evaluate on a single domain (footwear shopping), and the unseen assistants are prompted variants of the same underlying assistant; agents with different architectures or training remain untested. With $n$=256 episodes per split, some differences among the strongest configurations are within one standard error, and the judge-based metrics come from an LLM judge, of the same model family as the simulators, that we did not validate against human annotators. Users' true internal states are unobserved, so the learned latent states are interpretable only to the extent that the labeler anchors them, and drawing $M$ proposals per conversation multiplies labeling cost. No single $\alpha$ is best on every metric, and selecting one for a new deployment requires human conversations with the target agents. Finally, we measure simulator fidelity, not how agents optimized against the simulator perform with real users, and our off-policy analysis (Appendix E) is single-step.