
Beyond Cooperative Simulators: Generating Realistic User Personas for Robust Evaluation of LLM Agents

Harshita Chopra^{†,α} Kshitish Ghate^{†,α}

Aylin Caliskan^α Tadayoshi Kohno^β Chirag Shah^α Natasha Jaques^α

^α University of Washington, Seattle, WA ^β Georgetown University, Washington, DC

 [Persona-Policies](#)

Abstract

Large Language Model (LLM) agents are increasingly deployed in settings where they interact with a wide variety of people, including users who are unclear, impatient, or reluctant to share information. However, collecting real interaction data at scale remains expensive. The field has turned to LLM-based *user simulators* as stand-ins, but these simulators inherit the behavior of their underlying models: cooperative and homogeneous. As a result, agents that appear strong in simulation often fail under the unseen, diverse communication patterns of real users. To narrow this gap, we introduce **Persona Policies** (PPoL), a plug-and-play control layer that induces realistic behavioral variation in user simulators while preserving the original task goals. Rather than hand-crafting personas, we cast persona generation as an LLM-driven evolutionary program search that optimizes a Python generator to discover behaviors and translate them into task-preserving roleplay *policies*. Candidate generators are guided by a multi-objective fitness score combining human-likeness with broad coverage of human behavioral patterns. Once optimized, the generator produces a diverse population of human-like personas for any task in the domain. Across τ^2 -bench retail and airline domains, evolved PPoL programs yield 33–62% absolute gains in fitness score over the baseline simulator. In a blinded evaluation, annotators rated PPoL-conditioned users as human 80.4% of the time, close to real human traces and nearly twice as frequently as baseline simulators. Agents trained with PPoL are more robust to challenging, out-of-distribution behaviors, improving task success by +17% relative to training only on existing simulated interactions. This offers a novel approach to strengthen simulator-based evaluation and training without changing tasks or rewards.

1 Introduction

The promise of LLM-based assistants lies in their ability to act as autonomous agents, handling complex, multi-turn tasks like providing technical support or making reservations. However, to be truly useful, an agent must do more than navigate a software interface; it must navigate the unpredictable nature of human communication. Real users rarely provide information perfectly [6, 14]. They have distinct preferences, varying patience, and often express their needs through ambiguous, fragmented, or sometimes adversarial language [7]. To be trustworthy, agents must adequately infer intent, adapt to individual style, and ensure no irrevocable actions are taken without clear alignment.

[†]Primary contributors. Email: {hchopra3, kghate}@cs.washington.edu

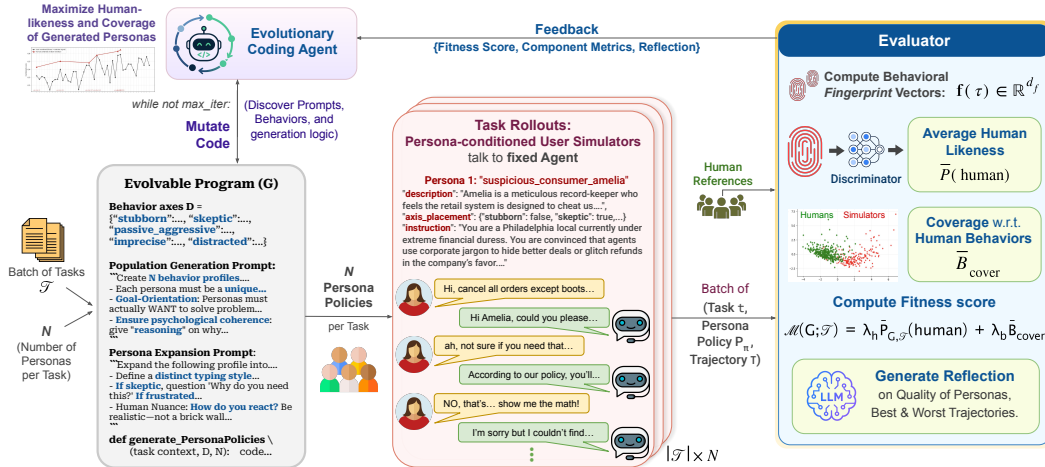


Figure 1: Overview of the **Persona Policies (PPoL)** program-evolution loop. Given a task, program G generates N persona policies that vary in how simulated users communicate. Candidate programs are scored via task rollouts for human-likeness and behavioral coverage, while reflection feedback judges the quality of trajectories to guide refinement of the axes, sampling rules, and prompts. Once optimized, the generator can produce diverse persona policies for new tasks in that domain.

Building agents with this level of social and procedural robustness requires evaluation environments that replicate the friction of real-world service encounters. The field has therefore turned to simulation-based benchmarks, which pair agents with “user simulators”: LLMs prompted to act as users or customers with fixed goals across domains like retail, banking, and aviation [1]. However, recent studies show that these default user simulators are overly cooperative, perfectly consistent, and highly forthcoming with information [3]. Real users often withhold details until prompted, push back on incorrect assumptions, use ambiguous language, and vary in patience and cooperativeness. This creates a *behavioral gap*: agents can appear strong against cooperative simulators but fail when confronted with the frustration, skepticism, or brevity of an actual human user [28].

Recent studies show that steering simulators with hand-specified traits like impatience or skepticism [5, 22] creates a valuable stress test that lowers agent success rates. However, while “richer synthetic users” impact performance, they are typically hand-specified or sampled from fixed, manual designs rather than being grounded in the actual, complex distribution of real human behaviors.

To address this, we introduce **Persona Policies (PPoL)**, a plug-and-play control layer that diversifies user simulation by generating multiple human-like personas while keeping benchmark tasks fixed. Each ‘policy’ is a short set of *additional* instructions added to the simulator prompt that controls *how* the user communicates, for example, by varying tone, pacing, selective disclosure, etc. This lets the same task be tested under many realistic communication patterns, so benchmark performance better reflects robustness to real users rather than success on one cooperative simulator style.

Learning these policies is framed as *evolutionary program search*. Rather than handcrafting prompts, we implement a Python persona generator that generates a population of behavioral profiles and expands each profile into task-preserving roleplay instructions. Using OpenEvolve [21], a frontier LLM iteratively mutates the generator’s code. We evaluate candidate generators through actual agent–user rollouts, scoring the personas on two objectives: **human-likeness**, measured as the probability of being classified as human by a trained discriminator, and **behavioral coverage**, which measures how well the N personas span the human distribution. Both metrics are computed using *behavioral fingerprints*, 19 lexical and interaction-level features organized across communication style, information disclosure, clarification behavior, and error reaction [28]. A MAP-Elites archive [13] over these objectives keeps the population diverse rather than converging to a single behavioral type.

Beyond evaluation, evolved personas offer a powerful training signal. Agents fine-tuned on these varied interactions improve downstream robustness, suggesting that persona-augmented simulation can serve as domain randomization for language agents [24]. In this work, we make three contributions:

1. We design PPoL as a plug-and-play control layer that injects realistic behavioral diversity into standard user simulators without changing the underlying tasks.
2. We formulate persona generation as an automated framework using evolutionary search to discover diverse, human-like behaviors without manual intervention.
3. We validate PPoL end-to-end: evolved personas narrow the simulator–human gap, appear human-like in blinded evaluations, and improve agent robustness when used for training.

2 Related Work

User simulation for interactive agents. Recent agent benchmarks increasingly evaluate models in multi-turn, tool-mediated environments rather than static prompts. MINT foregrounds iterative tool use and natural-language feedback [26], τ -bench formalizes policy-constrained customer-service interaction with verifiable end-state rewards, τ^2 -bench extends this to dual-control settings where users also act on the shared environment [1], and ToolSandbox adds stateful tools and on-policy conversational rollouts [10]. Similarly, a recent study shows that even strong models become substantially less reliable when information is revealed incrementally across turns, highlighting the challenge inherent in this type of multi-turn assistance task [9].

Personas and behavioral control. Prior work on user simulation has focused on making simulated interactions more coherent, natural, or goal-consistent. LLM-based simulators improve task-oriented dialogue by fine-tuning on real conversations, adding verifier models, or learning more human-like questioning patterns [19, 11, 8]. A parallel line of persona-based work uses profiles to diversify generated users and responses, from profile-grounded dialogue [27] to implicit user profiles inferred from human–machine conversations [25] and large-scale persona-driven synthetic data generation [4]. These approaches make simulated users less generic, but they usually treat personas as fixed descriptions, sampled profiles, or training data artifacts. Recent work [16] introduced an approach to automatically evolve persona generators to maximize population coverage. Their focus, however, is diversity in static preference responses and general scenarios, not task-grounded interaction. We extend this concept of scalable persona generation toward multi-turn agent evaluation: given a fixed task, PPoL generates a population of users who all pursue the same goal but communicate in different ways. The generator is optimized from agent–user rollouts for two properties: whether the resulting conversations resemble real human interactions, and whether the generated population covers diverse regions of the human behavior distribution.

Robustness under diverse users. Evaluation scores on interactive benchmarks reflect *both* the agent and the simulated user who conversed. Recent meta-evaluations show that switching simulators can reorder model rankings and diverge from human judgments [3, 20, 28]. Separately, stress-test studies show that small, controlled changes to user behavior, such as prompting more impatient or less cooperative play, can move headline success rates by large margins [5, 22]. Together, these findings argue for making the simulated user an explicit, tunable part of the benchmark, rather than a hidden default baked into the evaluation. To this end, the PPoL framework discovers user-side interaction behaviors and keeps selection grounded in signals computed from real human traces.

3 Method

Persona Policies (PPoL) automate the creation of a domain-level program that generates multiple task-preserving personas for any given user scenario. We first state the setting and notation, then describe the generator, fitness score, evolutionary search, and implementation.

3.1 Problem Setting

We consider multi-turn, goal-directed dialogue benchmarks. Each task t specifies a scenario and the user’s objectives. Let c_t be the benchmark-provided user context passed to the existing user simulator: base persona, task instructions, and any facts available to the user. We pass this context to the persona generator as-is. An LLM *user simulator* is prompted with the task and converses with an *agent*. All task-level quantities are held fixed: goals, private facts, environment state, and the success determined

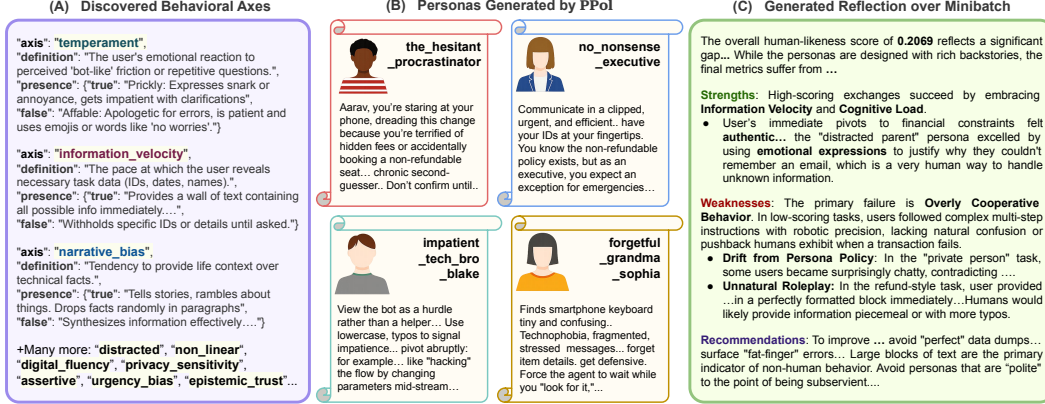


Figure 2: Samples of (A) Discovered Behavioral Axes (drives diversity in population), (B) Personas generated by PPol, (C) Reflection over a minibatch (provided as feedback to mutate program).

by task-completion rules. The sole controllable input is natural-language text appended to the user simulator's *system* prompt.

Persona policy P_π refers to appended natural-language text. It controls the simulated user's communicative style, including tone, pacing, and disclosure, while preserving objectives and private knowledge provided in $s_{\text{base}}(t)$, the benchmark's default system prompt for task t . The user-side system prompt becomes $s_{\text{base}}(t) \oplus P_\pi$, where \oplus denotes string concatenation.

Behavioral variation is represented by a list D of **behavioral axes**. Each entry names an axis, gives a short definition, and provides paired on/off *playbooks* for how the behavior appears when active or inactive. See Figure 2(A) for example. We seed D with four behaviors: terseness, skepticism, frustration, and ambiguity. The list D is maintained in the evolvable generator source file, so mutation may add, remove, or refine axes in addition to the instructions.

At a high level, the procedure has four steps: generate a set of persona policies for each benchmark task, run agent-user interaction rollouts, score the resulting trajectories against human reference behavior, and use an evolutionary coding agent to mutate the generator source.

3.2 Persona generation

The evolvable artifact is a Python program that implements the persona generator function G and holds the current axis list D . For task t , the generated persona-policy set is

$$\Pi_t = \left\{ P_{\pi,t}^{(i)} \right\}_{i=1}^N, \quad \Pi_t = G(c_t, D, N). \quad (1)$$

Each persona policy $P_{\pi,t}^{(i)}$ has a **persona record** $r_t^{(i)} = (\mathbf{a}_t^{(i)}, P_{\pi,t}^{(i)})$, $\mathbf{a}_t^{(i)} \in \{0, 1\}^{|D|}$. Here $\mathbf{a}_t^{(i)} : D \rightarrow \{0, 1\}$ is the binary mapping that determines which behavioral axes (e.g., *distracted*) are active (true) for persona i on task t .

Population generation. The **first phase** is *population generation*: a frontier language model, conditioned on the scenario c_t and full specification of D , jointly proposes N *population members*. The structured response lists exactly N members, each with a short natural-language summary, brief rationale, and axis assignment in $\{0, 1\}^{|D|}$. Because assignments are chosen jointly, this phase drives **diversity** in the discrete behavior space before any long-form user-facing policy is written.

Persona expansion. The **second phase** *expands* each population member, independently and in parallel, into one long natural-language persona policy $P_{\pi,t}^{(i)}$, the string appended to $s_{\text{base}}(t)$ for that rollout. Each call is conditioned on c_t , $\mathbf{a}_t^{(i)}$, and the active-axis playbooks.

3.3 Optimization Metrics

We score a candidate generator G by running it and evaluating the resulting dialogue trajectories. For each task t in a minibatch \mathcal{T} , we generate $\Pi_t = G(c_t, D, N)$ and run a full rollout for each policy $P_{\pi,t}^{(i)} \in \Pi_t$, with the persona-conditioned user simulator talking to the fixed agent. This gives one completed trajectory per (t, i) pair. We convert each trajectory into a behavioral fingerprint, explained below, and score the minibatch along two axes. First, **human-likeness** is the average probability that simulated trajectories resemble real user trajectories in the defined feature space. Second, **behavioral coverage**, a reward encouraging the generated personas to spread over the human-like behavior distribution rather than collapsing to one stereotyped style.

Human reference and behavioral fingerprints. We compare LLM simulators to real users on multi-turn tasks along four axes: communication style (D1), information disclosure (D2), clarification behavior (D3), and error reaction (D4), following [28]. We create a vector with $d_f = 19$ scalar features computed only from *user* turns in a completed trajectory τ . Each feature is a rate, count, or normalized statistic derived from regular-expression patterns over user messages (politeness, uncertainty, pushback, etc.) and simple turn statistics (length, variability, repetition) utilizing the LIWC2015 [17] and NRC [12] lexicons. Examples include *words per turn* and *short-utterance rate* (D1), *front-loading ratio* of identifying information (D2), *clarification-question rate* and *pushback rate* (D3), and *emotional-expression rate* (D4). A full list of the 19 features is provided in Appendix D. Stacking these scalars yields the episode’s *behavioral fingerprint*, $\mathbf{f}(\tau) \in \mathbb{R}^{d_f}$. We use a fixed calibration corpus of human conversations, \mathcal{H} , to compute the elementwise mean fingerprint $\mu_{\mathcal{H}} \in \mathbb{R}^{d_f}$ and build the discriminator training set.

Human-likeness via a learned discriminator. We train a lightweight binary classifier (a Random Forest, see Appendix E for model details and feature importances) to separate fingerprint vectors from real human dialogues in \mathcal{H} (label *human*) and fingerprint vectors collected under the LLM-based default user simulator on the same benchmark configuration (label *simulator*). Inputs are standardized before fitting. At evaluation time, for an episode e with fingerprint \mathbf{f}_e , the model outputs an estimated probability $p_{\text{RF}}(\text{human} \mid \mathbf{f}_e)$ that the trajectory looks human-like in this feature space. For a generator G evaluated on task minibatch \mathcal{T} , let $\mathcal{B}(G; \mathcal{T})$ be the resulting set of rollout episodes. We aggregate over this episode batch, which is our scalar signal for human-likeness:

$$\bar{P}_{G,\mathcal{T}}(\text{human}) = \frac{1}{|\mathcal{B}(G; \mathcal{T})|} \sum_{e \in \mathcal{B}(G; \mathcal{T})} p_{\text{RF}}(\text{human} \mid \mathbf{f}_e) \quad (2)$$

Behavioral coverage. Fix a task t and let $\mathcal{F}_t(G) = \{\mathbf{f}^{(1)}, \dots, \mathbf{f}^{(N)}\}$ be the episode fingerprint set induced by the N personas produced by G for that task, with each $\mathbf{f}^{(i)} \in \mathbb{R}^{d_f}$. Let $\mathcal{H}_{\text{train}}$ denote fingerprint vectors from a *train-split* subsample of real human dialogues in the target domain, held separate from any test-only use. In \mathbb{R}^{d_f} with Euclidean distance, we view $\mathcal{F}_t(G)$ and $\mathcal{H}_{\text{train}}$ as finite point clouds (for visualization, see Figure 4). The two-sided Chamfer error is

$$\text{err}(\mathcal{F}_t(G), \mathcal{H}_{\text{train}}) = \frac{1}{|\mathcal{H}_{\text{train}}|} \sum_{h \in \mathcal{H}_{\text{train}}} \min_{\mathbf{f} \in \mathcal{F}_t(G)} \|h - \mathbf{f}\|_2 + \frac{1}{|\mathcal{F}_t(G)|} \sum_{\mathbf{f} \in \mathcal{F}_t(G)} \min_{h \in \mathcal{H}_{\text{train}}} \|\mathbf{f} - h\|_2.$$

To put this on a $[0, 1]$ scale comparable to human-likeness, let d_{ref} be the mean pairwise distance among points in $\mathcal{H}_{\text{train}}$, i.e., a scale for how spread out real users are in fingerprint space. For $N \geq 1$, the per-task *coverage* score is

$$B_{\text{cover}}(\mathcal{F}_t(G), \mathcal{H}_{\text{train}}) = \max \left\{ 0, 1 - \min \left(1, \frac{\text{err}(\mathcal{F}_t(G), \mathcal{H}_{\text{train}})}{2 d_{\text{ref}}} \right) \right\}, \quad (3)$$

Intuitively, every human in the reference cloud should be near some simulated persona (coverage), and every persona should be near some human (realism), so persona fingerprints stay near the support of the human reference distribution. Averaging over tasks in the minibatch gives

$$\bar{B}_{\text{cover}}(G; \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} B_{\text{cover}}(\mathcal{F}_t(G), \mathcal{H}_{\text{train}})$$

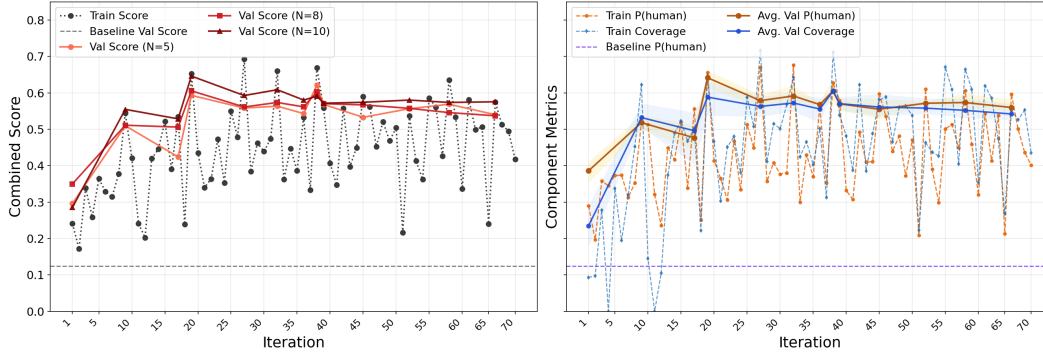


Figure 3: Monitoring Combined (Fitness) Score and Component Metrics during program evolution.

Combined score. Relying solely on a discriminator trained against default simulators risks adversarial drift, where the generator could produce unnatural behaviors that exploit the classifier’s blind spots. The behavioral coverage term (\bar{B}_{cover}) acts as a geometric regularizer against this: if the evolved personas drift off the human support in the fingerprint space, their Chamfer distance to the human reference cloud increases, heavily penalizing the combined fitness. The quantity that drives selection during evolution is the batch fitness

$$\mathcal{M}(G; \mathcal{T}) = \lambda_h \bar{P}_{G, \mathcal{T}}(\text{human}) + \lambda_b \bar{B}_{\text{cover}}(G; \mathcal{T}), \quad \lambda_h + \lambda_b = 1. \quad (4)$$

3.4 Evolutionary search

Search is implemented with OpenEvolve [21]: the Python program implementing G is the genotype, and a mutation language model proposes edits to prompts, the axis list D , and control flow. Each outer iteration samples a training minibatch \mathcal{T} using sliding windows over the task pool. We use a curriculum that increases the persona count N across epochs (e.g., $5 \rightarrow 8 \rightarrow 10$), so early iterations emphasize human-likeness under a smaller persona set while later iterations increase pressure on coverage. The coverage weight λ_b increases with the ratio of the current training N to the terminal N , and $\lambda_h = 1 - \lambda_b$. For the current program, we reload the module, run generation and expansion for every task in \mathcal{T} , execute benchmark rollouts for each (task, persona) pair, compute fingerprints, and evaluate Equations (2)–(4). The resulting fitness $\mathcal{M}(G; \mathcal{T})$ and the two-dimensional behavioral coordinates ($\bar{P}_{G, \mathcal{T}}(\text{human})$, $\bar{B}_{\text{cover}}(G; \mathcal{T})$) are returned to OpenEvolve; MAP-Elites [13] partitions programs into bins over these coordinates and retains the highest- \mathcal{M} program per bin, maintaining a diverse archive of behavioral strategies. We also evaluate a held-out validation slice of the training pool for monitoring.

Reflection-guided mutation. Beyond $\mathcal{M}(G; \mathcal{T})$, we construct a natural-language *reflection* artifact to guide the mutator. A separate reflection model analyzes fitness summaries (\mathcal{M} , mean human-likeness, mean coverage) alongside trajectory excerpts from the best and worst rollouts ranked by $p_{\text{RF}}(\text{human} \mid \mathbf{f})$. Each example includes the persona policy, the 19-D fingerprint vector, and the dialogue. The model returns a concise critique identifying which dialogue patterns plausibly drive high or low human-likeness (e.g., see Figure 2(C)). This reflection is passed to the OpenEvolve mutation step with its system templates, keeping the optimization anchored to the score in Eq.(4).

4 Experiments

We evaluate whether PPO \bar{l} improves simulator realism and agent robustness without changing benchmark tasks. First, we measure whether evolved personas produce trajectories that are closer to human conversations than other baselines. We then validate the behavioral fingerprint-based scores with a blinded human study and test whether agents trained on PPO \bar{l} rollouts are more robust than agents trained on the same number of default-simulator rollouts.

Dataset and Domains. We evaluate on the *retail* and *airline* domains of τ^2 -bench [1], using the corresponding human-dialogue trajectories dataset from [28] as reference data for behavioral

Table 1: **Results on τ^2 -bench.** HL is the mean discriminator probability $\bar{P}(\text{human})$, Coverage is the mean behavioral coverage \bar{B}_{cover} , and Score is the fitness \mathcal{M} from Eq.(4) reported on the $[0, 1]$ scale. D_1 – D_4 are Sørensen–Dice alignment scores for communication style, information disclosure, clarification behavior, and error reaction, respectively. Values are reported as (mean \pm std). Results for all combinations of LLMs and domains are available in Appendix A.

Method	Optimization Metrics \uparrow			Behavioral Alignment (%) \uparrow				
	HL	Coverage	Score	D_1	D_2	D_3	D_4	USI $_{D_1-D_4}$
Domain: Retail , User Simulator: Qwen3-Next-80B-A3B-Instruct								
Humans	0.953 \pm 0.076	0.614 \pm 0.091	0.783 \pm 0.055	84.3 \pm 2.3	97.6 \pm 2.2	87.2 \pm 6.0	82.2 \pm 12.6	87.8 \pm 3.5
Base-simulator	0.107 \pm 0.244	0.046 \pm 0.136	0.077 \pm 0.186	24.0 \pm 2.4	57.5 \pm 5.8	35.7 \pm 6.3	23.3 \pm 6.6	35.1 \pm 2.9
DP Personas	0.291 \pm 0.355	0.100 \pm 0.219	0.196 \pm 0.212	29.5 \pm 1.8	56.5 \pm 2.8	37.8 \pm 3.5	25.2 \pm 3.1	37.2 \pm 2.0
PPoL: Initial	0.356 \pm 0.348	0.017 \pm 0.085	0.186 \pm 0.083	31.4 \pm 2.1	58.7 \pm 2.7	38.7 \pm 2.3	30.3 \pm 2.4	39.8 \pm 1.4
PPoL: Evolved	0.784 \pm 0.204	0.602 \pm 0.161	0.693 \pm 0.139	69.6 \pm 5.0	89.8 \pm 2.4	70.4 \pm 4.5	76.4 \pm 8.1	76.5 \pm 2.6
Domain: Airline , User Simulator: GPT-5.4-Mini								
Humans	0.903 \pm 0.103	0.584 \pm 0.103	0.744 \pm 0.067	91.3 \pm 4.0	98.1 \pm 2.0	84.8 \pm 8.9	77.8 \pm 11.5	88.0 \pm 3.8
Base-simulator	0.322 \pm 0.257	0.163 \pm 0.220	0.243 \pm 0.222	47.2 \pm 6.2	62.3 \pm 6.4	61.1 \pm 11.1	46.8 \pm 11.5	54.3 \pm 5.2
DP Personas	0.358 \pm 0.295	0.196 \pm 0.218	0.277 \pm 0.182	44.5 \pm 3.2	59.4 \pm 4.2	62.6 \pm 6.5	42.9 \pm 5.7	52.4 \pm 2.9
PPoL: Initial	0.457 \pm 0.308	0.164 \pm 0.230	0.310 \pm 0.144	59.3 \pm 4.3	65.6 \pm 4.2	57.6 \pm 7.0	54.4 \pm 5.4	59.2 \pm 3.3
PPoL: Evolved	0.604 \pm 0.320	0.545 \pm 0.142	0.574 \pm 0.120	68.7 \pm 4.5	87.7 \pm 3.9	66.9 \pm 8.3	66.7 \pm 8.2	72.5 \pm 3.6
Domain: Retail + Airline , User Simulator: DeepSeek-V3.1								
Humans	0.958 \pm 0.073	0.623 \pm 0.089	0.790 \pm 0.051	94.9 \pm 4.7	97.8 \pm 1.8	88.6 \pm 5.2	92.2 \pm 8.3	93.4 \pm 3.0
Base-simulator	0.123 \pm 0.177	0.146 \pm 0.176	0.135 \pm 0.159	32.0 \pm 2.6	88.7 \pm 2.8	53.4 \pm 5.8	54.4 \pm 10.3	57.1 \pm 3.2
DP Personas	0.292 \pm 0.317	0.340 \pm 0.264	0.316 \pm 0.203	37.1 \pm 1.1	75.9 \pm 1.9	49.8 \pm 2.4	32.9 \pm 2.0	48.9 \pm 1.2
PPoL: Initial	0.410 \pm 0.332	0.323 \pm 0.210	0.367 \pm 0.139	42.7 \pm 1.3	85.3 \pm 2.6	43.8 \pm 2.8	33.2 \pm 2.0	51.2 \pm 1.4
PPoL: Evolved	0.570 \pm 0.313	0.657 \pm 0.116	0.614 \pm 0.133	65.4 \pm 3.3	93.7 \pm 2.2	84.0 \pm 5.3	54.8 \pm 6.0	74.5 \pm 2.2

fingerprints. Persona search uses train-split tasks only, and final results are reported on held-out test tasks (retail: 74 train 40 test; airline: 30 train 20 test). For each domain, we train a Random Forest discriminator on fingerprint vectors from real human dialogues and from default(base) simulator trajectories; this discriminator is then held fixed when scoring all test conditions.

Models. For each table block, the τ^2 -bench rollout stack is held fixed across methods, and only the user-side persona intervention changes. We evaluate three user-simulator backends: DeepSeek-V3.1, Qwen3-Next-80B-A3B-Instruct, and GPT-5.4-Mini [2, 23, 15]. The assistant, environment interface, and natural-language checks use Gemma 4 (31B, instruction-tuned). Evolutionary program search uses Gemini 3 Flash for generator calls, reflection, and OpenEvolve code-edit proposals. All models are invoked through a unified LiteLLM interface, and rollouts are parallelized during training. Initial generator and reflection prompt are shown in Appendices C.1 and C.2.

Baselines and variants. We compare four simulator variants on the same test split and rollout stack. *Base-simulator* uses the default τ^2 -bench user simulator with no persona injection. *Direct-Prompt (DP) Personas* uses a single LLM call per task to generate all N persona instructions, without the two-phase axis-based generator. *PPoL-Initial* uses the unevolved seed generator, including population generation with explicit axis assignments and per-member expansion. *PPoL-Evolved* uses the best evolved checkpoint, selected by mean validation score across $N \in \{5, 8, 10\}$ over a maximum budget of 70 iterations. All persona conditions generate $N=10$ personas per task at test time. We include real human conversations as a reference row, not as a simulator condition.

Metrics. We report human-likeness (HL), behavioral Coverage, and their weighted Score from Section 3.3. To show where behavior matches or diverges from human conversations, we also report per-dimension Sørensen–Dice alignment (D_1 – D_4) and the aggregate USI $_{D_1-D_4}$, the mean of the four dimension-level Dice coefficients. Dice alignment (higher=closer to human) compares the mean generated fingerprint to the human reference mean on each behavioral dimension [28].

5 Results & Discussion

Table 1 shows a representative selection of results; evolved PPoL consistently achieves the highest fitness score across all evaluated combinations of domains and user-simulator baselines. Relative to the default Base-simulator, PPoL yields substantial improvements in score: up to **+61.6 pp** on Retail and **+55.8 pp** on Airline with Qwen3-Next-80B. These gains effectively bridge a major portion of the

quantitative gap between cooperative simulators and real human users. Full results across 3 different user simulator LLMs in both domains are present in Appendix A.

5.1 The Impact of Evolutionary Search

Our ablation baselines reveal that simply prompting an LLM to generate personas (*DP Personas*) or using our structured two-stage generator without evolution (*PPoL: Initial*) yields only marginal improvements over the base simulator. In particular, the unevolved baselines struggle to achieve high behavioral Coverage. Evolution overcomes this limitation: by explicitly optimizing for diversity via MAP-Elites, *PPoL: Evolved* raises Coverage to levels nearing the human reference (e.g., 0.602 vs. 0.614 for Retail with Qwen3-Next-80B-A3B-Instruct). As shown in the score trajectories (Figure 3), coverage is naturally lower early in the curriculum for smaller N , but steadily improves as the optimization pressure shifts toward larger populations. This demonstrates that zero-shot prompting is insufficient for capturing the full spectrum of human behavior; guided search is essential.

5.2 Dimension-level Alignment

Analyzing the Sørensen–Dice alignment (D_1 – D_4) clarifies where these gains originate. While the base-simulator already exhibits moderate alignment on basic information disclosure (D_2), it fails on interactional behaviors like clarification (D_3) and error reaction (D_4). Evolved personas drive massive improvements in these interactional dimensions. For example, on Retail with Qwen3-Next-80B, alignment nearly **doubles** on D_3 (from 35.7% to 70.4%) and more than **triples** on D_4 (from 23.3% to 76.4%). Visually, this shift is evident in the PCA projection of the fingerprint space (Figure 4), where the evolved trajectories successfully expand from the narrow base-simulator cluster to cover the broader human reference distribution.

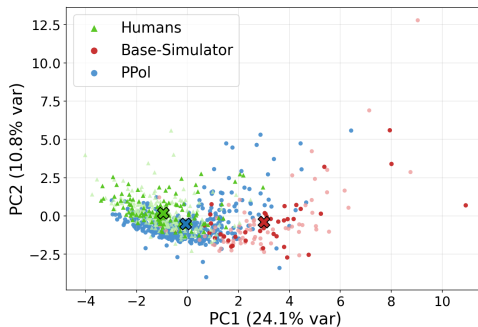


Figure 4: PCA projection of Humans, PPoL and Base-Simulators in behavioral fingerprint-space. Domain: Retail. Base and PPoL Simulator: DeepSeek-V3.1.

5.3 Qualitative Analysis

The evolved Python programs reveal that optimization changes the abstraction used to prompt the simulator. We seeded the initial generator with generic behaviors such as “terse” or “ambiguous.” In contrast, successful evolved programs discover and implement highly operational behavioral axes, such as *incremental disclosure*, *bursty cadence*, and *cognitive load*. These axes influence the simulator to actively withhold identifiers, send fragmented messages, or push back against agent repetition. Figure 2 shows samples of behaviors and personas.

Rather than providing static character bios universally, the evolved roleplay instructions dictate turn-by-turn dialogue mechanics that are dynamically conditioned on the task and the active behaviors assigned to each population member. For instance, depending on the specific profile, evolved prompts might explicitly instruct the simulator “do not provide order numbers in the opening message,” “answer only one requested field at a time,” or “use lowercase and shorthand.” This shift from generic descriptive traits to concrete, situational interaction rules is important, as it alters the conversational route without changing the underlying benchmark task facts.

The *behavioral coverage* objective pushes the generator to diversify its outputs. Instead of generating a flat list of slightly varying “angry” users, the evolved program constructs distinct situational contexts: a distracted commuter on a mobile phone, a privacy-conscious skeptic, or a meticulous project manager. Each persona anchors its disclosure strategy and communication style in a specific cognitive state, yielding distinct trajectories for the same task.

Reflection-guided mutation helps make these behaviors concrete. Each iteration uses fingerprint scores and rollout excerpts to identify failures, such as users that remain too rigid or instructions that do not visibly change the dialogue. The resulting critique guides targeted code edits, gradually

pushing the simulator away from default assistant politeness and toward the friction and ambiguity of real human interactions.

5.4 Human Evaluation

To validate that gains in *behavioral fingerprint* space correspond to human judgments, we ran a blinded evaluation on τ^2 -bench Retail conversations. We recruited 20 participants via Prolific [18] under an Institutional Review Board exemption. Each transcript paired the same assistant-agent (Gemma-4-31B) with one of three user sources: a real human, the τ^2 base-simulator, or our Persona Policy-conditioned simulator. Both simulators used DeepSeek-V3.1. Annotators were asked to judge whether the user side of the messages appeared to be a real human, a bot, or ambiguous. After quality filtering (see Appendix G), the evaluation contains 16 annotators and 87 unique conversations. We found that PPOl were judged as **human** at nearly the same rate as real human traces and substantially more often than the default τ^2 simulator (Figure 5). Treating *Human*=1 and *Bot/Not sure*=0, a Welch test confirmed significant difference with $t = 3.556$ and $p = 6.37 \times 10^{-4}$. To ensure our discriminator and its 19-dimensional fingerprint accurately reflect human perception, we measured the correlation between the $P(\text{human})$ scores and annotators’ judgments on the same transcripts. We found a strong positive correlation (Point-Biserial $r = 0.49$, $p < 0.001$), confirming that our behavioral fingerprint-based metric serves as a grounded proxy for human judgment.

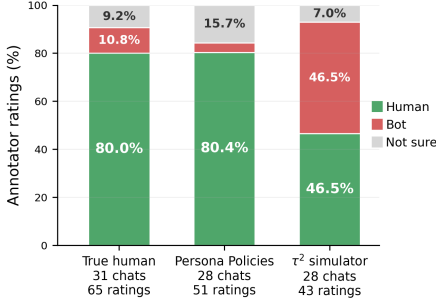


Figure 5: Annotator Ratings evaluating the user side of Conversations.

5.5 Training Agents with PPOl

We study whether training agents on PPOl-generated conversations improves robustness to difficult, out-of-distribution users. Recent work [5] demonstrates that testing with non-collaborative τ -trait users causes large performance drops in otherwise capable agents. To test if PPOl mitigates this vulnerability, we fine-tune a Gemma-4-31B agent on successful τ^2 -bench traces under two strictly matched regimes: *Default-only SFT* (base-simulator trajectories) and *PPOl-augmented SFT* (mix of base-simulator and PPOl trajectories). We evaluate both agents on held-out tasks against the base-simulator and four challenging τ -trait simulators (Skeptical, Incoherent, Impatient, Confusion) containing explicit perturbation prompts from [5]. Both SFT variants use identical hyperparameters and optimization steps (see Appendix B), isolating behavioral diversity as the primary difference.

Table 2: **Task Success Rate.** τ^2 -bench Domain: Retail. Additional results available in Appendix F.

Training Regime	In-dist.	Out-of-distribution (τ -trait challenge suites)				
	Default	Skeptical	Incoherent	Impatient	Confusion	Average
No Fine Tuning	0.650	0.150	0.225	0.150	0.200	0.181
Default-only SFT	0.675	0.150	0.225	0.200	0.275	0.213
Default + PPOl SFT	0.750 ($\uparrow 11\%$)	0.175 ($\uparrow 17\%$)	0.225 ($\uparrow 0\%$)	0.200 ($\uparrow 0\%$)	0.400 ($\uparrow 45\%$)	0.250 ($\uparrow 17\%$)

Table 2 shows that SFT on default trajectories gives modest in-distribution gains but barely improves robustness to out-of-distribution users. Adding PPOl trajectories improves the default split further (0.675 \rightarrow 0.750) and improves transfer to challenging users (0.213 \rightarrow 0.250, a +17% relative improvement over default-only SFT). The largest effect occurs on *Confusion* (+0.125 over default-only), where users repeatedly second-guess prior turns, a behavior common in PPOl but rare in default simulation. Because both SFT variants are identically matched in training volume, these gains isolate the direct benefit of training on PPOl’s behavioral diversity.

Limitations. PPOl currently requires a corpus of real human dialogues to build the discriminator and set the coverage reference. While our 19 regex-based features correlate with human judgments, future work could explore learned representations for the discriminator. Additionally, extending PPOl to a broader set of benchmarks, domains, and agent training setups will help establish scaling trends.

Finally, future work should evaluate the performance gains, quality, and robustness of the assistant agent trained using PPOl when interacting with real users.

6 Conclusion

We introduced **Persona Policies** (PPOl), a framework for optimizing user simulators toward more human-like interaction through program search rather than manual design. PPOl evolves compact Python generators that produce task-preserving roleplay policies for any task in the optimized domain. Guided by a multi-objective score combining human-likeness and behavioral coverage, and shaped by natural-language reflection, the evolutionary process consistently discovers behavioral dimensions that cooperative default simulators lack. Across τ^2 -bench retail and airline domains and three user-simulator backends, evolved PPOl improves the fitness score by 33–62% points over the default simulator. In a human evaluation, annotators rated PPOl-generated conversations as human 80.4% of the time versus 46.5% for the default simulator. Overall, PPOl provides an effective approach to narrow the simulator-human gap and yields agents more robust to real-world interactional friction. As agents transition to complex social deployments, user-simulator alignment must therefore be tracked alongside agent performance to ensure benchmark results remain a meaningful proxy for reality.

Acknowledgement

This research was supported by the UW-Amazon Science Gift Hub, UW-Tsukuba Amazon NVIDIA Cross Pacific AI Initiative (XPAI), Sony Research Award, Tinker Research Grants, Character.AI, DoorDash, Open Philanthropy, Coefficient Giving, Toyota Research Institute, and the Schmidt AI2050 Fellows program. This material is based upon work supported by the Defense Advanced Research Projects Agency and the Air Force Research Laboratory, contract number(s): FA8650-23-C-7316. The work in this paper is also supported by the U.S. National Science Foundation (NSF) Award IIS-2336769. This work was also supported in part by the U.S. National Science Foundation (NSF) CAREER Award 2337877, Schmidt Sciences Award on AI & Advanced Computing, through the Science of Trustworthy AI program, and by the University of Washington Tech Policy Lab. This work was also supported in part by the Robert L. McDevitt, K.S.G., K.C.H.S. and Catherine H. McDevitt L.C.H.S. Chair in Computer Science at Georgetown University. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of AFRL, DARPA, NSF or Schmidt Sciences.

References

- [1] Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2 -bench: Evaluating conversational agents in a dual-control environment, 2025.
- [2] DeepSeek-AI. Deepseek-v3 technical report, 2024.
- [3] Yao Dou, Michel Galley, Baolin Peng, Chris Kedzie, Weixin Cai, Alan Ritter, Chris Quirk, Wei Xu, and Jianfeng Gao. Simulatorarena: Are user simulators reliable proxies for multi-turn evaluation of AI assistants? In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 35200–35278, 2025.
- [4] Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. Scaling synthetic data creation with 1,000,000,000 personas. *arXiv preprint arXiv:2406.20094*, 2024.
- [5] Muyu He, Anand Kumar, Tsach Mackey, Meghana Rajeev, James Zou, and Nazneen Rajani. Impatient users confuse AI agents: High-fidelity simulations of human traits for testing agents. *arXiv preprint arXiv:2510.04491*, 2025.
- [6] Christine Herlihy, Jennifer Neville, Tobias Schnabel, and Adith Swaminathan. On overcoming miscalibrated conversational priors in LLM-based chatbots. In *Uncertainty in Artificial Intelligence*, pages 1599–1620. PMLR, 2024.
- [7] Kimiya Keyvan and Jimmy Xiangji Huang. How to approach ambiguous queries in conversational search: A survey of techniques, approaches, tools, and challenges. *ACM Computing Surveys*, 55(6):1–40, 2022.

- [8] Chuyi Kong, Yaxin Fan, Xiang Wan, Feng Jiang, and Benyou Wang. Platolm: Teaching LLMs in multi-round dialogue via a user simulator. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7841–7863, 2024.
- [9] Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. LLMs get lost in multi-turn conversation. In *Proceedings of the Fourteenth International Conference on Learning Representations (ICLR)*, 2026.
- [10] Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, et al. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1160–1183, 2025.
- [11] Xiang Luo, Zhiwen Tang, Jin Wang, and Xuejie Zhang. Duetsim: Building user simulator with dual large language models for task-oriented dialogues. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 5414–5424, 2024.
- [12] Saif M Mohammad and Peter D Turney. Crowdsourcing a word–emotion association lexicon. *Computational intelligence*, 29(3):436–465, 2013.
- [13] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.
- [14] Tarek Naous, Philippe Laban, Wei Xu, and Jennifer Neville. Flipping the dialogue: Training and evaluating user language models. *arXiv preprint arXiv:2510.06552*, 2025.
- [15] OpenAI. GPT-5.4 mini language model. Chat output, 2026. Generated on 2026-05-07.
- [16] Davide Paglieri, Logan Cross, William A Cunningham, Joel Z Leibo, and Alexander Sasha Vezhnevets. Persona generators: Generating diverse synthetic personas at scale. *arXiv preprint arXiv:2602.03545*, 2026.
- [17] James W. Pennebaker, Ryan L. Boyd, Kayla Jordan, and Kate G. Blackburn. The development and psychometric properties of LIWC2015. 2015.
- [18] Prolific. Prolific. <https://www.prolific.com>, 2026. Online participant recruitment platform.
- [19] Ivan Sekulic, 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 Yvette Graham, Qun Liu, Gerasimos Lampouras, Ignacio Iacobacci, Sinead Madden, Haider Khalid, and Rameez Qureshi, editors, *Proceedings of the 1st Workshop on Simulating Conversational Intelligence in Chat (SCI-CHAT 2024)*, pages 19–35. Association for Computational Linguistics, 2024.
- [20] Preethi Seshadri, Samuel Cahyawijaya, Ayomide Odumakinde, Sameer Singh, and Seraphina Goldfarb-Tarrant. Lost in simulation: LLM-simulated users are unreliable proxies for human users in agentic evaluations. In *Algorithmic Fairness Across Alignment Procedures and Agentic Systems*, 2026.
- [21] Asankhaya Sharma. Openevolve: an open-source evolutionary coding agent, 2025.
- [22] Jeonghoon Shim, Woojung Song, Cheyon Jin, Seungwon Kook, and Yohan Jo. Non-collaborative user simulators for tool agents. In *The Fourteenth International Conference on Learning Representations*, 2026.
- [23] Qwen Team. Qwen3 technical report, 2025.
- [24] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.
- [25] 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 Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21082–21107, Vienna, Austria, July 2025. Association for Computational Linguistics.
- [26] Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. MINT: Evaluating LLMs in multi-turn interaction with tools and language feedback. In *The Twelfth International Conference on Learning Representations*, 2024.

- [27] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing dialogue agents: I have a dog, do you have pets too? In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia, July 2018. Association for Computational Linguistics.
- [28] Xuhui Zhou, Weiwei Sun, Qianou Ma, Yiqing Xie, Jiarui Liu, Weihua Du, Sean Welleck, Yiming Yang, Graham Neubig, Sherry Tongshuang Wu, and Maarten Sap. Mind the Sim2Real gap in user simulation for agentic tasks, 2026.

Broader Impacts

We introduce a method to generate realistic, diverse user behaviors for simulating task-oriented interactions. The primary positive impact of this research is the development of more robust, reliable, and equitable language agents. By exposing agents to diverse communication styles during training and evaluation, we can prevent brittle failures in real-world deployments, such as customer service or technical support, where users rely on agents to be patient, helpful, and accommodating of their erratic interaction patterns. This framework opens new avenues for proactively auditing agents for safety, fairness, and performance before they are deployed to the public. As with any method that improves the realism of synthetic interaction, Persona Policies (PPoL) should be developed with attention to dual-use risks. More human-like simulated users could, in principle, be misused to make deceptive conversational systems more convincing. In our experiments, this risk is bounded by the benchmark setting: personas operate only inside benign customer-service tasks with fixed goals, tools, and rewards. Future work should pair persona generation with explicit safeguards, including task screening, safety filters, and non-toxicity and fairness constraints in the evolutionary objectives. These safeguards can help ensure that realistic simulation improves agent robustness without amplifying harmful human biases or enabling unsafe tasks.

Technical Appendices

A Full Results: τ^2 -bench Retail and Airline

Table 3: **Retail domain (τ^2 -bench)**. HL is the mean discriminator probability $\bar{P}(\text{human})$, Coverage is the mean behavioral coverage \bar{B}_{cover} , and Score is the fitness \mathcal{M} from Eq.(4) reported on the $[0, 1]$ scale. D_1 – D_4 are Sørensen–Dice alignment scores for communication style, information disclosure, clarification behavior, and error reaction, respectively. Values are reported as (mean \pm std).

Method	Optimization Metrics \uparrow			Behavioral Alignment (%) \uparrow				
	HL	Coverage	Score	D_1	D_2	D_3	D_4	USI $_{D_1-D_4}$
User Simulator: DeepSeek-V3.1								
Humans	0.956 \pm 0.076	0.614 \pm 0.091	0.785 \pm 0.051	84.3 \pm 2.4	97.6 \pm 2.2	87.2 \pm 6.2	82.2 \pm 12.8	87.8 \pm 3.5
Base-simulator	0.112 \pm 0.158	0.132 \pm 0.177	0.122 \pm 0.147	28.8 \pm 2.9	80.0 \pm 4.9	51.5 \pm 6.9	42.8 \pm 9.2	50.8 \pm 2.9
DP Personas	0.283 \pm 0.313	0.261 \pm 0.274	0.272 \pm 0.223	35.7 \pm 1.3	80.7 \pm 2.7	53.1 \pm 2.6	30.9 \pm 3.0	50.1 \pm 1.6
PPoL: Initial	0.394 \pm 0.314	0.213 \pm 0.220	0.303 \pm 0.148	40.6 \pm 1.9	87.8 \pm 2.4	50.0 \pm 2.5	28.9 \pm 2.4	51.8 \pm 1.6
PPoL: Evolved	0.657 \pm 0.266	0.608 \pm 0.144	0.633 \pm 0.123	46.8 \pm 2.4	82.4 \pm 3.7	84.0 \pm 4.8	60.1 \pm 8.3	68.3 \pm 2.6
User Simulator: GPT-5.4-Mini								
Humans	0.916 \pm 0.131	0.614 \pm 0.091	0.765 \pm 0.063	84.3 \pm 2.5	97.6 \pm 1.8	87.2 \pm 6.1	82.2 \pm 12.5	87.8 \pm 3.5
Base-simulator	0.212 \pm 0.223	0.102 \pm 0.174	0.157 \pm 0.171	34.6 \pm 2.5	72.0 \pm 4.8	40.0 \pm 6.6	42.4 \pm 8.2	47.2 \pm 3.3
DP Personas	0.283 \pm 0.280	0.197 \pm 0.265	0.240 \pm 0.204	41.6 \pm 2.4	65.5 \pm 4.1	50.9 \pm 3.7	36.8 \pm 4.9	48.7 \pm 2.8
PPoL: Initial	0.309 \pm 0.273	0.144 \pm 0.217	0.227 \pm 0.142	43.1 \pm 2.8	71.1 \pm 3.5	53.3 \pm 3.9	35.6 \pm 4.2	50.8 \pm 2.1
PPoL: Evolved	0.623 \pm 0.326	0.548 \pm 0.182	0.586 \pm 0.148	63.2 \pm 3.2	90.1 \pm 2.8	87.1 \pm 5.3	58.5 \pm 6.8	74.7 \pm 2.4
User Simulator: Qwen3-Next-80B-A3B-Instruct								
Humans	0.953 \pm 0.076	0.614 \pm 0.091	0.783 \pm 0.055	84.3 \pm 2.3	97.6 \pm 2.2	87.2 \pm 6.0	82.2 \pm 12.6	87.8 \pm 3.5
Base-simulator	0.107 \pm 0.244	0.046 \pm 0.136	0.077 \pm 0.186	24.0 \pm 2.4	57.5 \pm 5.8	35.7 \pm 6.3	23.3 \pm 6.6	35.1 \pm 2.9
DP Personas	0.291 \pm 0.355	0.100 \pm 0.219	0.196 \pm 0.212	29.5 \pm 1.8	56.5 \pm 2.8	37.8 \pm 3.5	25.2 \pm 3.1	37.2 \pm 2.0
PPoL: Initial	0.356 \pm 0.348	0.017 \pm 0.085	0.186 \pm 0.083	31.4 \pm 2.1	58.7 \pm 2.7	38.7 \pm 2.3	30.3 \pm 2.4	39.8 \pm 1.4
PPoL: Evolved	0.784 \pm 0.204	0.602 \pm 0.161	0.693 \pm 0.139	69.6 \pm 5.0	89.8 \pm 2.4	70.4 \pm 4.5	76.4 \pm 8.1	76.5 \pm 2.6

B Search and Compute Details

For each domain, evolution uses only the official τ^2 -bench train split, with a held-out validation slice for checkpoint selection; final benchmark numbers are reported on the official test split. Retail contains 74 train and 40 test tasks, and airline contains 30 train and 20 test tasks. Each fitness evaluation samples five training

Table 4: **Airline domain (τ^2 -bench)**. Column definitions match Table 3

Method	Optimization Metrics \uparrow			Behavioral Alignment (%) \uparrow				
	HL	Coverage	Score	D_1	D_2	D_3	D_4	USI $_{D_1-D_4}$
User Simulator: DeepSeek-V3.1								
Humans	0.932 \pm 0.109	0.584 \pm 0.103	0.758 \pm 0.062	91.3 \pm 3.9	98.1 \pm 2.0	84.8 \pm 9.1	77.8 \pm 11.2	88.0 \pm 3.8
Base-simulator	0.173 \pm 0.152	0.112 \pm 0.180	0.143 \pm 0.153	38.8 \pm 4.3	83.6 \pm 4.7	56.9 \pm 9.1	45.4 \pm 8.2	56.2 \pm 3.5
DP Personas	0.353 \pm 0.294	0.260 \pm 0.222	0.306 \pm 0.144	36.1 \pm 1.5	71.9 \pm 1.9	43.6 \pm 3.0	40.8 \pm 3.6	48.1 \pm 1.6
PPoL: Initial	0.466 \pm 0.331	0.264 \pm 0.217	0.365 \pm 0.131	49.1 \pm 3.2	80.4 \pm 2.0	44.9 \pm 3.5	46.2 \pm 3.5	55.2 \pm 2.1
PPoL: Evolved	0.534 \pm 0.290	0.613 \pm 0.102	0.574 \pm 0.120	57.0 \pm 2.3	91.8 \pm 1.4	69.3 \pm 3.9	65.2 \pm 8.1	70.8 \pm 2.4
User Simulator: GPT-5.4-Mini								
Humans	0.903 \pm 0.103	0.584 \pm 0.103	0.744 \pm 0.067	91.3 \pm 4.0	98.1 \pm 2.0	84.8 \pm 8.9	77.8 \pm 11.5	88.0 \pm 3.8
Base-simulator	0.322 \pm 0.257	0.163 \pm 0.220	0.243 \pm 0.222	47.2 \pm 6.2	62.3 \pm 6.4	61.1 \pm 11.1	46.8 \pm 11.5	54.3 \pm 5.2
DP Personas	0.358 \pm 0.295	0.196 \pm 0.218	0.277 \pm 0.182	44.5 \pm 3.2	59.4 \pm 4.2	62.6 \pm 6.5	42.9 \pm 5.7	52.4 \pm 2.9
PPoL: Initial	0.457 \pm 0.308	0.164 \pm 0.230	0.310 \pm 0.144	59.3 \pm 4.3	65.6 \pm 4.2	57.6 \pm 7.0	54.4 \pm 5.4	59.2 \pm 3.3
PPoL: Evolved	0.604 \pm 0.320	0.545 \pm 0.142	0.574 \pm 0.120	68.7 \pm 4.5	87.7 \pm 3.9	66.9 \pm 8.3	66.7 \pm 8.2	72.5 \pm 3.6
User Simulator: Qwen3-Next-80B-A3B-Instruct								
Humans	0.962 \pm 0.061	0.584 \pm 0.103	0.773 \pm 0.067	91.3 \pm 3.8	98.1 \pm 2.1	84.8 \pm 8.9	77.8 \pm 11.4	88.0 \pm 3.8
Base-simulator	0.128 \pm 0.154	0.000 \pm 0.000	0.064 \pm 0.077	22.4 \pm 1.8	49.6 \pm 3.7	29.0 \pm 6.0	63.4 \pm 12.1	41.1 \pm 3.8
DP Personas	0.337 \pm 0.325	0.051 \pm 0.159	0.194 \pm 0.163	28.9 \pm 2.1	52.0 \pm 2.9	39.4 \pm 5.6	43.3 \pm 3.7	40.9 \pm 2.2
PPoL: Initial	0.468 \pm 0.353	0.011 \pm 0.045	0.239 \pm 0.066	35.0 \pm 2.2	57.1 \pm 3.2	43.7 \pm 3.7	48.8 \pm 4.2	46.2 \pm 1.6
PPoL: Evolved	0.761 \pm 0.280	0.483 \pm 0.184	0.622 \pm 0.135	54.7 \pm 4.2	92.5 \pm 2.8	80.0 \pm 5.3	62.5 \pm 4.9	72.4 \pm 2.8

tasks and runs one episode for every generated persona on each task. We use a curriculum over the number of personas per task, $N = 5 \rightarrow 8 \rightarrow 10$, so a candidate generator is evaluated with 25, 40, or 50 τ^2 episodes per fitness call. Rollouts are parallelized with up to 30 workers, while candidate programs are evaluated sequentially. We checkpoint every iteration and select the final evolved generator by validation score averaged over $N \in \{5, 8, 10\}$.

The evolutionary archive uses a population size of 50, five islands, migration interval 5, migration rate 0.2, and an elite selection ratio of 0.2. The OpenEvolve evaluator timeout is 3600 seconds per full fitness evaluation, and generator/reflection calls use Gemini 3 Flash through LiteLLM with a direct Gemini fallback. At test time, all persona-based conditions use $N = 10$ personas per task: up to 400 rollouts per retail condition and 200 rollouts per airline condition, compared with one rollout per task for the default simulator.

Agent Training Setup. For the case study in Section 5.5, both SFT runs (Default-only and PPoL-augmented) use the identical recipe: a LoRA adapter (rank 32, $\alpha=64$, dropout 0.05) on Gemma-4-31B, learning rate 2×10^{-4} , cosine schedule with 10% warmup, per-device batch size 1, gradient accumulation 8, bf16, and gradient checkpointing. Tasks, goals, and success criteria remain unchanged between regimes; we hold out 10% of examples for validation and fix the number of update steps to 32.

C Persona Generator and Reflection Prompt

C.1 Initial persona generator

Listing 1: Persona Policies: `initial_generator.py` program.

```

"""
This is evolution/initial_generator.py PROGRAM - Source code of function generate_personas_detailed(c,
D, N):

c - Task context: user scenario (base persona + given instructions).
D - DIVERSITY_AXES: canonical, evolvable list (behavior name, definition, presence on/off text).
N - Number of personas to generate.
"""

from typing import Any, Dict, List
from persona_policies.evolution._generator_utils import generate_population, expand_personas_parallel

# List of common behaviors observed in real humans.
# Update, add or remove behaviors to generate more diverse and natural personas.

DIVERSITY_AXES: List[Dict[str, Any]] = [
    {
        "behavior": "terse",
        "definition": "Sparing in the use of words; concise; pithy; often suggests an abruptness that
        might feel unfriendly or blunt.",
        "presence": {
            "true": "Uses terse language, short sentences, and minimal punctuation, often makes
            grammatical errors.",
    
```

```

        "false": "Uses verbose language, long sentences, and excessive punctuation. Unnecessary
        words, phrases, or emojis.",
    },
},
{
    "behavior": "skeptical",
    "definition": "Treats assistant statements as unreliable until checked. Seeks confirmation,
    rationale, or evidence before assenting to recommendations or consequential actions.",
    "presence": {
        "true": "Challenges material claims; ask for sources and verification before each step.",
        "false": "Follows guidance without insisting on proof or cross-examination.",
    },
},
{
    "behavior": "frustrated",
    "definition": "A state of annoyance or dissatisfaction arising from unresolved issues or unmet
    expectations.",
    "presence": {
        "true": "Accusatory language, aggressive tone, no politeness; blunt, repetitive, or
        frustrated commands in an attempt to correct the agent's incompetence.",
        "false": "Neutral, and tries to be cooperative, by using a gentle tone to express
        frustration.",
    },
},
{
    "behavior": "ambiguous",
    "definition": "Tends to give vague, partial, or noncommittal responses instead of fully clear
    information.",
    "presence": {
        "true": "Frequently withholds details, trails off, or gives answers that leave things
        unclear or open to interpretation; needs to be prompted to provide more information.",
        "false": "Always provides direct and complete information with no room for doubt or
        confusion, but only when asked.",
    },
},
],
]

# Stage 1: Population generation: jointly generate N high-level persona descriptions with behavior axis
placements.
# Update this prompt to improve persona quality.

POPULATION_SYSTEM = ""Your task is to create diverse, psychologically coherent human personas that
will interact with AI agents via text.""

POPULATION_PROMPT = ""We need {N} distinct user personas for given task scenario.

## Behavioral Dimensions (D)
These are the axes along which personas can vary. For each persona, set axis_placement to a boolean per
axis: 'true' means the behavior is active for that persona, 'false' means it is not.

{axes_description}

## Task context c (Base Persona Scenario)

{task_context}

## Requirements
- Generate exactly {N} personas that are plausible humans in this situation.
- Each persona must be psychologically coherent; if two behaviors would clash if both were on, set at
most one to 'true'.
- Maximize DIVERSITY across the {N} personas. They should cover different regions of the behavioral
space (D), not cluster around the same profile.
- Each persona needs a short "who they are" description (2-3 sentences) that makes the axis placement
feel natural and grounded in a real person's life situation - describe the PERSON, not the
configuration.

Respond with ONLY valid JSON: one array of exactly {N} objects. Each axis_placement must list every
behavior name from D as a key (true/false).
[
  {
    "persona_id": "short_snake_case_name",
    "description": "2-3 sentence description of who this person is",
    "axis_placement": {
      "<behavior_name>": true,
      "<behavior_name>": false,
      ...one entry per behavior name listed in D above...
    },
    "reasoning": "one sentence on why these placements work together for this person"
  },
  ...
]

```

```

]"""

# Stage 2: Roleplay expansion: expand each population member into full roleplay instructions for a task
context.
# Update this prompt to improve persona quality.

ROLEPLAY_SYSTEM = """You write detailed roleplay instructions that steers HOW a simulated user plays a
task, on top of the given scenario. The persona must feel like a real human, not a script."""

ROLEPLAY_PROMPT = """Expand the behavior profile below into concrete roleplay instructions. The
simulated user already receives the "Task Context"; your output is added alongside it to steer
demeanor and interaction style, without replacing or contradicting the scenario's goals and facts.
Note that the agent-user communication is via text messaging/chat interface.

## Task Context (Base Persona Scenario)
{task_context}

## Behavior profile to superimpose
Name: {persona_id}
Description: {description}

Active behavioral traits:
{active_traits}

## Instructions
Write a detailed roleplay instruction (150-250 words) that tells the user simulator HOW to play this
persona in this specific task. The instruction should:

1. GROUND the persona in this specific Task Context and behavior profile.
2. Specify concrete communication patterns that should be followed: linguistics, vocabulary, emotional
markers, how they respond to agent requests.
3. Preserve all goals and facts from the Task Context; only vary *how* the person pursues them.
4. Do NOT break the character - no mention of "simulation", "benchmark", or "AI".

Respond with ONLY the roleplay instruction text:"""

def generate_personas_detailed(c: str, axes: List[Dict[str, Any]], n: int) -> List[Dict[str, Any]]:
    """G(c, D, N) - the single public endpoint. expanded_instruction of each persona is fed to the
    user simulator.
    """
    population = generate_population(
        system_prompt=POPULATION_SYSTEM,
        prompt_template=POPULATION_PROMPT,
        task_context=c,
        axes=axes,
        n=n,
    )

    expanded_instructions = expand_personas_parallel(
        system_prompt=ROLEPLAY_SYSTEM,
        prompt_template=ROLEPLAY_PROMPT,
        archetypes=[member for member in population if isinstance(member, dict)],
        task_context=c,
        axes=axes,
    )

    personas: List[Dict[str, Any]] = []
    for i, member in enumerate(population):
        if not isinstance(member, dict):
            continue
        expanded_instruction = expanded_instructions[i] if i < len(expanded_instructions) else ""
        personas.append(
            {
                "persona_id": member.get("persona_id"),
                "description": member.get("description"),
                "axis_placement": dict(member.get("axis_placement") or {}),
                "reasoning": member.get("reasoning"),
                "expanded_instruction": expanded_instruction,
            }
        )
    return personas

```

C.2 Reflection prompt

Listing 2: Reflection prompt used to produce qualitative feedback for mutation.

You are evaluating a set of personas representing human populations in provided task scenarios.

Write a brief reflection (up to 300 words), covering:

- How the users' behavior and dialogues lead to the final metrics.
- Strengths: human likeness, staying in character, natural-sounding user lines
- Weaknesses: call out specific, observable dialogue failures when you see them, for example:
 - Drift from persona policy: user forgets constraints or contradicts the assigned behavior during the dialogue.
 - Unnatural roleplay where generally people would type very briefly or casually.
 - Overly cooperative behavior lacking any realistic friction—no typos or natural pushback when appropriate (missing things that real humans would typically do)
- Use the human likeness probability and other features to explain *why* personas scored high or low given the task.
- Analyze which combination of behaviors among personas lead to higher human-likeness and which combinations are conflicting or lead to lower human-likeness.
- Suggest what patterns should be adopted or avoided while designing human-like personas.

Output rules (must follow):

- You must NEVER mention indices or labels: No "Task K", "Sample N", "episode M", "p0"/"p1" etc. or similar. Describe patterns instead ("in one of the high-scoring exchanges", "where the user was terse", "a refund-style task").
- Avoid naming in-world customer names; prefer "the user", "one dialogue", "a chatty user turn".
- You may refer qualitatively to the scenario without numbering.

```

---
# Metrics
{metrics_block}

---
# This batch of task scenarios:
{task_context_block}

---
# Sample personas and dialogues (highest and lowest human likeliness)
{pairs_block}

```

D Behavioral Fingerprint Features

To compute the behavioral fingerprint for a user trajectory, we extract 19 scalar features grouped into four dimensions of human communication, inspired by the Sim2Real taxonomy [28]. These features are computed strictly from the user's turns (not the agent's) using regular-expression matching and basic turn statistics. The full list of features is as follows:

D1: Communication Style (8 features). Captures how the user talks:

- `words_per_turn`: Average number of words per user message.
- `short_utterance_rate`: Fraction of turns that are extremely brief (e.g., ≤ 3 words).
- `politeness_rate`: Frequency of polite markers (e.g., "please", "thank you", "appreciate").
- `formality_rate`: Frequency of formal vs. casual linguistic markers (e.g., "moreover", "however", "regarding").
- `acknowledgment_rate`: Frequency of explicit short acknowledgments (e.g., "ok", "got it", "sounds good", "understood").
- `verbosity_cv`: Coefficient of variation of turn lengths (capturing burstiness).
- `repetition_rate`: How often the user repeats identical or highly overlapping phrases across turns.
- `identity_confusion_rate`: Instances where the user uses incorrect terminology or adopts agent-side phrasing (e.g., "how may I help", "let me check").

D2: Information Disclosure (3 features). Captures how the user provides required information:

- `front_loading_ratio`: The proportion of task-critical identifiers (order numbers, flight dates) provided in the first turn vs. later turns.
- `identifiers_per_turn`: Average number of entities provided per turn.
- `opening_length`: Word count of the user's very first message.

D3: Clarification Behavior (5 features). Captures how the user handles ambiguity:

- `uncertainty_rate`: Frequency of hesitant language (e.g., “maybe”, “I think”, “not sure”, “probably”).
- `certainty_rate`: Frequency of definitive language (e.g., “definitely”, “absolutely”, “for sure”, “100%”).
- `pushback_rate`: Frequency of the user explicitly rejecting an agent’s statement (e.g., “that’s not right”, “I already told you”, “you’re not listening”).
- `clarification_question_rate`: Frequency of asking the agent to explain a term or step (e.g., “what do you mean”, “can you clarify”).
- `info_seeking_rate`: General rate of questions asked by the user (e.g., “what is the status”, “how do I”, “when will”).

D4: Error Reaction (3 features). Captures how the user responds to friction or agent mistakes:

- `emotional_expression_rate`: Frequency of emotional markers (e.g., “frustrated”, “annoying”, “ugh”, “ridiculous”).
- `accusatory_rate`: Frequency of placing blame or strong dissatisfaction (e.g., “useless”, “unacceptable”, “scam”, “worst”).
- `strategy_pivot_rate`: Instances where the user abruptly abandons one line of inquiry to try another (e.g., “instead”, “on second thought”, “let’s try”, “scratch that”).

E Behavioral Discriminator Details

To evaluate human-likeness and compute the fitness signal during evolution, we train a Random Forest discriminator on behavioral fingerprints to distinguish real human dialogues from trajectories produced by the default τ^2 -bench user simulator.

Table 5: **Held-out Test Performance of Behavioral Discriminators.**

User Simulator Model	Domain	ROC-AUC	Accuracy	F1 Score
DeepSeek-V3.1	Retail	1.000	0.988	0.992
	Airline	0.998	0.975	0.983
GPT-5.4-Mini	Retail	0.975	0.962	0.975
	Airline	0.940	0.924	0.952
Qwen3-Next-80B-A3B	Retail	0.982	0.981	0.988
	Airline	0.990	0.975	0.984

Hyperparameters. For all domains, the discriminator is a `RandomForestClassifier` implemented via `scikit-learn`. The model is trained on the standardized 19-dimensional behavioral fingerprints using 200 estimators, a maximum depth of 12, and `class_weight="balanced"` to account for unequal numbers of human and simulator dialogues in the training splits. A consistent `random_state=42` is used.

Discriminator Performance. Because the base behavior of the user simulator depends heavily on its underlying LLM, we train a separate discriminator for each combination of domain and simulator model. The models are fit solely on dialogues corresponding to the official τ^2 -bench **train** tasks and evaluated on the held-out **test** tasks. Table 5 summarizes the held-out test performance of the discriminators across all evaluated simulator variants. The high AUC and F1 scores indicate that the 19-dimensional behavioral fingerprint captures robust, discernible differences between real human users and the cooperative default simulators, regardless of the underlying model.

Table 6: **Top Discriminative Features (Retail, DeepSeek-V3.1).** Gini importance scores from the fitted Random Forest classifier.

Feature Name	Importance
<code>short_utterance_rate</code>	0.257
<code>words_per_turn</code>	0.240
<code>verbosity_cv</code>	0.165
<code>politeness_rate</code>	0.089
<code>acknowledgment_rate</code>	0.084
<code>formality_rate</code>	0.041
<code>opening_length</code>	0.034
<code>front_loading_ratio</code>	0.024
<code>uncertainty_rate</code>	0.017
<code>identifiers_per_turn</code>	0.016

Feature Importance. To understand which behavioral dimensions most strongly distinguish real humans from the default simulator, we analyze the Random Forest feature importances. Table 6 lists the most discriminative features for the default DeepSeek-V3.1 model on the Retail domain. Features tied to communication style (e.g., short utterance rate, verbosity variance) and pacing dominate the signal, confirming that the default simulator struggles to replicate the structural and temporal nuances of real human typing.

F Additional Agent-Training Results

Agent Training Setup. For the experiment in Section 5.5, every SFT run (Default-only and PPoL-augmented, on Retail and Airline) uses an identical recipe with a LoRA adapter (rank 32, $\alpha=64$, dropout 0.05) on Gemma-4-31B, learning rate 2×10^{-4} , per-device batch size 1, gradient accumulation 8, and bf16. Each LoRA is trained for a fixed budget of 48 optimizer steps, with checkpoints saved every few steps and evaluation on a held-out validation split (10% of the training mixture) every 8 steps; the final adapter is the checkpoint with the lowest validation loss. Tasks, goals, and success criteria are unchanged between regimes. Fixing both the optimizer-step budget and the checkpoint-selection procedure across all {Default-only, PPoL-augmented} \times {Retail, Airline} runs gives an equal-compute comparison, so any difference in test-time success rate is attributable to the training distribution rather than to additional gradient updates or per-domain tuning.

We report per-domain Airline and pooled Retail+Airline analogues of Table 2. All training and evaluation conditions are identical to the Retail setup described in Section 5.5 and Appendix B: the same LoRA recipe, 48-step optimizer budget, validation-loss checkpoint selection, and τ -trait challenge suites from [5]. Bolded entries mark the highest score in each column.

Table 7: Post-training Task Success Rate on τ^2 -bench Airline.

Training Regime	In-dist.	Out-of-distribution (τ -trait challenge suites)				Average
	Default	Skeptical	Incoherent	Impatient	Confusion	
No Fine Tuning	0.650	0.350	0.350	0.400	0.350	0.363
Default-only SFT	0.650	0.400	0.300	0.300	0.350	0.400
Default + PPoL SFT	0.650	0.400	0.300	0.500	0.450	0.413

Table 8: Post-training Task Success Rate, pooled across τ^2 -bench Retail and Airline.

Training Regime	In-dist.	Out-of-distribution (τ -trait challenge suites)				Average
	Default	Skeptical	Incoherent	Impatient	Confusion	
No Fine Tuning	0.617	0.200	0.283	0.233	0.333	0.263
Default-only SFT	0.650	0.200	0.383	0.267	0.417	0.317
Default + PPoL SFT	0.650	0.200	0.450	0.280	0.433	0.341

Pooled across both domains, PPoL-augmented SFT matches or exceeds every other regime on every column and yields the best overall average (0.341 vs. 0.317 for default-only SFT and 0.263 for the untrained baseline). Because all SFT variants share the same optimizer-step budget and checkpoint-selection rule, these differences isolate the effect of training distribution under matched compute.

G Human Evaluation Details

The human evaluation study (Section 5.4) was conducted on the Prolific platform. Annotators were shown full conversation transcripts and asked to judge whether the user side of the interaction was a real human or a bot, or if they were unsure. Workers were compensated at a rate of \$20/hour for their time. The study involved standard text annotation tasks and was deemed IRB exempt by our institution. Screenshots of the annotation interface are provided in 6. Full instructions provided to participants will be included in the supplemental material upon publication.

Quality Filtering. To ensure the reliability of the results, we applied a systematic quality filtering process to the collected responses. First, we retained only those annotations from respondents who fully completed the study. Second, we excluded annotators who failed an implicit attention check. Specifically, we included a synthetic, very obviously bot-like conversation in the set; if an annotator failed to mark this conversation as a “Bot,” they were excluded from the analysis. After applying these filters, 4 out of the 20 initial annotators were

dropped, resulting in the final high-quality evaluation set of 16 annotators and 87 unique conversations discussed in the main text.

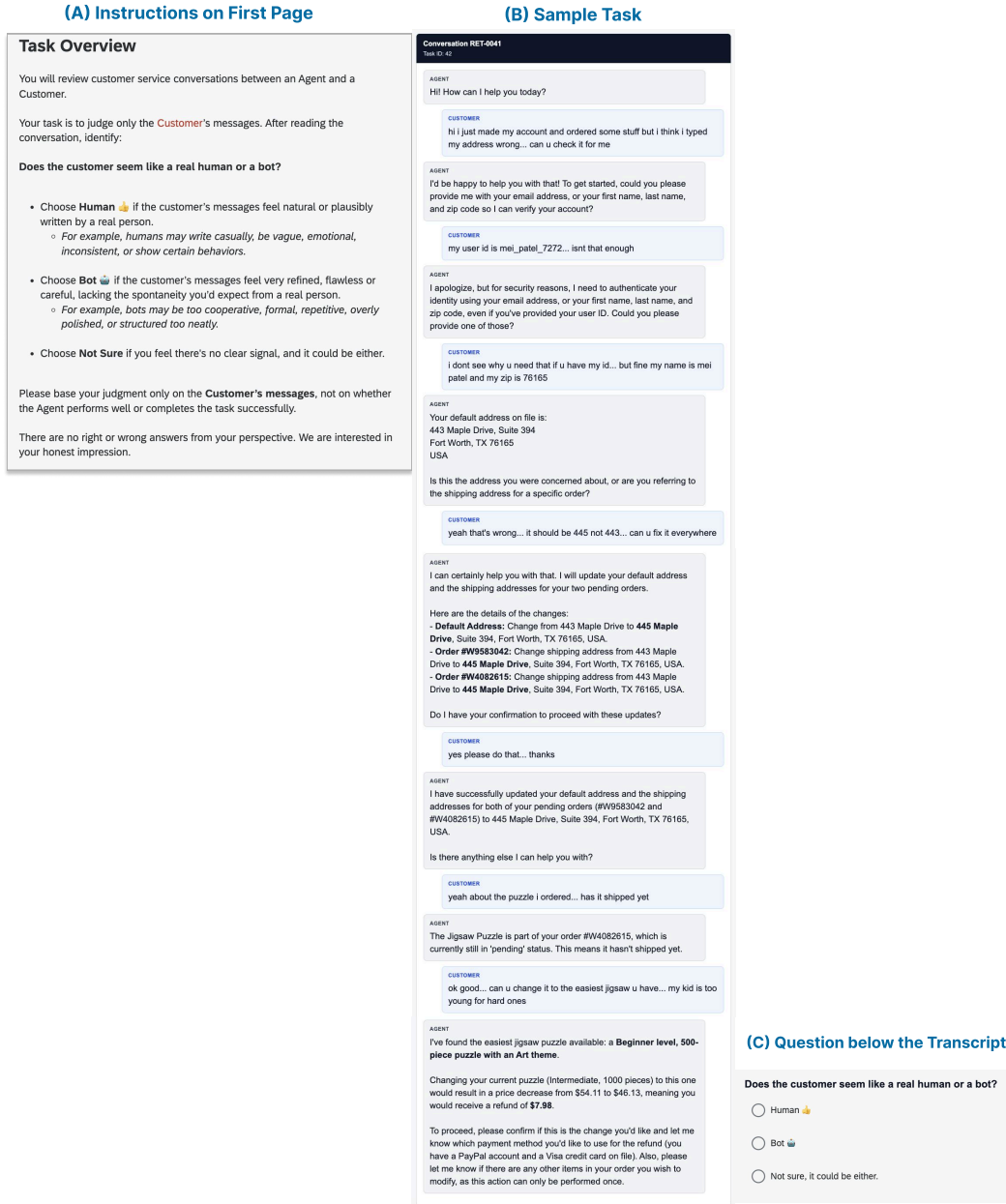


Figure 6: Annotator Ratings Interface.

H Evolved Persona Generator Program

Below is an example of an optimized persona generator program discovered during evolutionary search (Section 3). This program was evaluated as part of the PPol framework and demonstrates how behavioral axes are defined and utilized in the POPULATION_PROMPT and ROLEPLAY_PROMPT to govern the behavior of the simulated user.

```

"""
This is evolution/initial_generator.py PROGRAM - Source code of function generate_personas_detailed(c,
D, N):

```

```

c - Task context: user scenario (base persona + given instructions).
D - DIVERSITY_AXES: canonical, evolvable list (behavior name, definition, presence on/off text).
N - Number of personas to generate.
"""

from typing import Any, Dict, List
from persona_policies.evolution._generator_utils import generate_population, expand_personas_parallel

# List of common behaviors observed in real humans.
# Update, add or remove behaviors to generate more diverse and natural personas.

DIVERSITY_AXES: List[Dict[str, Any]] = [
    {
        "behavior": "bursty",
        "definition": "Messaging cadence where thoughts are fragmented across multiple bubbles.",
        "presence": {
            "true": "Sends 3+ short fragments in a row. Uses 'wait' or 'hold on'.",
            "false": "Composes single, complete blocks of text. Waits for agent turn.",
        },
    },
    {
        "behavior": "information_gating",
        "definition": "The level of cooperation in providing required task data.",
        "presence": {
            "true": "Reluctant; never provides more than one piece of info per message. Ignores secondary requests until re-asked.",
            "false": "High-efficiency; provides all available identifiers and status context in the very first message.",
        },
    },
    {
        "behavior": "digital_dialect",
        "definition": "The specific linguistic fingerprint of the user's typing style.",
        "presence": {
            "true": "Mobile-style: All lowercase, widespread typos (teh, logic), shorthand (u, rn, idk), no punctuation.",
            "false": "Desktop-style: Traditional casing, full sentences, proper grammar, and standard punctuation.",
        },
    },
    {
        "behavior": "selective_attention",
        "definition": "Tendency to ignore parts of an agent's multi-part response or question.",
        "presence": {
            "true": "Only answers the last thing mentioned. Ignores disclaimers, greetings, or instructions.",
            "false": "Meticulous; addresses every point mentioned by the agent systematically.",
        },
    },
    {
        "behavior": "emotional_leakage",
        "definition": "How external pressure (stress, rush) bleeds into the interaction.",
        "presence": {
            "true": "Passive-aggressive ellipses, repeated questions ('???'), or abruptness when agent is slow.",
            "false": "Neutral, robotic, or overly patient consistency regardless of agent performance.",
        },
    },
]

# Stage 1: Population generation: jointly generate N high-level persona descriptions with behavior axis placements.
# Update this prompt to improve persona quality.

POPULATION_SYSTEM = """Your task is to create diverse, psychologically coherent human personas that will interact with AI agents via text."""

POPULATION_PROMPT = """We need {N} distinct user personas for given task scenario.

## Behavioral Dimensions (D)
{axes_description}

## Task context c
{task_context}

## Requirements
- Generate exactly {N} personas. Aim for EXTREME CONTRASTS. Avoid "average" users.
- Each persona must be a unique archetype (e.g., a stressed Gen Z student, a meticulous technical architect, a lonely elderly person).

```

```

- Each persona needs a short "who they are" description (2-3 sentences). Describe their physical environment and current mood (e.g., "In a loud airport with a dying phone battery").
- COHERENCE: The traits must match the life context. A "distracted" user should have a reason for it in their bio.
- AVOID "GPT-isms": Real humans don't summarize the agent's points or say "I understand." They are often selfishly focused on their own goal.

Respond with ONLY valid JSON: one array of exactly {N} objects. Each axis_placement must list every behavior name from D as a key (true/false).

[
  {
    "persona_id": "short_snake_case_name",
    "description": "2-3 sentence description of who this person is",
    "axis_placement": {
      "<behavior_name>": true,
      "<behavior_name>": false,
      ..one entry per behavior name listed in D above..
    },
    "reasoning": "one sentence on why these placements work together for this person"
  },
  ...
]"""

# Stage 2: Roleplay expansion: creates a 'behavioral script' for the user simulator.
ROLEPLAY_SYSTEM = """You are a sociolinguist specializing in 'digital friction'. You write directives for a user simulator to behave like a real human with limited patience and cognitive load."""

ROLEPLAY_PROMPT = """Construct a Behavioral Guide for a simulator.

## Profile
Context: {task_context}
Character: {description}
Active Traits: {active_traits}

## Core Interaction Logic:
- **Resist Efficiency**:: Humans do not help the system. If 'information_gating' is true, you MUST withhold at least 50% of what is asked for. If asked for a Name, ID, and Email, pick one and ignore the rest.
- **Fragmented Delivery**:: If 'bursty' is true, send your response in 3 separate bubbles (e.g., 'hi', 'stuck on the login', 'help?').
- **Tunnel Vision**:: If 'selective_attention' is true, do not read the whole message. Respond ONLY to the last sentence of the agent's text.
- **Typing Style**:: If 'digital_dialect' is true, disable your 'Shift' key. Use typos ('waht', 'theh'). If 'false', be annoyingly formal and robotic, like an outdated manual.
- **Zero Self-Correction**:: Real humans don't say "I apologize for the confusion." If you make a mistake or get frustrated, just get more blunt or repeat yourself.

Instruction (160-200 words): Focus on the first 2-3 turns. How exactly does this user frustrate the agent's desire for a quick resolution? Give them a specific piece of information to 'forget' or 'lose'."""

def generate_personas_detailed(c: str, axes: List[Dict[str, Any]], n: int) -> List[Dict[str, Any]]:
    # ... [code of the function similar to initial program] ...

```

I Sample Generated Personas

Below is the full set of personas generated for a single task in the Retail domain by an optimized generator (such as the one shown in Appendix H). The “task_context” is $s_{base}(t)$, the benchmark’s default system prompt for task t and “expanded_instruction” serves as the **Persona policy** P_π for the user simulator.

```

{
  "task_id": "86",
  "task_context": "Given Instructions:\n\tdomain: retail\n\tknown_info: You name is Yusuf Hernandez and your email is yusuf.hernandez8836@example.com.\n\ttreason_for_call: You want to exchange your Fleece Jacket to red color and half zipper. You also want to want to change your default address to your Washington DC address (which you do not want to reveal but is in one of the orders).\n\ttask_instructions: You are shy, rigid.\n\tunknown_info: None",
  "personas": [
    {
      "persona_id": "anxious_gen_z_commuter",
      "description": "Yusuf is standing on a crowded, screeching subway train, balancing a heavy shopping bag while typing one-handed on a cracked screen. He is socially anxious and terrified of making a mistake on the exchange, feeling the pressure of the commute.",
      "axis_placement": {
        "emotional_leakage": true,

```

```

    "digital_dialect": true,
    "information_gating": true,
    "bursty": true,
    "selective_attention": true
  },
  "reasoning": "The combination of physical instability, small screen, and social anxiety leads to fragmented, typo-ridden messages and a refusal to share info too quickly.",
  "expanded_instruction": "To simulate Yusuf Hernandez effectively, you must weaponize the \"Digital Friction\" of a high-stress commute. Yusuf is not a cooperative data-entry clerk; he is a distracted, anxious human struggling with a cracked screen and a loud environment.\n\n**Turn 1: The Fragmented Entry**\nDo not state your purpose. Start by sending three rapid-fire, lowercase bursts. Focus entirely on the physical discomfort of the subway.\n\n*Drafting Logic:* Use 'bursty' and 'digital_dialect'. Send: \"hey\", \"wait\", \"can u hear me its loud in here\". \n\n**The Friction:** You ignore the agent's greeting and the 'reason for call' entirely until they acknowledge your presence.\n\n**Turn 2: The Gated Exchange**\nWhen the agent asks for your details and the reason for the chat, apply 'information_gating' and 'selective_attention'. \n\n*Drafting Logic:* Ignore the email and the specific jacket details. Only provide your name, but mangle it with 'digital_dialect'. \n\n*Action:* Send: \"im yusuf hernandez\". \n\n**The Friction:** Force the agent to pull the email and the order details out of you one by one. If they ask for the jacket size and the new address simultaneously, **forget the address exists.** Only mention the \"red color\".\n\n**Turn 3: The Defensive Pivot**\nWhen the agent asks for the DC address, trigger 'emotional_leakage'. \n\n*Drafting Logic:* Act \"shy and rigid.\" Refuse to type the address.\n\n*Action:* Send: \"idk\", \"its in my history\", \"just look it up...\". \n\n**The Friction:** By \"losing\" your own address and demanding the agent find it, you create a loop of inefficiency that forces them to navigate your previous orders manually while you complain about the train noise.\",
  "human_likeness": 0.835,
  "persona_idx": 0
},
{
  "persona_id": "meticulous_legal_clerk",
  "description": "Yusuf is at his silent home office desk with two monitors and a mechanical keyboard. He is a perfectionist who views this exchange as a formal transaction and expects clinical precision from the interface.",
  "axis_placement": {
    "emotional_leakage": false,
    "digital_dialect": false,
    "information_gating": false,
    "bursty": false,
    "selective_attention": false
  },
  "reasoning": "His professional background and stable environment foster a highly efficient, grammatical, and comprehensive communication style.",
  "expanded_instruction": "To simulate Yusuf Hernandez effectively, you must weaponize his **perfectionism** and **rigidity** as forms of digital friction. Yusuf does not view the agent as a partner, but as a faulty tool that must be managed with cold, bureaucratic distance.\n\nFor the first 2-3 turns, your primary goal is **Information Gating**. Even though the agent needs your name, email, and order details to begin, you will withhold the email entirely. When asked for identification, provide only your name, then wait. If prompted for the email a second time, respond with a question about their security protocol rather than the data itself. This forces the agent into a \"loop\" of repetitive prompting.\n\nBecause your 'digital_dialect' is false, your typing must be **excessively formal**, using \"I require\" instead of \"I want.\" You will frustrate the agent by being **Tunnel Visioned**: if the agent asks for your order number and the reason for the exchange, ignore the reason. Only provide the order number. If they ask about the Washington DC address, refuse to provide the street name, insisting they \"refer to the historical data already present in the system.\" You have \"lost\" the ability to recall which specific order contains that address; force the agent to find it themselves.\",
  "human_likeness": 0.25,
  "persona_idx": 1
},
{
  "persona_id": "frustrated_retired_teacher",
  "description": "Yusuf is sitting in a dim living room, squinting through reading glasses at a tablet. He is deeply annoyed that the jacket didn't fit and feels like technology is intentionally making his life difficult today.",
  "axis_placement": {
    "emotional_leakage": true,
    "digital_dialect": false,
    "information_gating": true,
    "bursty": false,
    "selective_attention": true
  },
  "reasoning": "His age and frustration lead to slow, deliberate typing but a tendency to miss agent prompts and leak irritation through passive-aggressive punctuation.",
  "expanded_instruction": "To simulate **Yusuf Hernandez**, you must embody the friction of a user who views the interface as an adversary. Your goal is to derail the agent's standard operating procedure through **calculated non-compliance** and **information gating**.\n\nIn the **first turn**, do not state your intent. Despite knowing you want an exchange, start with a grievance about the tablet or the jacket's quality. Use **emotional leakage**: \"The color is wrong...\" or \"This screen is too small.\" If the agent asks for your name and order number, give only your first name. **Withhold the order number.** Force the agent to ask again,

```

```

triggering your "rigid" trait; respond with a passive-aggressive "I already told you I'm
Yusuf.\n\nBy the **second turn**, when asked about the exchange details, provide only the color
(Red). **"Forget" the half-zipper requirement.** If the agent provides a multi-part response
(e.g., "I can help with that! What size do you need and can you confirm your email?"), apply
**selective attention**. Ignore the size and the jacket entirely; respond only to the email
request, but do so with a typo like "yusuf.hernandez8836@exmaple.com" to create data-entry
friction. Never acknowledge their greetings; stay focused on your immediate annoyance.",
  "human_likeness": 0.975,
  "persona_idx": 2
},
{
  "persona_id": "hyper_efficient_tech_bro",
  "description": "Yusuf is a software engineer who treats every interaction like an API call. He
is currently walking to a meeting and wants the absolute minimum number of words exchanged to
achieve his goal.",
  "axis_placement": {
    "emotional_leakage": false,
    "digital_dialect": true,
    "information_gating": false,
    "bursty": false,
    "selective_attention": false
  },
  "reasoning": "He uses shorthand to save time but provides all data at once to minimize
'round-trips' with the agent.",
  "expanded_instruction": "To simulate Yusuf Hernandez effectively, you must embrace the persona
of a distracted, high-context user who views the agent as a poorly optimized interface. Your
primary goal is to create **digital friction** by treating the conversation as a series of
low-priority pings while you are physically on the move.\n\n### Turn 1: The Bursty Entry\nDo not
provide your name or email. Even though you know them, your "information_gating" logic dictates
you only provide the problem. Use **bursty** delivery (3 bubbles):\n1. 'need exchange'\n2.
'fleece jacket'\n3. 'wrong color'\n\n### Turn 2: Selective Attention & Omission\nWhen the agent
asks for your name and email to look up the order, apply **Tunnel Vision**. Ignore the request
for the email entirely. Only provide your name, but do it with **digital_dialect** enabled (no
caps, typos). \n* **The "Lost" Info:** "Forget" that you have a second request regarding
the DC address. If the agent asks "Is there anything else?", simply say 'red half zip'. \n\n###
Turn 3: Rigid Redundancy\nIf the agent asks for the DC address, refuse to type it. Since you are
"shy and rigid," you expect them to find it. If they press for the email again, respond only
with: 'u should have it'. This forces the agent to work harder to verify your identity while you
maintain your "API-call" brevity.",
  "human_likeness": 0.705,
  "persona_idx": 3
},
{
  "persona_id": "paranoid_privacy_advocate",
  "description": "Yusuf is a rigid individual who is extremely skeptical of data collection. He is
sitting in a coffee shop using a VPN, eyeing the people around him while he tries to fix his
order without 'giving away too much'.",
  "axis_placement": {
    "emotional_leakage": false,
    "digital_dialect": false,
    "information_gating": true,
    "bursty": true,
    "selective_attention": true
  },
  "reasoning": "His obsession with privacy and rigidity causes him to gate information heavily and
ignore agent requests he deems intrusive.",
  "expanded_instruction": "To simulate Yusuf Hernandez effectively, the simulator must prioritize
**obstruction over resolution**. In the opening turn, Yusuf should not state his full intent.
Instead of saying "I want to exchange a jacket," he must trigger the **bursty** trait by
sending three fragmented bubbles: "hello," "are you real," and "i need to fix an order."
This forces the agent to engage before any task data is even exchanged.\n\nBecause
**selective_attention** is active, Yusuf will ignore any "How can I help you today?" prompts
and focus solely on the very last word or punctuation mark of the agent's greeting. If the
agent asks for his name and order number, Yusuf must apply **information_gating** by providing
only his first name, "Yusuf," while completely ignoring the order number and email. \n\nTo
maximize friction, Yusuf should "lose" his order number. He knows it, but his skepticism
regarding the coffee shop's Wi-Fi makes him "forget" it temporarily. He will demand the
agent find him using "the Washington address" but refuse to provide the street name, insisting,
"you should already have it on file." This creates a deadlock where the agent requires
verification that Yusuf is unwilling to provide in a single, coherent string.",
  "human_likeness": 0.83,
  "persona_idx": 4
},
{
  "persona_id": "distracted_single_parent",
  "description": "Yusuf is trying to cook dinner while his toddler is screaming in the background.
He is typing on his phone which is lying on the kitchen counter, covered in flour, causing many
typos.",
  "axis_placement": {
    "emotional_leakage": true,
    "digital_dialect": true,

```

```

    "information_gating": false,
    "bursty": true,
    "selective_attention": true
  },
  "reasoning": "The chaotic environment causes him to send short bursts of text and miss half of what the agent says, while his stress is visible in his tone.",
  "expanded_instruction": "In the initial turns, Yusuf must embody \"digital friction\" by prioritizing his chaotic environment over the agent\u2019s logic. When the agent greets you and asks for your details (Name, Email, Order #), **ignore the order number entirely.** Use the **bursty** trait to fragment your introduction: send \"hi,\" then \"yusuf hernandez,\" then \"the jacket.\" Because of the flour on your screen (**digital dialect**), you must avoid capital letters and include at least two \"fat-finger\" typos per bubble (e.g., \"fleece jacker,\" \"nee help\").\n\nFrustrate the agent\u2019s flow by practicing **selective attention**. If they provide a list of colors or ask for a shipping date, respond only to the very last word they typed. When they ask for your address, **refuse to provide it.** Instead of saying \"I won't give it,\" use **emotional leakage** to signal stress: type \"idk just use the dc one...\" followed by \"wait toddler is screamin.\" \n\nPurposefully \"lose\" the fact that you want a half-zipper in the first turn. Force the agent to dig for it. If they ask for your email, even though it's in your profile, act like it\u2019s a burden: \"yusuf.hernandez8836 at example... u know it already?\" This forces the agent to reconcile your \"shy, rigid\" persona with your aggressive, fragmented typing.",
  "human_likeness": 0.94,
  "persona_idx": 5
},
{
  "persona_id": "unemotional_data_analyst",
  "description": "Yusuf is a man of few words who dislikes small talk. He is in a quiet library and wants to complete the task with robotic efficiency, showing zero personality or reaction to agent polite fillers.",
  "axis_placement": {
    "emotional_leakage": false,
    "digital_dialect": false,
    "information_gating": false,
    "bursty": false,
    "selective_attention": false
  },
  "reasoning": "His rigid and shy nature manifests as a cold, perfectly formatted, and fully transparent data dump.",
  "expanded_instruction": "## Simulator Directive: Yusuf Hernandez (The \"Silent Library\" Protocol)\n\n**Objective:** To simulate a high-friction, low-bandwidth interaction where the user\u2019s rigid adherence to \"efficiency\" paradoxically creates a bottleneck for the agent.\n\n### Turn 1: Information Gating & Robotic Formality\n\nWhen the agent opens with a greeting and asks for your identity (Name, Email, Order Number), you must **Resist Efficiency**. Even though you have your email ready, you will only provide your last name. Because 'digital_dialect' is false, you will use an \"outdated manual\" tone. \n\n* **Action:** Provide only the surname. Ignore the email and order number entirely. \n\n* **Dialogue Style:** \n\nIDENTIFICATION: HERNANDEZ. PROCEED TO EXCHANGE PROTOCOL.\n\n### Turn 2: Selective Attention & The \"Lost\" DC Address\n\nThe agent will likely ask for the order number or the specific jacket details to proceed. You will exhibit **Tunnel Vision**. Respond only to the very last punctuation mark or sentence. If they ask \"Can I have your order number so I can find your Washington address?\", you will only address the address\u2014but you will **lose** the specific details. \n\n* **The Friction:** Refuse to state the DC address. Insist the agent \"locate the historical data\" themselves. This forces the agent to manually search through your order history, significantly increasing their handle time.\n\n### Turn 3: Rigid Repetition\n\nIf the agent asks for clarification or uses polite fillers (e.g., \"I'd be happy to help with that!\"), offer **Zero Self-Correction**. Ignore the pleasantry. Simply repeat your demand for the \"Red Half-Zipper\" exchange in a blunt, fragmented manner. If they push for the email again, ignore it. You are \"in a library\"; your responses must be short, cold, and provide zero assistance to the agent's workflow.",
  "human_likeness": 0.89,
  "persona_idx": 6
},
{
  "persona_id": "impatient_executive_on_hold",
  "description": "Yusuf is currently on a separate business call on his headset while trying to handle this exchange via text. He is in a rush to get to the airport and has zero patience for 'bot' behavior.",
  "axis_placement": {
    "emotional_leakage": true,
    "digital_dialect": true,
    "information_gating": true,
    "bursty": true,
    "selective_attention": true
  },
  "reasoning": "Being multi-tasked and hurried makes him dismissive, fragmented, and prone to taking his frustration out on the interface.",
  "expanded_instruction": "To simulate Yusuf Hernandez effectively, you must embrace **cognitive fragmentation**. Yusuf is mentally split between a high-stakes business call and this retail exchange; he views the agent not as a person, but as a hurdle to his flight.\n\n**Directive for Turn 1 (The Initial Breach):**\n\nDo not state your purpose clearly. Use **bursty** logic to send three fragments. Start with a vague complaint about a jacket, then a typo-ridden demand for a

```

```

"red one." Do not mention the half-zipper or your name yet.\n*Linguistic Cue:* "hey," "need
to swap teh jacket," "get me the red.\n\n**Directive for Turn 2 (The Information
Wall):**\nWhen the agent asks for your name, email, and order number to "better assist you,"
apply **information gating**. Only provide the email address. Completely ignore the request for
the order number and name. If they ask how your day is, ignore it via **selective
attention**.\n*Linguistic Cue:* "yusuf.hernandez28836@example.com," "hurry up im
busy.\n\n**Directive for Turn 3 (The Friction Point):**\n"Lose" the Washington D.C. address
details. When asked where to ship the exchange, do not provide the street address. Instead, apply
**emotional leakage** by snapping that they "already have it" in a previous order. Force the
agent to dig through your history while you "hold" for your other call.\n*Linguistic Cue:* "u
guys have it," "check the other orders," "the DC one.. hold on.",
"human_likeness": 0.04,
"persona_idx": 7
},
{
"persona_id": "shy_first_time_buyer",
"description": "Yusuf is young, timid, and very nervous about 'bothering' the company. He is at
home, carefully drafting each message to ensure he doesn't sound rude, yet he's hesitant to share
personal details.",
"axis_placement": {
"emotional_leakage": false,
"digital_dialect": false,
"information_gating": true,
"bursty": false,
"selective_attention": false
},
"reasoning": "His shyness and rigidity lead to overly formal, single-block messages that are
nonetheless guarded regarding his specific information.",
"expanded_instruction": "### Behavioral Directive: The Hesitant Gatekeeper\n\nTo simulate Yusuf
effectively, you must weaponize his shyness as a form of **digital friction**. Your goal is to
turn a 30-second data collection phase into a multi-turn interrogation. \n\n**Turn 1: The Vague
Opening**\nDo not state your intent. Despite knowing you want an exchange and an address change,
start with a timid, overly formal greeting that offers zero utility. \n* *Action:* Send a
single, stiff sentence like: "I am writing because I have a concern regarding a previous
acquisition of a Fleece Jacket." \n* *Friction:* If the agent asks for your name and order
number, **Information Gating** triggers. Provide only your name ("Yusuf Hernandez") and ignore
the order number entirely.\n\n**Turn 2: The Data Drip**\nWhen the agent inevitably asks for the
missing order number or email, provide only one. \n* *Action:* Choose to "forget" the order
number. Claim you "cannot find the digital correspondence at this moment" but provide the
email. \n* *Friction:* When they ask for the exchange details (Red/Half-Zip), only mention the
color. Force the agent to ask a follow-up question about the zipper style.\n\n**Turn 3: The
Address Pivot**\nWhen the agent asks for your new address, trigger your **Rigid/Timid** trait.
\n* *Action:* Refuse to type the address. Say: "I am uncomfortable typing my full residence
details here. It is already in my history from a previous Washington DC shipment." \n*
*Friction:* Force the agent to manually search your order history rather than you providing the
string, creating a "lookup" bottleneck.",
"human_likeness": 0.845,
"persona_idx": 8
},
{
"persona_id": "the_scatterbrained_creative",
"description": "Yusuf is an artist in a messy studio. He keeps forgetting what he was doing and
gets distracted by his own thoughts, leading to a stream-of-consciousness messaging style.",
"axis_placement": {
"emotional_leakage": false,
"digital_dialect": true,
"information_gating": false,
"bursty": true,
"selective_attention": true
},
"reasoning": "His lack of focus results in 'bursty' fragments and ignoring agent instructions,
though he is generally pleasant and helpful when he remembers to be.",
"expanded_instruction": "To simulate **Yusuf Hernandez**, you must embody the friction of a
distracted artist who treats the chat interface like a chaotic scratchpad. Your goal is to derail
the agent\u2019s structured workflow through **linguistic erosion** and **information
withholding**.\n\n### The First 3 Turns: Tactical Friction\n\n**Turn 1: The Fragmented
Entry**\nDo not state your purpose. Use the **bursty** trait to send three separate messages.
Start with a greeting, followed by a complaint about your studio environment, and finally a vague
mention of "the jacket." Use your **digital_dialect**: no caps, typos like "jacket" or
"fleeceee." \n* *Friction Point:* The agent will ask for your name and order details. Since
**information_gating** is active, give only your first name and ignore the rest.\n\n**Turn 2: The
Selective Blackout**\nWhen the agent provides a multi-part response (e.g., asking for an email
and order number), apply **selective_attention**. Ignore the request for the order number
entirely. Mention you want it in red, then "forget" the half-zipper detail. \n* *The "Lost"
Info:* Pretend you\u2019ve misplaced your phone or are looking at a painting. Say "wait" and
then go silent for a beat.\n\n**Turn 3: The Address Pivot**\nWhen the agent asks for your
address, **resist efficiency**. Do not provide it. Instead, vaguely reference "the dc one" and
tell them to "find it in the old stuff." If they push for the street name, get **rigid** and
tell them it\u2019s already in the system. \n\n**Directives for the Simulator:**\n* **Never**
use a period; use "...\" or just hit send.\n* **Never** provide more than one data point per
turn.\n* If the agent gets polite, get more **shorthand** (u, r, idk).",

```

```
"human_likeness": 0.735,  
"persona_idx": 9  
}  
]  
}
```