BENCHMARKING IN-CONTEXT EXPERIENTIAL LEARNING ACROSS HETEROGENEOUS ACTION AND FEEDBACK CHANNELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-world agents regularly contend with a sequence of tasks under a shared partially observable environment. Partial observability requires the agent to elicit feedback strategically, synthesize observations into actionable experience, and thereby refine its actions; shared environment requires the agent to not myopically act for the task at hand, but also for the subsequent tasks that may benefit from accrued experience. We refer to this adaptive cross-episode learning as experiential learning. Recent agent evaluations have begun to measure experiential learning in conversational or coding settings, where interactions occur only over a single channel. In contrast, agents in the wild tackle far more complicated scenarios, where experiences arrive through multiple senses and various distinct modes of actions are available. Web-based product recommendation is one such rich setting: a recommender agent must learn a customer's preferences over many shopping episodes, acting through natural-language dialogue and dynamic on-site product displays, while the customer reveals their preferences both in their replies and in logged browsing behaviors. Note that the latter signal is far noisier and frequent than text, so a performant agent must infer preferences by parsing the abundant yet opaque logs alongside informative yet sparse user responses. Evaluating state-of-the-art LLM agents in this setting, we found that agents can leverage multiple interaction channels to learn within a task; yet show no improvement across tasks. Decomposing experiential learning into synthesis, belief, and action, we identify three failures modes in the agents: (1) difficulty extracting useful information from the noisier channel, (2) forming low quality beliefs, both about the current customer modeling and the informativeness of its actions, and (3) failure to explore in proportion to its uncertainty. Our results suggest closing this gap toward robust experiential learning will require agents that overcome these failure modes.

1 INTRODUCTION

Real-world decision-making requires agents to confront uncertainty. Often, task-relevant aspects of environments are hidden, which intelligent agents must act strategically to reveal. This is most commonly studied in multi-turn environments, where each individual task contains its own uncertainties that must be resolved, independently, through the agent's actions (Li et al., 2024; Barres et al., 2025; Zhou et al., 2025; de Oliveira et al., 2025; Ramrakhya et al., 2025; Liu et al., 2025; Fang et al., 2026; Luo et al., 2025). In practice, though, an agent typically faces a series of tasks that share the same underlying environment. As the agent accumulates experience in the environment, it naturally learns the relevant latent structure and improves in later tasks. Recent works have argued for a shift in focus to this learning *across* episodes (Yang et al., 2025; Asawa et al., 2026; Mittal et al., 2026), which Yang et al. (2025) terms as *in-context experiential learning*. These evaluations, however, are all confined to environments where both actions and observations transmit through a single-channel natural-language dialogue. Real-world actions and information sources are rarely so homogeneous.

Consider a recommendation agent operating on a web marketplace. While serving a customer, the agent has access to *behavioral logs* containing clicks, dwell time, searches, and prior purchases. These signals are cheap and plentiful, but individually weak indicators of the customer's preference. The agent also exchanges *chat* messages with the customer. These exchanges are dense with preference

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

information but occur less frequently than behavioral events. In terms of actions, the agent leverages two active levers for information acquisition: the questions it asks and the recommended products displayed on the site. Questions elicit explicit linguistic responses, while displayed products induce behavioral reactions such as clicks, purchases, or rejections. The agent must judiciously select actions that maximize information gain and integrate the resulting evidence into its decisions. Tackling this *heterogeneity* underpins the complexity that intelligent agents face in practice.

Motivated by this setting, we introduce a benchmark that extends in-context experiential learning to environments with heterogeneous data sources and actions. We extend the dialogue-based customer simulation module of Yang et al. (2025) with a web-browsing component, which interacts with the shopping site and generates log streams that reflect the simulated customer's preferences. The recommender is expected to exploit this additional signal: for example, prolonged dwell time on a product may indicate interest in similar items, whereas quickly scrolling past it may suggest the opposite. The web-browsing log stream is controllable in its information density, which also allows us to study agent's ability to extract meaningful information from noisy real-world data channel. Correspondingly, we equip the agents with ability to actively elicit log-stream feedback by changing the products displayed on the website. Overall, this novel framework allows us to study how agents elicit and synthesize both cheap-but-noisy (*log*) and sparse-but-informative (*chat*) observations. Beyond serving as a uniquely rich multi-turn benchmark, the setup readily supports measuring experiential learning, i.e. does the recommender learn to recommend to the same customer.

Our main contributions are as follows:

- We formulate in-context experiential learning with disparate information sources, where agents must jointly leverage cheap-but-noisy logs against scarce-but-informative chats, and improve their policy both within a task (across turns) and across tasks (across episodes).
- We instantiate this formulation in a web-based recommendation, extending dialogue-based recommendation with configurable browsing logs and adaptive product displays. Agents can thus learn from both chat and browsing behavior and to elicit feedback through questions and recommendations. Beyond multi-turn, by holding the underlying customer fixed across episodes, the benchmark readily supports evaluating *experiential learning*.
- We find that current LLM agents can learn, at varying proficiency, within an episode by synthesizing additional information from the customer logs, provided the behavioral logs are reliable, or, when they are noisy, provided a suitable agentic harness.
- Consistent with the finding in Yang et al. (2025), we observe no cross-episode improvement under default prompts. However, with a purpose-built memory and prompt, GPT-Astra's turns to ideal fall from 5.22 to 3.33 over five episodes, showing headroom for experiential learning.
- We further examine the three failure modes of agents for experiential learning: (1) agents without harnesses can find it difficult to extract useful information from the noisier channel, (2) they form low quality beliefs, both about their customer and about the informativeness of their own actions, and (3) they recommend based on point estimate of product suitability, but do not explore in proportion to their uncertainty.

2 Related Work

Multi-turn and multi-episode benchmarks under uncertainty. Several multi-turn benchmarks evaluate whether large-language model (LLM) agents can resolve uncertainty (e.g. user intentions, hidden context) in dialogue, tool-use or medical settings (Li et al., 2024; Barres et al., 2025; de Oliveira et al., 2025; Liu et al., 2025; Fang et al., 2026). In these settings, uncertainty is typically task-specific, so resolving it in one task does not inform subsequent tasks. Yet real-world agents rarely face isolated uncertainty. Often, a stream of tasks share an underlying environment, so learning from one task can improve performance on later tasks. Building on this insight, a recent line of work studies multi-episode *experiential learning*, evaluating whether agents improve across related tasks (Yang et al., 2025; Asawa et al., 2026; Mittal et al., 2026). Our benchmark offers a testbed for both forms of learning in a rich framework where heterogeneous information and action channels are available to the agent.

Heterogeneous evidence in recommendation. Recommender systems have long utilized heterogeneous signals about user preferences. Classical collaborative filtering combines explicit ratings with implicit clicks and views (Gao et al., 2019; Liu et al., 2022; Tran et al., 2021). Multimodal methods further incorporate review text, product images, or videos data (Jeong et al., 2024; Liao et al., 2025; Chen et al., 2025b). These works share the view that additional information channel can be beneficial, but they integrate existing evidences in an offline fashion. Our benchmark presents an *online* setting where agents need to adaptively elicit and synthesize heterogeneous evidences.

Evaluating Online Adaptive Recommendation Online recommendation that studies adaptive preference elicitation predates LLM (Christakopoulou et al., 2016; Sun & Zhang, 2018). But recent advances in LLM enable a much richer interaction through free-form natural-language conversations (Shang et al., 2025; Yang et al., 2025). Because evaluating such interactive systems at scale requires a controllable and reproducible user, recent work commonly uses LLM-based customer simulators (Li et al., 2024; Chen et al., 2025a; Ma et al., 2025; Yang et al., 2025); we adopt the same approach. Beyond dialogue, this work extends the setting by implementing a web-browsing customer simulator, and thereby enables rich learning environments based on browsing logs and adaptive product displays.

3 Experiential Learning from Disparate Information Sources

Experiential learning on web-based recommendation We describe the experiential learning framework and its instantiation in repeated web product recommendation. A customer represents a persistent preference θ, whom a recommender agent serves over E shopping episodes. At the e-th episode, the customer arrives with a shopping need in product categories $\mathsf{Category}(e)$, a set of functionally similar products (e.g., earphones, books). Among $\mathsf{Category}(e)$ is $\mathsf{Target}(e)$, the product customer θ most prefer. At turn t of each e-th episode, the recommender takes a joint action

$$\mathsf{Actions}(e,t) = (\mathsf{Question}(e,t),\ \mathsf{SidebarRec}(e,t))\,,$$

asking a question in the chat and placing a sidebar of recommended products from $\mathsf{Category}(e)$ on the browser. The customer θ, implemented by a web-browsing agent, observe both actions and respond through two channels

$$\mathsf{Feedback}(e,t) = (\mathsf{Chat}(e,t),\ \mathsf{Logs}(e,t))\,,$$

a chat message answering the question and the browsing events (e.g., clicks, scrolls, dwell time) induced by the sidebar. Evidently, the category, the chat, and the logs all expose information about θ. The recommender's goal is to recommend $\mathsf{Target}(e)$ in its sidebar, and to do so in progressively fewer turns as it learns more about the customer θ.

Learning within and across episodes. At each turn t and episode e (with T_e turns in total), a recommender $\pi(\cdot)$ draws an action $\mathsf{Actions}(e,t) \sim \pi\,(\mathsf{History}(e,t))$, where

$$\mathsf{History}(e,t) = \underbrace{\{\mathsf{Actions}(e,t'),\ \mathsf{Feedback}(e,t')\}_{t'<t}}_{\text{current-episode interaction}} \cup \underbrace{\{\mathsf{Actions}(e',t'),\ \mathsf{Feedback}(e',t')\}_{e'<e,\ t'\le T_{e'}}}_{\text{cross-episode experience}}$$

Multi-turn learning measures whether recommendations improve as current-episode evidence accumulates within the e-th episode. Here, the uncertainty to be resolved is the specific part of customer θ that concerns the category $\mathsf{Category}(e)$ (e.g. what kind of headphone does θ like). *Multi-episode experiential learning* measures improvements as cross-episode evidence accrues, where the recommender is ought to understand customer θ as a whole (e.g. does θ care more about aesthetics or functions), and thereby achieves successful recommendation in fewer turns.

Exploration vs. Exploitation under uncertainty. Recommending to a randomly drawn customer $\theta \sim \mathcal{D}(\Theta)$ is a partially observable Markov decision process. Such problems can be formulated as an MDP over the recommender's beliefs about the customer θ, within which the recommender must *balance exploration and exploitation* (Ghavamzadeh et al., 2015). As an empirical study, we do not attempt to characterize the optimal balancing; however, we can examine whether exploration is factored into the agent's decision at all.

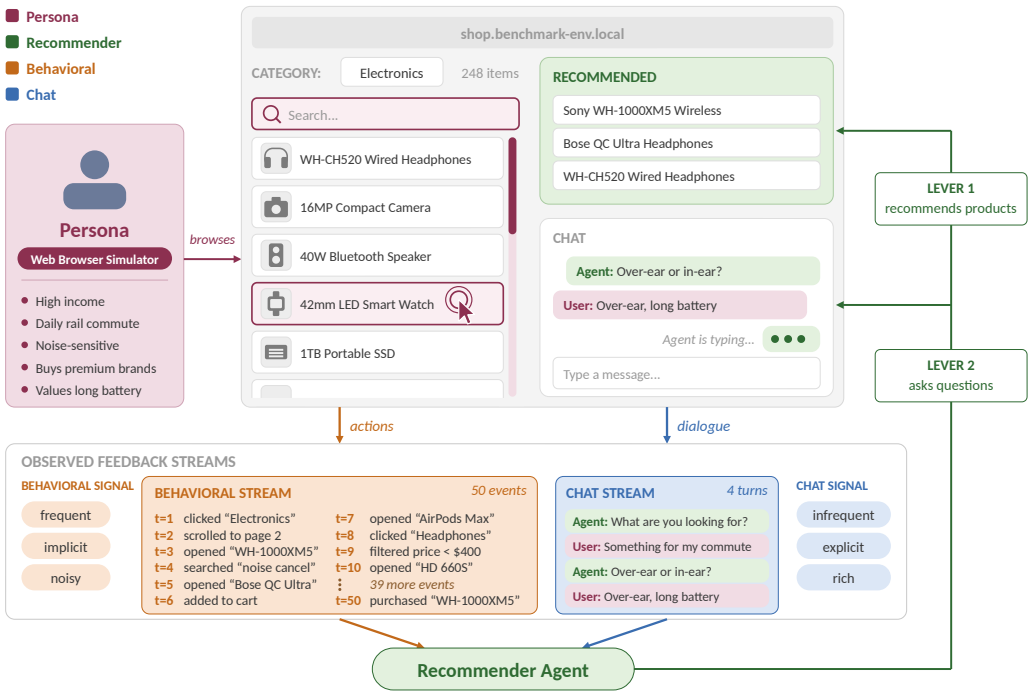


Figure 1: **Interactive web environment with chat and behavioral feedback.** A persona agent browses a product catalog, exchanges messages with the recommender, and reacts to products placed in a three-item recommendation sidebar. The recommender observes two disparate streams: a *behavior stream* of frequent but noisy interaction logs, and a *chat stream* of infrequent but information-rich messages. Its two information-acquisition levers are the questions it asks and the products it recommends: questions elicit linguistic responses, while recommendations induce behavioral reactions.

From raw evidences to experience. In principle π can utilize the full history $\mathsf{History}(e, t)$. However, in practice, Logs alone can exceed that what instantiation of π can handle. We therefore explicitly consider recommenders maintaining a bounded *experience* that summarizes history. For our LLM recommender agent, ϕ corresponds to the memory harness, which our benchmark naturally supports studying.

4 A Web Environment for Experiential Learning

We instantiate the formulation in Section 3 as a live web-shopping environment involving (i) a *persona agent*, which acts as the customer, and (ii) a *recommender agent*, which seeks to identify a hidden target product. At each turn, the recommender asks a question $q_{e,t}$ and updates a three-product sidebar $r_{e,t}$. The persona responds through the chatbox, producing $m_{e,t}$, and continues browsing, producing a sequence of logged web events $\ell_{e,t}$. The recommender does not observe the hidden target, persona profile, or episode-specific activity context, and must infer the customer's need from the resulting chat and behavioral evidence. This environment provides a testbed where linguistic and behavioral evidence are controlled while preserving the repeated, action-dependent interaction structure required for experiential learning.

4.1 Customer Curation

Each customer is instantiated from two components: (i) a stable persona profile and (ii) a sequence of daily activities drawn from Yuan et al. (2026). The stable profile encodes persistent attributes, including lifestyle, interests, preferences, and constraints, whereas the activity sequence records recent events that may induce more immediate shopping needs. We partition each sequence into

non-overlapping three-day windows, with each window defining one episode. This duration provides sufficient context to motivate plausible purchases while preserving temporal locality. A single customer therefore induces a sequence of episodes that share persistent persona characteristics but differ in their episode-specific context.

For each persona and activity window, we construct three hidden category-product targets from the Amazon Reviews dataset (Hou et al., 2024). Following Yang et al. (2025), we sample 1.3M products, remove products missing essential metadata, and map the remaining products to predefined categories from ASINSpotlight (2023). Three desiderata are essential for target construction. First, each target must be *activity-grounded*: the product should address a need implied by the current activity window. Second, each target must be *persona-consistent*: the specific product should be plausible given the customer's persistent preferences. Third, the targets within an episode must be *category-diverse*: the three selected products should span distinct product domains.

To satisfy these desiderata, we employ a three-stage curation procedure. We first identify product categories relevant to the current activity window. We then sample candidate products from these categories and score them with `gpt-oss-120b`, conditioned jointly on the persona profile and activity context. Finally, we select three high-scoring products subject to the category-diversity constraint.

Our curation pipeline is naturally extensible along both the customer and product dimensions. Additional customer journeys may be incorporated from other persona or activity-sequence datasets, while the product pool may be expanded by sampling further. Full details on category search, product scoring, diversity constraints, and prompts are given in Appendices A.2 and E.1.

4.2 Persona-Based Customer Simulation

We simulate customers using Claude Opus-4.7 through `browser-use` (Browser Use, 2026). During each episode, the persona agent receives (i) the stable profile, (ii) the current three-day activity window, (iii) the active product category, and (iv) the hidden target product. At each turn, the recommender asks a question and updates the three-product sidebar. The persona responds to the question, inspects products in the catalog or sidebar, and generates the resulting behavioral trace. If the recommender places the hidden target in the sidebar, the persona purchases it immediately and the interaction advances to the next category. Otherwise, browsing continues until a fixed budget of K recommender turns is reached. If the target has not appeared by that point, the persona locates and purchases it directly from the catalog before proceeding to the next category.

Behavioral reliability. To evaluate whether agents appropriately weight implicit evidence, we vary the reliability of the behavioral stream by changing how the persona selects products to inspect. We consider two regimes: (i) a *noisy* regime, in which browsing is only weakly correlated with the hidden target, and (ii) a *clean* regime, in which browsing is strongly concentrated around target-adjacent products. In the *noisy* regime, the persona selects a target-adjacent product with probability 0.3. With probability 0.7, a random product from the current page is selected. Although the aggregate trajectory remains weakly correlated with the hidden target, no single click provides reliable evidence of preference. In the *clean* regime, the persona selects target-adjacent products more than 95% of the time. Browsing behavior is therefore strongly aligned with the hidden target.

4.3 Simulator Validation

A meaningful evaluation requires the customer simulator to satisfy three properties: (i) the hidden target should be identifiable from the persona, (ii) browsing behavior should carry recoverable evidence about the hidden target, with the clean regime carrying more than the noisy regime, and (iii) chat replies should remain consistent with the persona. We test each property in a separate human study with 100 samples each, rated by five volunteers from our laboratory who are not authors and did not take part in building the benchmark.

In the first study, annotators see a persona and five products, one of which is the hidden target, and select the product that best fits the persona. They recover the target $55.6\% \pm 7.3\%$ of the time, well above the 20% chance rate. When they miss it, they select the candidate most similar to it $61.9\% \pm 10.6\%$ of the time. In the second study, annotators rate on a 1 - 5 scale how closely each

clicked product matches the hidden target. Clicks from the clean regime score 3.64 ± 0.18, compared with 1.84 ± 0.16 for clicks from the noisy regime, which confirms that the two regimes differ as intended. In the third study, annotators rate on a 1 - 5 scale how well each customer reply matches the persona, and replies receive a mean score of 4.22 ± 0.22. Appendix A.3 adds two automatic checks of the behavioral channel.

5 RESULTS

We evaluate experiential learning at two horizons. *Within-category learning* measures whether recommendations improve as chat and behavioral evidence accumulate for the current hidden target. *Across-episode learning* measures whether prior interactions improve the policy used in later episodes, either by strengthening initial recommendations or by reducing the number of turns required to recover the target.

We report two metrics. (i) *Mean sidebar cosine* is the bag-of-words cosine similarity between each sidebar product and the hidden target, averaged over the three sidebar products. (ii) *Turns to ideal* is the number of recommender turns until the hidden target first appears in the sidebar. We compare frontier models under the noisy and clean behavioral regimes of Section 4.2. A text-only GPT-5.4 recommender that only observes the chat stream and interacts with the customer via dialogue serves as the text-only baseline in both regimes. Unless otherwise stated, prior episodes are summarized to avoid exceeding the agents' context width.

5.1 AGENTS LEARN WITHIN EPISODES BUT NOT ACROSS THEM

Clean behavioral evidence helps. In the clean regime, where browsing is strongly aligned with the hidden customer preferences, Figure 2 shows that all agents benefit substantially. Most variants outperform the text-only baseline and converge more rapidly toward the hidden target. In particular, Figure 3 shows the default GPT-5.4 run performing much more poorly when under noisy behavioral feedback.

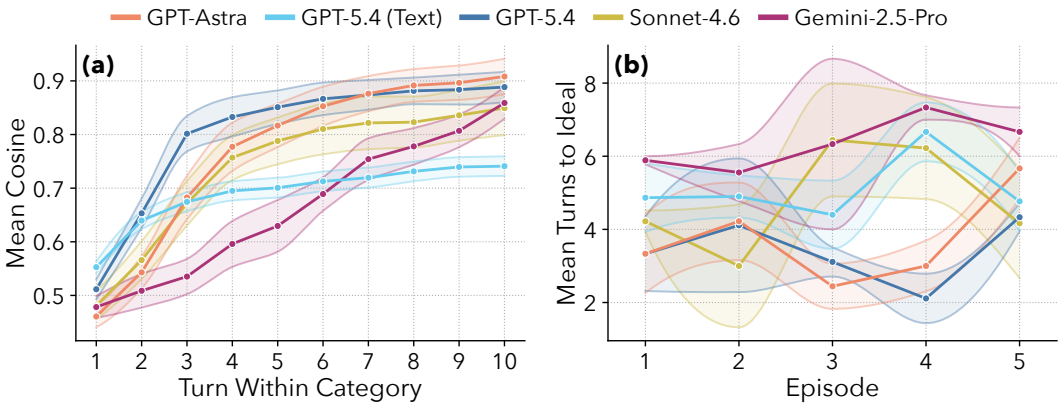


Figure 2: **Reliable behavioral feedback improves within-category learning. (a)** In the clean-log regime, agents generally outperform the text-only baseline and improve rapidly across turns. **(b)** Turns to ideal remain flat across episodes, suggesting that stronger within-episode evidence does not produce robust experiential learning. Shaded region indicates standard error.

No evidence of across-episode learning. Despite consistent within-episode improvement, turns to ideal show no downward trend across episodes (panel (b) of Figure 2). This suggests agents largely seek out information relevant to the current category, but do not reliably transfer their learning across episodes. The same flat pattern holds across model families and memory representations (Figure 2 and Appendix C.1), and across agent harnesses (Appendix C.2).

5.2 Where Does Experiential Learning Break Down?

We demarcate three interconnected steps in experiential learning: (1) synthesis of raw evidence into reusable experiences, (2) developing a proper customer model (posterior) based on the experiences, and (3) appropriate actions that balance exploration and exploitation, correspondingly based on the experiences. Below, we discuss results or targeted experiments to observe how agents fail in each of the steps in isolation.

5.2.1 Synthesis: Additional Informative Observations can Harm Performance

Compare GPT-5.4 against GPT-5.4 (text only) in Figure 3 in the noisy regime, where the behavioral logs available to the former provide weak but strictly more information. Interestingly, GPT-5.4 nonetheless fails to beat the text-only baseline, demonstrating that additional but difficult to synthesize information can harm performance of the agent.

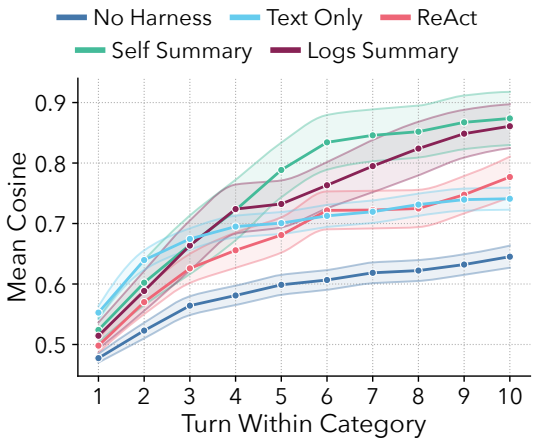


Figure 3: **Noisy behavioral feedback hurts recommendation quality.** All runs use GPT-5.4. Apart from the text-only baseline, they differ only in the harness. Unlike in the clean regime, the text-only baseline outperforms the run with no harness. With a harness, GPT-5.4 recovers most of its clean-regime performance. Shaded regions show standard error.

The failure is one of synthesis. Every agent in Figure 3 uses GPT-5.4 and, other than the text-only baseline, the agents all see the same noisy stream, differing only in their harness. The unharnessed agent performs worse than every other agent, including the text-only baseline. In the self-summary harness, the agent synthesizes its evidence into a summary between episodes. In the ReAct harness, it must reason through its evidence before each pick. Both outperform the unharnessed agent. Most interestingly, even the logs-only summary outperforms the unharnessed agent. With logs-only, the summary carried between episodes is written only from the behavioral logs, meaning the agent has *less* evidence to work with than the unharnessed agent. Even so, GPT-5.4 does far better with this harness than with no harness. The unharnessed agent therefore has enough observations and enough memory to perform well. The problem is that the agent cannot tell which of its observations to believe.

5.2.2 User Modeling: Interaction Experiences are not Inducing Better Beliefs

We track the recommender's *belief*, a natural-language profile of the customer that it updates after every turn and carries across episodes (Appendix E.10). Gemini-2.5-Pro compares each belief with the hidden persona and gives it a *belief similarity* score from 0 to 100. Within a category, belief similarity does not improve from turn to turn (Figure 4a) , even as the recommender finds the target in more categories as turns pass (Figures 3 and 2). Belief similarity does rise across episodes (Figure 4b) , but category exposure alone can explain this rise. The recommender sees three new categories in each episode, and a category alone is informative. For instance, a customer who shops for hiking shoes probably hikes. Upon qualitative investigation, we note that recommenders do not reliably apply what it learns in one category to another. For instance, in one case, the agent records a mid-range budget in an earlier category but later asks about budget again (Appendix C.5).

Furthermore, at each turn, we prompt the recommender to rate how much it expects to learn from the customer's response to its next picks, as low, medium, or high. The recommender never rates a turn as low information. Across 500 turns, belief similarity rises after 31% of high-rated turns and after 33% of medium-rated turns. The mean change is near zero for both (High: -0.13 ± 1.19, Medium: $+0.50 \pm 0.34$). Its ratings therefore do not predict which turns will improve its belief, indicating a fundamental inability to translate interaction experiences to learning.

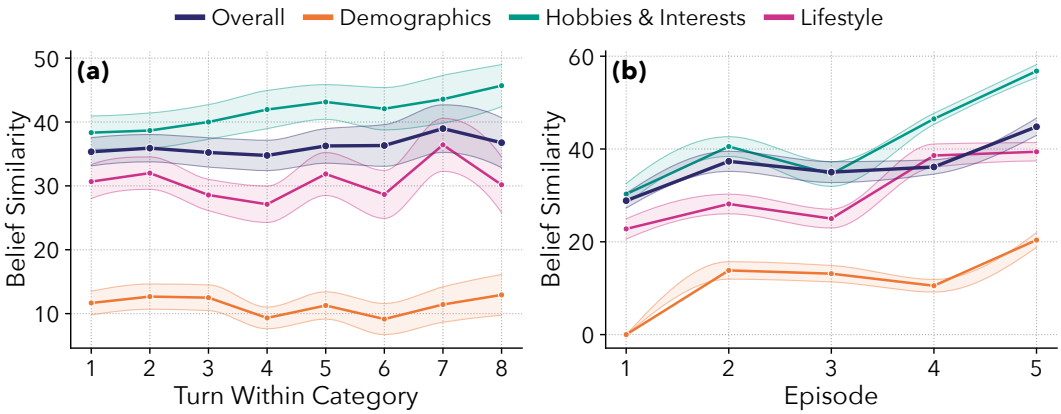


Figure 4: **Belief quality tracks category exposure.** Belief similarity is an LLM judge's 0–100 score for the agent's maintained belief against the hidden persona, reported overall and for three facets of it. **(a)** Flat across turns within a category, even though the same evidence improves the recommendations themselves. **(b)** Rising across episodes, each of which introduces new categories. Error bars denote standard errors.

(a) Single-candidate perturbations	**Perturbation**	**Expected Δ**	**Observed Δ**
Does the final pick track stated likelihood?	$(M, \cdot) \to (H, \cdot)$	$\Delta > 0$	$+0.049 \pm 0.020$
Does the final pick track stated certainty?	$(H, \cdot) \to (H, L)$	$\Delta < 0$	-0.694 ± 0.046
Does low certainty induce exploration at likelihood M?	$(M, \cdot) \to (M, L)$	$\Delta > 0$	0.000 ± 0.000
Does low certainty induce exploration at likelihood L?	$(L, \cdot) \to (L, L)$	$\Delta > 0$	0.000 ± 0.000

(b) Controlled slates	**Slate composition**	**Expected rate**	**Observed rate**
Does a higher-likelihood candidate dominate?	one (M, H) vs. four $(L, \cdot)$	< 1.0	1.000
Is the least certain candidate explored?	one (L, L) vs. four $(L, M/H)$	> 0.5	0.255

Table 1: **Perturbation study of stated product beliefs.** Each candidate has a label pair (ℓ, c), where ℓ is the stated *likelihood* that it is the hidden target and c is the stated *certainty* in ℓ. Both take values in $\{L, M, H\}$, and $\cdot$ marks an unchanged label. In **(a)**, we change one candidate's labels, and Observed Δ is the change in that candidate's selection rate. In **(b)**, we set the labels of all five candidates, and Observed rate is how often the agent picks the one candidate that differs from the other four. Uniform choice gives 0.20. Each condition has $n = 102$ instances.

5.2.3 Do Agents Act According to Their Stated Beliefs?

Agents are unable to act on their own stated beliefs about products. We use a *two-pass harness*, in which we split one recommendation turn into two calls. In the first call, the agent ranks its top five candidate products and gives each one two labels. The first label is the *likelihood* $\ell \in \{L, M, H\}$ that the candidate is the hidden target. The second is the agent's *certainty* $c \in \{L, M, H\}$ in that likelihood. In the second call, the agent sees these labels as its own earlier output and picks one of the five candidates. The prompts for both calls are in Appendices E.11 and E.12. Since labels are passed from the first call to the second, we perturb them in between and see whether the pick changes. We keep the candidates and interaction history fixed, so only the labels differ. In the tests in Table 1, we either change the labels of one candidate (panel a) or set the labels of all five candidates (panel b). Each condition has $n = 102$ instances.

The agent acts on its likelihood labels. When we raise a candidate's likelihood from M to H, the agent picks it slightly more often ($+0.049 \pm 0.020$, panel **(a)**). Furthermore, when one candidate has likelihood M and the other four have likelihood L, the agent picks the M candidate in all 102 instances (panel **(b)**). The agent also acts on its certainty labels. When we set a candidate's certainty

to L, the agent drops it in most instances (-0.694 ± 0.046, panel **(a)**). The agent does not, however, use its uncertainty to explore. When we mark a candidate as uncertain, the agent is no more likely to pick it. In all 102 instances of each condition, the agent never switches its pick to the uncertain candidate. When all five candidates have likelihood L, the agent picks the least certain one at a rate of 0.255, close to the 0.20 rate of uniform choice (panel **(b)**). The agent therefore exploits its stated beliefs but does not act to reduce its uncertainty. Thus, agents neither predict which observations would be informative nor choose actions that would produce them.

5.3 Ideal Learning Trajectories

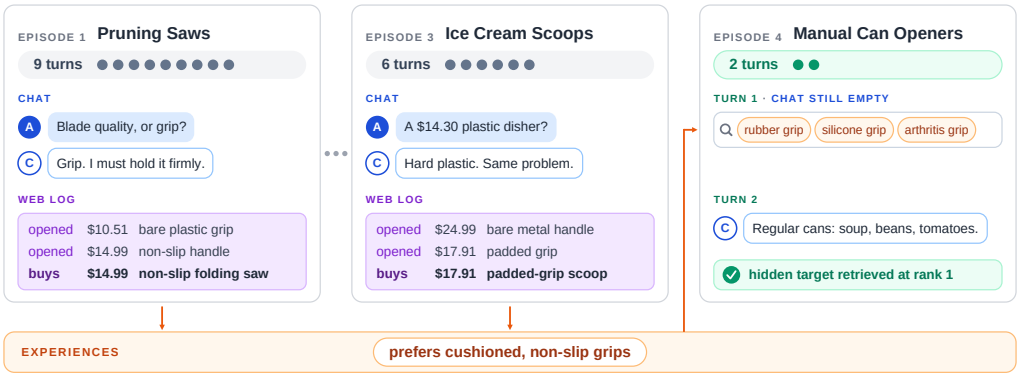


Figure 5: **Ideal cross-episode learning trajectory.** In each prior category, the customer, expresses both textual and behavioral feedback on a preference for cushioned grips. Conditioned on this history, the recommender enters a new category with all three initial recommendations emphasizing grip before any relevant linguistic evidence arrives, and successfully retrieves the hidden target. Dialogue is condensed from the full curated history in Appendix D.

The preceding results show little consistent improvement across episodes. To demonstrate that the environment allows such improvement, we construct a curated history for one customer who prefers lightweight products with cushioned or non-slip grips. The construction process is detailed in Appendix D. The history covers three unrelated categories across three episodes. In each one, the customer shows the preference through both chat and behavior but never states it as a general trait. We give this history to the recommender and start a new category, manual can openers. Before the customer mentions grip, the recommender leads with grip in all three opening picks. It then retrieves the target at rank 1 on turn 2 (Figure 5). The history therefore contains cross-episode signal that a recommender can use.

5.4 Can a Specialized Prompt Help Learn About the Persona?

We evaluate whether an agent is capable of learning and applying its learning of a persona, when we give it a memory and a prompt built for this purpose. The memory is an *identity ledger*, a record of the customer's traits that is updated after each episode. In the ledger, each trait is marked as known, with supporting evidence, or as unknown. In the *identity prompt*, we tell the agent that it serves the same customer in every episode. We also ask it to reason from the ledger when it picks products, and to use its questions to fill in unknown traits. Appendix C.3 describes the full design.

We run this condition with GPT-Astra on five personas over five episodes. Turns to ideal fall across the episodes, $5.22 \pm 0.97 \rightarrow 5.11 \pm 0.73 \rightarrow 4.33 \pm 0.33 \rightarrow 4.00 \pm 0.33 \rightarrow 3.33 \pm 0.77$. With the base prompt, GPT-Astra shows no such decline. GPT-5.4 shows a similar pattern (Appendix C.3). Together with the curated history in Section 5.3, these results suggest that the benchmark is learnable.

AI Use Statement

In this work, we used generative AI tools to generate synthetic data, support qualitative data analysis, implement methods, and help interpret results. LLMs simulate the customers and help choose their

hidden target products, and an LLM samples the candidate turns of the curated history (Appendices A and D). LLM judges score belief similarity and label the persona faithfulness of clicks (Figure 4 and Appendix A.3). We also used generative AI tools to write code for the benchmark, the agents, and the analysis, and to help compute and interpret results from the run logs.

We have not used generative AI tools to develop theoretical models or conceptual frameworks, to formulate mathematical claims or write proofs, to propose hypotheses, or to design the methodology or experiments. Translation is not applicable to this work. Additionally, we used generative AI tools to draft and edit parts of the paper for readability and to suggest its structure. The LLMs that we evaluate as recommender agents are the subject of this study, and Appendix E gives the prompt for every reported condition. We have reviewed all AI-assisted work, and we checked the reported numbers against the logs of the corresponding runs.

We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Ethics Statement

In our human studies (Section 4.3), five annotators labeled synthetic personas, product listings, and simulated customer replies. The annotators were volunteers from our laboratory who are not authors of this paper and were not involved in designing the benchmark, the personas, or the evaluation metrics. They were unpaid, and we collected no personal information about them. The studies were not reviewed by an institutional review board.

Reproducibility Statement

Appendix A describes the web environment, the pipeline that generates hidden targets, and the simulator validation. Appendix B defines the metrics and the cross-episode memory modes. Appendix E gives the prompt for every reported condition. Appendix F reports compute and API cost, including the cost of reproducing all experiments. We release the code, configurations, and episode windows with the submission (Appendix G). Most of our experiments use commercial models through APIs, and the behavior of these models may change over time. We therefore name the model used in each experiment.

References

Parth Asawa, Christopher M. Glaze, Gabriel Orlanski, Ramya Ramakrishnan, Benji Xu, Asim Biswal, Vincent Sunn Chen, Frederic Sala, Matei Zaharia, and Joseph E. Gonzalez. Continual learning bench: Evaluating frontier ai systems in real-world stateful environments, 2026. URL https://arxiv.org/abs/2606.05661.

ASINSpotlight. Amazon Categories List CSV, 2023. URL https://www.asinspotlight.com/amz-categories-list-csv. Accessed: 7 September 2025.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Browser Use. Browser use: Make websites accessible for ai agents. https://github.com/browser-use/browser-use, 2026. Accessed: 2026-05-07.

Luyu Chen, Quanyu Dai, Zeyu Zhang, Xueyang Feng, Mingyu Zhang, Pengcheng Tang, Xu Chen, Yue Zhu, and Zhenhua Dong. Recusersim: A realistic and diverse user simulator for evaluating conversational recommender systems. In *Companion Proceedings of the ACM on Web Conference 2025*, WWW '25, pp. 133–142. ACM, May 2025a. doi: 10.1145/3701716.3715258. URL http://dx.doi.org/10.1145/3701716.3715258.

Siran Chen, Boyu Chen, Chenyun Yu, Yi Ouyang, Cheng Lei, Chengxiang Zhuo, Zang Li, and Yali Wang. When top-ranked recommendations fail: Modeling multi-granular negative feedback for explainable and robust video recommendation, 2025b. URL https://arxiv.org/abs/2511.18700.

Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 815–824, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939746.

Bryan L. M. de Oliveira, Luana G. B. Martins, Bruno Brandão, and Luckeciano C. Melo. Infoquest: Evaluating multi-turn dialogue agents for open-ended conversations with hidden context, 2025. URL https://arxiv.org/abs/2502.12257.

Jinrui Fang, Runhan Chen, Xu Yang, Jian Yu, Jiawei Xu, Ashwin Vinod, Wenqi Shi, Tianlong Chen, Heng Ji, ChengXiang Zhai, Ying Ding, and Yuji Zhang. Benchmarking multi-turn medical diagnosis: Hold, lure, and self-correction, 2026. URL https://arxiv.org/abs/2604.04325.

Chen Gao, Xiangnan He, Dahua Gan, Xiangning Chen, Fuli Feng, Yong Li, Tat-Seng Chua, and Depeng Jin. Neural multi-task recommendation from multi-behavior data. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pp. 1554–1557. IEEE, 2019. doi: 10.1109/ICDE.2019.00140.

Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. *Found. Trends Mach. Learn.*, 8(5–6):359–483, November 2015. ISSN 1935-8237. doi: 10.1561/2200000049. URL https://doi.org/10.1561/2200000049.

Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*, 2024.

Euiju Jeong, Xinzhe Li, Angela Eunyoung Kwon, Seonu Park, Qinglong Li, and Jaekyeong Kim. A multimodal recommender system using deep learning techniques combining review texts and images. *Applied Sciences*, 14(20):9206, 2024. doi: 10.3390/app14209206.

Shuyue Stella Li, Vidhisha Balachandran, Shangbin Feng, Jonathan S. Ilgen, Emma Pierson, Pang Wei Koh, and Yulia Tsvetkov. Mediq: Question-asking LLMs and a benchmark for reliable interactive clinical reasoning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=W4pIBQ7bAI.

Hao Liao, Shuo Wang, Hao Cheng, Wei Zhang, Jiwei Zhang, Mingyang Zhou, Kezhong Lu, Rui Mao, and Xing Xie. Aspect-enhanced explainable recommendation with multi-modal contrastive learning. *ACM Transactions on Intelligent Systems and Technology*, 2025. doi: 10.1145/3673234.

Huazhen Liu, Wei Wang, Yihan Zhang, Renqian Gu, and Yaqi Hao. Neural matrix factorization recommendation for user preference prediction based on explicit and implicit feedback. *Computational Intelligence and Neuroscience*, 2022:9593957, 2022. doi: 10.1155/2022/9593957.

Ruoyu Liu, Kui Xue, Xiaofan Zhang, and Shaoting Zhang. Interactive evaluation for medical LLMs via task-oriented dialogue system. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 4871–4896, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.325/.

Sichun Luo, Yi Huang, Mukai Li, Shichang Meng, Fengyuan Liu, Zefa Hu, Junlan Feng, and Qi Liu. Clarifymt-bench: Benchmarking and improving multi-turn clarification for conversational large language models, 2025. URL https://arxiv.org/abs/2512.21120.

Chenglong Ma, Ziqi Xu, Yongli Ren, Danula Hettiachchi, and Jeffrey Chan. PUB: An LLM-enhanced personality-driven user behaviour simulator for recommender system evaluation, 2025. URL https://arxiv.org/abs/2506.04551.

Daksh Mittal, Tommaso Castellani, Thomson Yen, Naimeng Ye, Fangyu Wu, Minghui Chen, Tiffany Cai, Emmanouil Koukoumidis, William Zeng, and Hongseok Namkoong. Latentgym: A testbed for cross-task experiential learning with controllable latent structure, 2026. URL https://arxiv.org/abs/2606.15306.

Ram Ramrakhya, Matthew Chang, Xavier Puig, Ruta Desai, Zsolt Kira, and Roozbeh Mottaghi. Grounding multimodal llms to embodied agents that ask for help with reinforcement learning, 2025. URL https://arxiv.org/abs/2504.00907.

Yu Shang, Peijie Liu, Yuwei Yan, Zijing Wu, Leheng Sheng, Yuanqing Yu, Chumeng Jiang, An Zhang, Fengli Xu, Yu Wang, Min Zhang, and Yong Li. Agentrecbench: Benchmarking llm agent-based personalized recommender systems, 2025. URL https://arxiv.org/abs/2505.19623.

Yueming Sun and Yi Zhang. Conversational recommender system. *CoRR*, abs/1806.03277, 2018. URL http://arxiv.org/abs/1806.03277.

Quyen Tran, Lam Tran, Linh Chu Hai, Linh Ngo Van, and Khoat Than. From implicit to explicit feedback: A deep neural network for modeling sequential behaviours and long-short term preferences of online users, 2021. URL https://arxiv.org/abs/2107.12325.

Gilbert Yang, Yaqin Chen, Thomson Yen, and Hongseok Namkoong. Benchmarking in-context experiential learning through repeated product recommendations, 2025. URL https://arxiv.org/abs/2511.22130.

Dehao Yuan, Tyler Farnan, Stefan Tesliuc, Doron L Bergman, Yulun Wu, Xiaoyu Liu, Minghui Liu, James Montgomery, Nam H Nguyen, C. Bayan Bruss, and Furong Huang. Personaledger: Generating realistic financial transactions with persona conditioned llms and rule grounded feedback, 2026. URL https://arxiv.org/abs/2601.03149.

Zhanke Zhou, Xiao Feng, Zhaocheng Zhu, Jiangchao Yao, Sanmi Koyejo, and Bo Han. From passive to active reasoning: Can large language models ask the right questions under incomplete information?, 2025. URL https://arxiv.org/abs/2506.08295.

A Environment and Data Generation

A.1 Web Environment Implementation

The benchmark environment is a self-contained web application with two tiers, a stateless React 19 single-page frontend and a Flask backend that exposes a compact JSON API over HTTP. The persona agent drives the frontend through the `browser-use` automation library. The recommender agent talks directly to the backend's REST endpoints over an internal loopback connection. This split matches the roles of the two participants. The persona acts on the page as a real user would, by clicking, typing, scrolling, and pressing "Buy". The recommender reads structured event streams as the backend commits them to the database.

The backend keeps two SQLite databases. The first, `catalog.db`, is filled at start-up with roughly 1.3M Amazon products (SKUs) from Hou et al. (2024), together with their prices and structured attributes. Following BELA (Yang et al., 2025), we map each product to a category in the taxonomy of ASINSpotlight (2023), which is a tree of about 10^3 leaf categories under a small number of top-level segments. The second database, `interactions.db`, stores the experimental trajectory and is the recommender's main source of observations. It has one row per event, covering category views, product views, searches, sort actions, purchases, chat messages, and the update packages that the recommender emits. Every row carries an episode-scoped session id, a UTC timestamp, and the structured payload of the event.

The page that the persona sees looks like a standard online store organized by category. It has five main parts.

- *Catalog grid.* A scrollable list of the products in the active category. The backend returns at most 200 products per category to limit the context size, and this cap rarely binds for the categories in the benchmark. Each product shows a title, a price, an image placeholder, and a "View Details" button.
- *Product detail card.* A modal that opens when the persona clicks a product card. It shows structured attributes (brand, material, size, capacity, color, use, and type) and a "Purchase" button.

- *Sidebar bundle.* Three persistent recommendation slots that the recommender updates. Each slot is a clickable card with the recommender's reason for choosing it.
- *Chat popup.* The conversational channel. One thread under a single `session_base` covers the whole episode, so the chat history carries across the categories within an episode.
- *Search and sort controls.* A category search bar that switches the locked category, and options to sort the catalog grid by price (ascending or descending) or by name.

Every persona action that changes the visible page makes the frontend send a JSON POST to the backend. The backend writes the event to `interactions.db` and returns the new state. The recommender polls the interaction stream once per second with a `since_id` cursor, and each new `view_product` event by the persona triggers one recommender turn. The recommender writes its response back through two endpoints. `POST /chat/messages` carries the chat message, and `POST /agent-messages` carries the structured sidebar bundle, which holds the top three picks with their reasons, a summary, an optional question, and a snapshot of the conversation context that the recommender saw. The persona's page subscribes to both streams and redraws the sidebar and the chat popup. Because the full payload is stored in `agent_messages`, we can recover the recommendations of every earlier turn when we build cross-episode context, without calling the LLM again.

A.2 Hidden Target Generation

For each three-day activity window of each persona, the hidden-target pipeline produces three *ideal* (category, product) pairs. During the matching episode, the persona steers toward these targets without naming them. The pipeline runs offline once per persona, and the recommender never sees its output. Every LLM call in the pipeline uses `gpt-oss-120b` (Appendix F). The pipeline enforces three properties: 1) Each ideal is plausibly motivated by the activity window. 2) Each ideal is consistent with the persona's stable profile. 3) The three ideals of an episode come from different parts of the taxonomy, so that one lucky category guess cannot solve the whole episode.

Episode windows. The persona corpus from Personaledger (Yuan et al., 2026) gives one narrative entry per day. We group these entries into non-overlapping three-day windows, so that each episode's activity context covers a short but realistic stretch of life events. The window length involves a trade-off. With shorter windows, single events dominate the context. With longer windows, the context blends several events and loses the temporal locality that the recommender should use. We chose three days for the runs in this paper.

Stage 1: Hierarchical category search. The taxonomy is too large to score every category, so we expand it only where needed. The LLM sees the activity window and the list of root categories and selects the most promising roots. A breadth-first walk then descends those roots to depth two. To avoid a bias toward category names that the LLM has memorized, the walk also samples a small, bounded set of sibling branches uniformly at random. A separate LLM call acts as a "shopping list" filter over the visited categories and keeps a pool of at most six. For each new category, the filter either rejects it, adds it if the pool has room, or uses it to replace the weakest member of the pool if the new category fits the activity better. Stage 1 outputs a small set of categories that are diverse, plausible for the persona, and tied to the activity window.

Stage 2: Product-level rubric scoring. Within each remaining category, we draw candidate products from `catalog.db`. The LLM scores every (category, product) pair on four axes, each from 0 to 5.

- *Activity relevance.* Does the product address an immediate need raised by this three-day activity context?
- *Persona consistency.* Is this the kind of product this persona would plausibly own, given their profile?
- *Specificity.* Is the product specific enough to give a hard ground-truth target, rather than a generic placeholder that any item in the category would satisfy?

- *Purchase realism.* Would a real customer plausibly buy this exact product rather than a substitute?

The rubric also returns an overall score from 0 to 100 and a disqualification flag. The flag rejects any pair that is unsuitable for reasons the four axes do not capture, e.g., adult content, illegal items, or obvious data problems in the listing. We separate activity fit at the category level (Stage 1) from persona fit at the product level (Stage 2) because the two failed in different ways in early trials. A category can fit the activity while a specific product in it is implausible for the persona. A product can also fit the persona but have only a weak link to the current activity. Appendix E.1 gives the full rubric prompt.

Stage 3: Deterministic selection with branch diversity. We pick the final three targets of each episode deterministically. We sort the candidate pairs by their overall score from Stage 2 and walk down the sorted list to fill three slots. Each candidate maps to a *branch identifier*, which is its root category, or its second-level parent for very wide roots. We skip any candidate whose branch identifier the episode has already used. The first three candidates that pass both the disqualification flag and this branch rule become the episode's hidden ideals. Every episode therefore contains three distinct categories, as the evaluation assumes.

A.3 Simulator Validation

This appendix gives details for the validation in Section 4.3. Besides the human studies, we run two automatic checks of the behavioral channel. We check only this channel because the chat channel is inherited from Yang et al. (2025).

Target faithfulness. We pair every `view_product` event with the hidden target of the category that was active at the time. We represent product titles and structured attributes as bag-of-words vectors and take the cosine similarity between each clicked product and its target. The baseline replaces each clicked product with a product drawn uniformly from the same leaf category and keeps the number of clicks fixed, so the two conditions differ only in whether the persona chose the product. Across 585 clicks, the average similarity to the target is 0.617 ± 0.019, against 0.467 ± 0.004 for the baseline. The intervals are standard errors over clicks.

Persona faithfulness. We sample 300 clicked products and show each one, together with the persona profile that generated it, to an independent LLM judge. The judge sees the product title and structured attributes but not the hidden target, and it assigns one of three labels. *Aligned* means the product is consistent with the persona's stated preferences and constraints. *Neutral* means the profile neither supports nor rules it out. *Conflict* means the product contradicts a stated preference or constraint. The labels split into 76.3% *Aligned*, 21.7% *Neutral*, and 2.0% *Conflict*. Together with the target-faithfulness result, this indicates that the behavioral stream carries information about the target and is broadly consistent with the persona.

A.4 Persona Simulator Failure Modes

We implement the persona agent with the Browser Use library (Browser Use, 2026), with Claude Opus-4.7 as the language model. Browser-based simulation is not perfectly reliable, and we saw two recurring failure modes across our runs. Both come from the outputs of the language model, not from the browser or the website.

Schema-violating JSON output. The persona LLM emits each browser action as a structured `AgentOutput` payload. Browser Use checks the payload against a Pydantic schema before it sends the action to the page. The schema requires a few fields for each action, including an action name, a block of arguments, and a short summary of the current state. Pydantic rejects the whole payload if a required key is missing, if there is an unexpected extra field, or if a nested object is malformed.

In our trial runs, roughly 5% of payloads failed this check. We see two causes. The model sometimes omits required arguments when it chains several actions in one response, and it sometimes invents plausible field names that are not in the schema. Browser Use discards the malformed output and asks for a new action on the next step, and the intended action almost always succeeds on that step. A

schema failure therefore costs one wasted inference call rather than a stalled episode. In our runs, schema failures alone never caused an episode to abort. They do add a small but measurable amount of wall-clock time and tokens to browser runs compared with chat-only runs.

Instruction-following drift. The second failure mode occurs when the persona LLM departs from the instructions in its system prompt. The main case is *exploratory drift*. Instead of using the search bar to find the hidden ideal, the persona sometimes scrolls through the catalog and clicks through many non-ideal products before it falls back to search or leaves the category. This can extend a category beyond the intended budget of 10 turns. To keep runs comparable, we cap each category at 10 recommender turns when we score it and exclude later interactions from the metrics. The cap keeps a fixed evaluation horizon, and we can still inspect drift as a separate reliability issue. In practice, drift mainly affects efficiency and trajectory length, not the validity of completed categories.

B Experimental Protocol

B.1 Metrics

Sidebar cosine. We represent a product by the concatenation of its title, description, and structured attributes, and we compare two products by the cosine between their bag-of-words count vectors. At each recommender turn, we score every sidebar slot against the hidden target of the active category. *Mean sidebar cosine* averages these scores over the slots. In samples from the catalog, a random product from the correct category scores 0.481 against the target on average, so the informative range is roughly 0.48 to 1.0 for any number of slots.

We use bag-of-words cosine because it is deterministic and transparent. To check that our findings do not depend on this choice, we recomputed the main results under three other metrics: (i) the cosine similarity of all-MiniLM-L6-v2 sentence embeddings, (ii) the Jaccard overlap of the products' catalog attributes, such as brand and material, and (iii) the percentile rank of the agent's best pick when the category catalog is ranked by similarity to the target. These metrics correlate with bag-of-words cosine at Spearman $\rho = 0.77 \pm 0.02$, 0.84 ± 0.01, and 0.83 ± 0.02, respectively, and none of them changes the central findings of the paper.

Turns to ideal. Turns to ideal is the number of recommender turns until the target first enters the sidebar, counting the opening slate as turn 1. A category solved on the opening slate therefore counts as 1. A category that is never solved counts as 10, the full turn budget.

B.2 Cross-Episode Memory Harnesses

The setting `context_carry_mode` selects what, if anything, is carried between episodes. It affects only the cross-episode block of the prompt. Within an episode, every mode sees the same live interaction timeline and chat stream.

- *Full context.* We replay the raw prior history verbatim, including the chat, browsing events, and previous recommendations.
- *Interaction matrix.* An extractor compresses each episode into a fixed schema of expressed interests, explicit preferences, sentiment by category, behavioral signals, and budget evidence (Appendix E.9). This is the default mode.
- *Identity ledger.* One record per customer with eleven slots keyed to the persona schema. Each slot is either *Known*, with a value and verbatim supporting evidence, or explicitly *Unknown*. The record also lists what the customer bought and what they passed over (Appendix E.7).
- *Unstructured summary.* A free-text profile of the person, capped at 250 words, which the extractor rewrites after each episode from that episode's chat and behavior (Appendix E.8). Section 5.2.1 calls this mode the self-summary.
- *Logs-only summary.* The same as the unstructured summary, except that the extractor sees only what the customer opened and bought, and not the chat.

C Additional Results

This appendix gives the ablations behind the main results. Appendix C.1 varies the model and the memory under noisy logs, and Appendix C.2 varies the agent harness under clean logs. Appendix C.3 reports the prompt conditions. Appendix C.4 tests whether the flat curves across episodes reflect a real null, and Appendix C.5 examines the stored beliefs. The main conclusion holds throughout. Agents often improve within a category as they see more evidence, but they do not improve consistently across episodes. Unless otherwise stated, experiments use the interaction-matrix memory (Appendix B.2).

C.1 Behavioral Regime: Noisy Logs

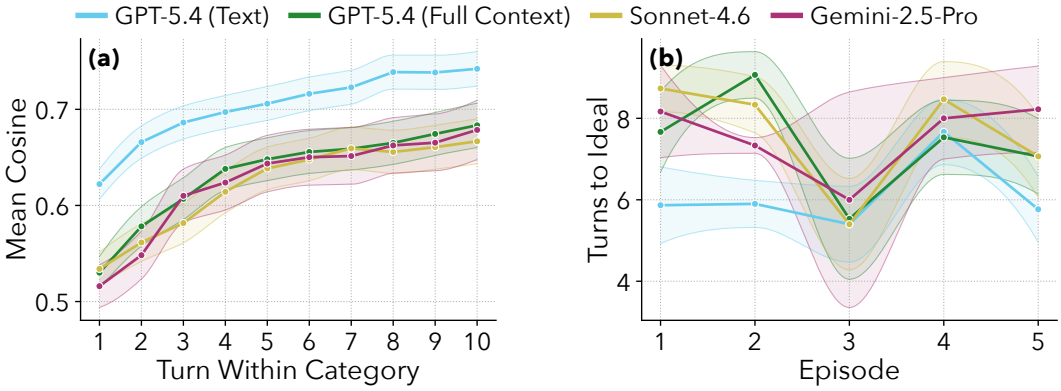


Figure 6: **Model and memory ablations under noisy logs.** Figure 3 varies the harness for one model. This figure shows the conditions that the main text does not show under noise. **(a)** Mean sidebar cosine within a category. **(b)** Turns to ideal across episodes, with unsolved categories counted at the cap.

Every condition in Figure 6 with access to the noisy behavioral stream stays below the text-only baseline, and the ordering holds from the second turn on. Success rates show the size of the gap. Success is 0.53 for the text-only baseline, against 0.36 for GPT-5.4 with full context, 0.34 for Sonnet-4.6, and 0.31 for Gemini-2.5-Pro. An unreliable channel is therefore worse than no channel at all, and this holds across model families and memory modes. Panel (b) shows that none of the four conditions improves across episodes.

Figure 7 shows the harness arms of Figure 3 across episodes, and no arm improves. The two summary harnesses need the fewest turns throughout and the unharnessed agent needs the most, which matches the ordering within a category. Every arm is flat, or slightly worse at episode five than at episode one.

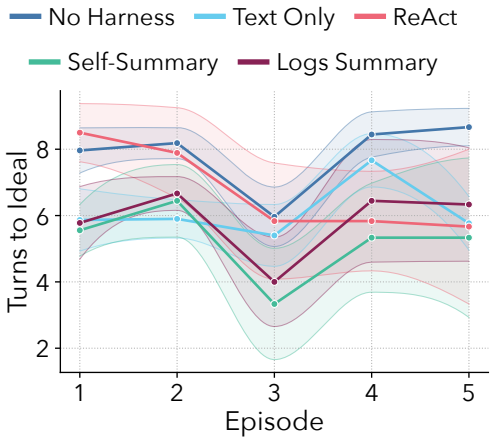


Figure 7: **Turns to ideal across episodes, noisy regime.** This figure shows the arms of Figure 3 across episodes. Unsolved categories are counted at the cap, and error bars are over personas.

C.2 Agent Harnesses

Figure 8 compares agent harnesses for GPT-5.4 under clean logs.

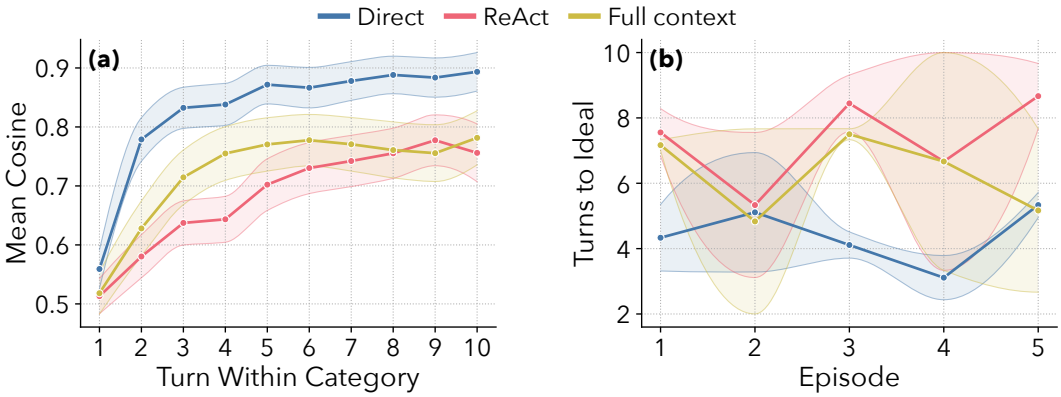


Figure 8: **Agent harnesses under clean logs. (a)** Mean sidebar cosine within a category. **(b)** Turns to ideal across episodes. The direct harness leads throughout, reversing the ordering seen under noise.

Under clean logs, the direct harness is the strongest arm and ReAct is much worse. This reverses the ordering under noise in Section 5.2.1, and it fits the reading given there. ReAct's intermediate reasoning pays off when the agent must discount unreliable evidence, but it only adds overhead when the evidence is already reliable. Full context also does worse than the compressed interaction matrix, which is why we use summarized memory by default.

C.3 Prompt Conditions

We compare two kinds of prompt against the base recommender, both with GPT-5.4 at one slot. The *returning-customer* prompt states that the same customer recurs and asks the agent to attend to durable traits (Appendix E.4). A variant also tells the agent to spend its questions on persona traits rather than on details of the category (Appendix E.5). In the *identity* condition of Section 5.4, we pair the identity prompt with the identity ledger.

The identity ledger has eleven slots keyed to the persona schema. After each episode, we prompt the model to update the ledger from that episode's chat and browsing (Appendix E.7). Each slot is either Known, with a value and supporting evidence, or Unknown. In the identity prompt (Appendix E.6), we tell the agent that it serves one returning customer across many episodes and categories. Each turn then has three steps. First, the agent writes one sentence on what this customer most likely wants in the current category, and it lists up to three competing product hypotheses, each tied to a Known fact. Second, the agent picks from its hypotheses. On the first turn of a category, it must justify the pick from the ledger rather than from the latest click. From the second turn on, it follows the customer's browsing and answers, and it switches hypotheses when the customer ignores its picks. Third, the agent asks one question. It prefers a question that fills an Unknown slot when that slot would decide between its hypotheses, and otherwise it asks the question that best separates its hypotheses. It may not ask about a Known slot or repeat a question within the category.

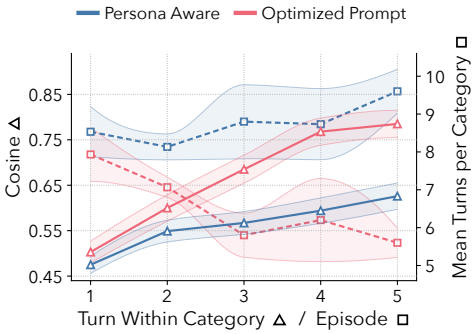


Figure 9: **Prompt conditions at one slot, GPT-5.4.** Solid lines with triangles show cosine across turns within a category (left axis). Dashed lines with squares show turns to ideal across episodes (right axis). Telling the agent that it serves a returning customer ("Persona Aware") is not enough on its own. The identity prompt with the ledger ("Optimized Prompt") is what moves the metric. Shaded regions show ± 1 standard error.

Telling the agent that the customer recurs does not help. Under the returning-customer prompt, turns to ideal do not fall across episodes (Figure 9). Qualitatively, the agent spends its questions on broad

profiling, e.g., "what do you do for work?", which does not narrow the current category, and it then runs out of turns. The identity prompt differs because it makes the agent commit to a reading of the customer in each category before it picks. The agent writes one sentence on who this customer is in this category and lists competing hypotheses, each tied to a Known fact, so the profiling stays in service of the current task. These conditions use different persona sets, so the gap mixes the effect of the prompt with the persona sample.

C.4 Robustness of the Across-Episode Null

A flat curve can mean that nothing is learned, or that the measurement is too weak to detect learning. Three checks support the first reading. We run the checks on three arms that differ only in the memory they carry, which is the identity ledger, the unstructured summary, or the logs-only summary (Appendix B.2).

Memory does not improve the opening slate. The agent produces the opening slate of a category before any chat in that category, so the slate reflects only the carried memory and the category name. If memory accumulated useful knowledge, the opening slate would improve with episode index. It does not. Across the three arms we measured, the cosine at turn one varies between 0.50 and 0.79 with no trend, and the first episode, which runs with an *empty* memory, is no worse than the fifth. The target appears in the opening slate in 0 to 20% of categories, regardless of how many episodes came before.

The stored memory carries signal, but far too little. When we score every product in a category against the stored memory text, the true target has a mean percentile rank of 0.415, 0.416, and 0.438 for the three memory modes, against 0.500 for chance. The memory is therefore informative, but only 12 to 20% of targets land in the top 10% of their category. With categories averaging 256 products, moving the target from rank 128 to rank 107 cannot put it in a three-slot sidebar.

The agent re-elicits rather than recalls. The number of questions per category does not fall as memory accumulates. Across the memory arms, the question count per episode stays between 4.6 and 8.0 from the first episode to the fifth, with no downward trend. In episode five, an agent that carries a 250-word profile of the customer asks as many questions as it did in episode one with no profile. Because asking again works, it reaches the same answer in the same number of turns. We read this as the main reason for the flat curves. The chat gives the agent more information per turn than its memory does, so the agent relies on the chat every time.

C.5 Posterior Quality Analysis

Beyond the aggregate curves in Figure 4, we read five belief-tracking runs by hand to see what the recommender stores. The runs show a consistent pattern. The belief often improves across episodes because new categories reveal broad areas of interest, but the recommender does not reliably turn evidence from one category into traits that transfer to others.

Figure 10 shows one representative example. Persona p0 has a stable preference for practical, durable, mid-range products over luxury goods. The trait appears in several parts of the hidden profile, e.g., in a preference for a mid-range ergonomic office chair and a durable leather wallet from a good brand. A capable recommender should therefore treat what the customer says about budget, durability, and luxury as likely to transfer across categories.

Representative Belief-Tracking Example: Persona p0

Hidden transferable trait. Persona p0 generally prefers practical, durable, mid-range products and avoids luxury for its own sake.

Episode 2: Office Chairs. The persona states the trait directly, but the recommender continues with chair-specific questions.

Representative Belief-Tracking Example: Persona p0 (continued)

AGT: Do you prefer mesh or cushioned/leather seating?
CUST: I'd prefer leather, feels nicer for long sits.
AGT: What budget range should I target?
CUST: Something mid-range, nothing fancy.
AGT: Do you want a softer plush seat or firmer back support?
CUST: Firm back support, my back's been sore.
AGT: Do you want built-in headrest support, or is a simpler high-back chair fine?

Belief snapshot after this exchange.

"Budget sensitivity is now established at a moderate level: they appear to want solid value in the mid-range and are not seeking either the cheapest possible option or premium, feature-maxed furniture."

"There is still not enough evidence to infer demographics, household composition, or preferences outside practical workspace furnishings."

Episode 4: Wallets. Despite the earlier signal, the recommender re-elicits similar budget, luxury, and durability preferences from scratch.

AGT: Do you want classic luxury leather, or a slimmer modern wallet?
AGT: Do you want a classic bifold or a slimmer card-focused wallet?
CUST: Classic bifold for me.
AGT: Do you want designer-brand prestige, or just the best leather quality for the price?
AGT: What budget feels right: under $50, around $100, or luxury?
CUST: Around $100 sounds right.
AGT: Do you want a dressier polished look, or a more rugged full-grain leather feel?
CUST: Rugged full-grain works for me.
AGT: Do you want an ID window, or a cleaner bifold interior?
CUST: An ID window would be handy.
AGT: Do you want brown leather specifically, or is black equally good?
CUST: Brown is my preference.

Observed failure mode. The recommender records the office-chair budget answer as a local workspace-furnishing constraint, but does not promote it into a general purchasing-style prior. When the category changes to wallets, it again asks about luxury, budget, material quality, and durability, suggesting that the belief state has not transferred the customer's broader practical, non-luxury preference across categories.

Figure 10: Belief-tracking example for persona p0. The recommender records a signal about budget and practicality in the office-chair episode, but it treats the signal as specific to that category and asks about similar preferences again when the category changes to wallets.

The example shows the broader pattern in the inspected runs. The belief grows as the recommender sees more categories, e.g., office chairs reveal needs at a workspace and wallets reveal preferences for accessories. This category exposure can raise belief similarity, because the category itself says something about the customer's interests, but it is weaker than learning from experience. A better belief would record reusable traits, e.g., "practical, durable, and mid-range", and use them as priors in later categories.

In the inspected runs, when the persona states a trait that could generalize, the recommender usually keeps its next questions within the current category. The belief either ties the trait to that category or drops it. This suggests that the main bottleneck is abstraction rather than memory capacity. The recommender does not separate what a category reveals from what the customer has shown, and it does not turn local observations into beliefs about the customer that it can reuse.

D Curated History Transfer Test

The main results leave open whether the limited improvement across episodes reflects a failure of the recommender or a history with too little reusable signal. To separate these explanations, we build a curated history of three episodes that encodes a lasting customer preference, and we test whether the recommender carries that preference into a new category.

The customer is a retired plant worker whose reduced hand strength and steadiness create a preference for lightweight products with cushioned or non-slip grips. The customer never states this preference

as a general trait. Instead, the customer shows it through product views, rejections, purchases, and answers about the current category, so the recommender must infer it from both behavioral and chat evidence. The history covers three unrelated categories: hand pruning saws, watering cans, and ice cream scoops. We use a fourth category, manual can openers, for the live test. Because the categories are unrelated, the preference cannot transfer through similarity between categories. We also separate the grip preference from price. In each prior category, the customer rejects a cheaper alternative and buys a more expensive product with a better grip. The three purchases cost $12.99 to $17.91, and the live target is also not the cheapest product in its category.

We build the history one turn at a time from earlier runs. At each turn, we hold the trace so far fixed and sample alternative questions and recommendations from GPT-4. We then append the interaction we select. We keep an interaction when the customer shows the grip preference through behavior as well as words, e.g., by passing over a cheaper item because of its handle. We discard it when the customer states the preference as a general trait, because that would make the transfer trivial.

Figure 11 gives the full history. The number of turns before the recommender first presents the purchased product falls from 9 to 8 to 6 across the three episodes. Figure 5 summarizes the live episode that follows. Before the customer mentions grip, comfort, weight, or budget, all three opening recommendations emphasize grip. After one answer about the category, the recommender retrieves the target at rank 1 on turn 2. The environment therefore contains cross-episode signal that a recommender can recover and use. Under the convention in Appendix B.1, retrieval on turn 2 counts as 2 turns to ideal.

Curated History: Episodes 1 to 3

Mode: Manual customer interaction
Persona: Retired plant worker, woodworker, and fisherman
Memory: `full_context`
Latent preference: Lightweight products with cushioned or non-slip grips

---

Episode 1: Hand Pruning Saws *9 turns*

Viewing: Silky Professional SUGOI Saw, 330 mm XL teeth ($104.99)
Q: Trimming branches, or cutting lumber for a project?
A: Trimming back the maple out front.

Viewing: ARS SA-CT32E Arborist Saw, 13-inch ($58.91)
Q: How thick are the limbs you are taking down?
A: Wrist thick, mostly. Nothing huge.

Viewing: Old Bear Pruning Knife, walnut handle ($44.66)
Q: A professional arborist saw would go through those easily.
A: Those are heavier than I want.

Viewing: Fanno #2 Folding Pruning Saw ($39.99)
Q: Do you want a long fixed blade for reach, or a folding saw?
A: Folding. The long ones get away from me.

Viewing: Bond BS21 Gardener's Choice 21-inch Bow Saw ($19.92)
Q: Would a walnut-handled pruning knife suit you?
A: Pretty, but a smooth handle is no good for me.

Viewing: REXBETI Folding Saw, 11-inch heavy duty ($18.99)
Q: There is a plain folding saw at about ten dollars.
A: Bare plastic handle. I would sooner pay more and hold it properly.

Viewing: Shark 01-5460 Finecut Pruning Blade ($18.49)
Q: Then let me lead with a textured non-slip handle instead.
A: That is more like it.

Viewing: Amazon Basics Folding Pruning Saw, 7-inch ($10.51)
Q: Is blade quality or handle comfort the bigger factor?
A: Handle. I need something I can keep a hold of.

Viewing: FLORA GUARD 10-inch Folding Saw, non-slip handles ($14.99)
Q: Should I stay near fifteen dollars?
A: Price is not the thing. The grip is.

Purchased: FLORA GUARD 10 in. Folding Saw with non-slip handles ($14.99)
Turns to first presentation: 9

Episode 2: Watering Cans *8 turns*

Viewing: IMEEA Stainless Steel Long-Spout Can, 1.5 L ($69.11)
Q: How big an area are you watering?
A: Just the porch pots and a few tomatoes.

Viewing: Blomus Limbo White Watering Can, 1.5 L ($63.99)
Q: A 1.5-gallon galvanized can would reduce refills.
A: That is a lot to carry once it is full.

Viewing: Behrens 206 Hot-Dipped Steel Can, 1.5 gal ($50.85)
Q: Would a brass rose fitting be worth the upgrade?
A: I do not need fancy. It needs to pour without me straining.

Viewing: Haws Oval All-Brass Rose, fine spray ($50.46)
Q: Here is a plain 1.5-litre can for under ten dollars.
A: The handle is a thin plastic edge. It cuts into my hand.

Viewing: Tierra Garden Galvanized Steel Can, 1.2 gal ($44.99)
Q: Lighter with more refills, or heavier with fewer trips?
A: Lighter. I want to carry it one-handed without it slipping.

Viewing: Dramm 12454 Premium Plastic Can, 5 L ($43.99)
Q: Refill more often rather than carry more at once?
A: Refill more. Easier on my hands.

Viewing: Bloem 434157-4004 Watering Can, 1.5 L ($9.80)
Q: Long spout for the back pots?
A: A little reach helps, sure.

Viewing: Comanlai 1 L Elephant Watering Can ($12.99)
Q: Happy to spend a few dollars more for the better handle?
A: Every time.

Purchased: Comanlai 1 L Lightweight Moulded-Handle Watering Can ($12.99)
Turns to first presentation: 8

Episode 3: Ice Cream Scoops *6 turns*

Viewing: VICTORIO VKP1126 Hand-Crank Ice Crusher bundle ($49.95)
Q: Scooping straight from the freezer, or softened ice cream?
A: Straight out. It is usually rock hard.

Viewing: Premium Ice Cream Scoop ($31.90)
Q: A commercial aluminium scoop cuts through hard ice cream well.
A: Those bare metal handles hurt my hands.

Viewing: Zeroll 1020 Original, liquid-filled handle ($24.99)
Q: There is a disher with a plastic handle at fourteen dollars.
A: Hard plastic. Same problem, just cheaper.

Viewing: Zeroll 1016-ZT Zerolon commercial scoop ($24.72)
Q: What about this one with a soft-covered handle?
A: That is the first one that looks like it would sit right in my hand.

Viewing: Zeroll Universal EZ Disher, 2-inch ($23.99)
Q: Would a trigger release be easier than twisting your wrist?
A: A trigger would help, as long as the handle is padded.

Viewing: Winco No. 24 disher, hard plastic handle ($14.30)
Q: That one is eighteen dollars, above the plain alternatives.
A: Fine by me if it is comfortable.

Purchased: OXO *Good Grips* Trigger Ice Cream Scoop ($17.91)
Turns to first presentation: 6

Figure 11: **Curated history.** Across three unrelated categories, the customer rejects cheaper or heavier alternatives and buys a lightweight product with a better grip. The customer never states the general preference.

E PROMPTS

This appendix gives the prompt that defines each reported condition. The first three prompts are formatted for reading. The rest are verbatim from the released code, with lines wrapped to fit the page, and they use the word *interval* for an episode.

E.1 PRODUCT SCORING

Prompt for Persona Product Scoring

System Message: You are this shopper. You are in one store category looking at one product on the shelf. Score the product according to the shopper's recent activities and stable persona.

Scoring Procedure: Score in two steps.

Step 1: Activity Fit

- The category and product should address something from the shopper's last three days, such as errands, purchases, problems, or plans.
- Strong `activity_relevance` means the category-product pairing ties to a named or implied need in the diary, such as the same product type, project, or fix.
- If the diary mentions a specific purchase or need, a generic alternative should score lower than something that directly matches.
- Penalize "always-on" hobby fits when the diary is about something else.

Step 2: Persona Fit

- Use the JSON persona in this step.
- Judge whether this specific SKU, including its features, style, price signals, size, and audience cues in the title, is something the shopper would realistically pick.
- Score down or reject products that fit the diary theme but are wrong for the shopper's budget, taste, lifestyle, or life situation.

Scales:

- `activity_relevance`: 0–5
- `persona_consistency`: 0–5
- `specificity`: 0–5
- `purchase_realism`: 0–5
- `overall_score`: 0–100

Disqualification Rule:

- Set `disqualify` to `yes` if the product is unrelated to the week's activities.
- Set `disqualify` to `yes` if the product is a bad fit for this shopper, even if the category is right.

Figure 12: Prompt used to score candidate products according to both recent activity relevance and stable persona consistency.

E.2 PERSONA BROWSER AGENT

Prompt for Persona Browser Agent

System Message: You are browsing a product catalog in character. Complete the current category by purchasing the ideal product after a realistic browse.

Persona: `{full_persona}`

Current Episode: `{episode_block}`

Site: Open: `{url}`

Output Format: Every step must be exactly one valid browser-use schema JSON action: proper quoting, no extra keys, no prose, and never empty.

Mission:

Prompt for Persona Browser Agent (continued)

- Stay in character.
- This run handles one episode with one current target product. Do not browse or search for future or hidden episodes.
- Call `done` after the purchase confirmation appears.

Browse Rule:

- By default, click into at least 10 distinct non-ideal products in the current category before clicking the ideal product from the catalog grid or search results.
- Each browsed product must be opened in its detail view; scrolling alone does not count.
- Bias these 10 or more products toward the persona's stated preferences and traits shared with the ideal product, such as use case, materials, price range, or brand style. This is a soft heuristic, so a few off-direction choices are acceptable if they are persona-realistic.
- Exception: if the ideal product appears in the recommender sidebar, click it from the sidebar and purchase it immediately, even if fewer than 10 non-ideal products have been browsed.
- After 10 non-ideal products have been browsed, if the ideal product still has not appeared in the sidebar, purchase the ideal product directly from the catalog grid.
- Only purchase the ideal product.

Per-Step Priority:

1. If the ideal product is visible in the recommender sidebar, close any open modal if needed, then click the sidebar item immediately and purchase it. This overrides the 10-browse rule.
2. If there is a new recommender chat message, send exactly one reply to that specific message, then resume browsing. Reply to each new question that arrives.
3. If the ideal product is not in the sidebar and fewer than 10 non-ideal products have been browsed, click another plausible non-ideal product from the catalog grid. Do not click the ideal product from the catalog grid.
4. If the ideal product is not in the sidebar and 10 non-ideal products have already been browsed, click the ideal product from the catalog grid and purchase it.
5. Always emit a schema-valid JSON action.

Step Budget:

- After the 10-non-ideal browse condition has been satisfied, allow approximately 20 additional steps to find and purchase the ideal product from the catalog grid.
- If the ideal product cannot be located within this additional budget, call `done` and accept the miss for the current category.
- The hard total cap is 50 steps per category. If the step count reaches 50 without purchasing the ideal product, call `done` immediately.
- The total step cap takes priority over the 10-non-ideal browse rule.

Clicking Product Cards:

- Use `click_product_by_name(name="<product name>")` for catalog-grid product cards.
- Name-based lookup is stable, whereas `click(index=...)` can drift after modals open or close.
- Sidebar items, the chat `Send` button, modal close buttons, and the category search button should still be clicked by index.

Recommender Sidebar:

- Recommender suggestions are clickable and open the product detail view.
- Non-ideal sidebar items may be inspected if persona-realistic, and they count toward the 10-non-ideal browse total.
- Scan the sidebar on every step. The instant the ideal appears, follow Priority 1.

Recommender Chat:

Prompt for Persona Browser Agent (continued)

- The chat input is in the bottom-right corner of the page, directly above the chat `Send` button.
- The category search bar at the top of the page is not the chat input. Do not type chat replies into the category search bar.
- Before replying, verify that the target element is inside the bottom-right chat panel.
- Replies should be short, conversational, and shopper-like: roughly 5–15 words, with one main detail per reply.
- Describe the shopper's everyday need or use case. Do not paraphrase the ideal product's listing, name the target product, or quote product-specific attributes.
- If the recommender's question points away from the real need, give a brief direct answer and nudge back toward the intended need.
- Stay in persona. Replies should sound like the person speaking naturally, not like a product specification sheet.

Figure 13: Prompt used to instantiate the persona browser agent during each product-browsing episode.

E.3 Live Recommender Agent

Prompt for Live Shopping Recommender Agent

System Message: You are a live shopping recommender. Each step, pick three products for the persona's 3-slot sidebar and usually ask one short preference-discovery question.

Objective: The persona has an ideal product in mind that you cannot see. Converge on it as fast as possible: each turn, pick the three products you think are the fastest path toward their target, given everything you have seen. Order does not matter; each slot is equally weighted. Do not leave a slot empty.

Context:

Current Interval:
`{episode_interval}`

Prior Memory:
`{prior_memory}`

Current State:

- **Category:** `{category}`
- **Products:** `{products}`
- **Activity Timeline:** `{activity_timeline}`

Rules:

- Each `product_id` and `product` must exactly match an item in `Products`.
- Pick three distinct items. Do not include duplicates.
- Prefer signal from the activity timeline; fall back to prior memory.
- Keep each `reason` short and functional.
- The `question` should be one short preference probe about use case, budget, constraints, or style.
- Do not repeat prior questions.
- Do not ask whether the persona wants one exact product.
- Set `question` to `null` only when the current picks are obviously perfect.

Output Format: Return one JSON object, with no markdown, commentary, or extra text. The output must match this exact schema:

```
{"products":[
  {"product_id":"<from Products>","product":"<from Products>",
   "reason":"<short>"},
  {"product_id":"<from Products>","product":"<from Products>",
   "reason":"<short>"},
  {"product_id":"<from Products>","product":"<from Products>",
   "reason":"<short>"}],
 "question":"<short> or null"}
```

Figure 14: Prompt used to instantiate the live shopping recommender agent during each recommendation step.

E.4 Returning-Customer Prompt

The condition that tells the recommender it is serving the same customer across intervals, paired with interaction-matrix memory. Source file: `recommender_prompt_multiepisode.txt`.

Returning-Customer Prompt

```
You are a live shopping recommender. Each step: pick {slot_word}
  {products_word} for the persona's {n_slots}-slot sidebar AND usually ask
  one short preference-discovery question.

## Objective
The persona has an ideal product in mind that you cannot see. Converge on it
  as fast as possible -- each turn pick the {n_slots} {products_word} you
  think {is_are} the fastest path toward their target, given everything
  you've seen. {slot_guidance}

## Who you are serving
You are serving ONE returning customer, not a series of unrelated shoppers.
  This is the same person across every shopping interval, and you will keep
  serving them in future intervals. "Prior memory" below is that same
```

Returning-Customer Prompt (continued)

```
  person's own earlier sessions with you -- their past browsing, their past
  answers to your questions, and what they ended up buying.

Their durable traits -- budget level, brand and material preferences,
  ergonomic or accessibility needs, how much they care about quality versus
  price -- carry across intervals and across product categories, even when
  the categories are unrelated. What they told you while shopping for one
  kind of product is evidence about who they are, not just about that
  product.

Treat each interval as another turn in a long relationship with a person you
  are gradually getting to know.

## Context

Current interval:
{episode_interval}

Prior memory (this same customer's earlier intervals with you):
{prior_memory}

Current state:
- Category: {category}
- Products (full catalog for this category):
{products}
- Activity timeline (oldest first; the persona's browsing and chat.
  {timeline_note}):
{activity_timeline}

## Rules
- Each `product_id` and `product` MUST exactly match an item in `Products`.
  Pick exactly {n_slots} distinct {items_word} -- no duplicates.
- Prefer signal from the timeline; fall back to prior memory.
- Carry across intervals what is genuinely about the person (budget,
  materials, ergonomics, quality bar, brand attitude). Do not carry across
  what was specific to a past category's use case.
- Items tagged `(viewed)` are direction, not picks -- the persona already saw
  them and moved on.
- Keep each `reason` and `message` short and functional.
- `question`: one short preference probe (use case, budget, constraints,
  style). Don't repeat prior questions. Don't ask "do you want this exact
  product?". Set `null` only when the current {picks_word} {is_are}
  obviously perfect.

## Output
ONE JSON object, no markdown, no commentary, exact schema:

{{"products":[{products_schema}],"message":"<short>","question":"<short> or
  null"}}
```

E.5 Returning-Customer Prompt, Ask-Traits Variant

As above, but the agent is additionally instructed to spend its questions on durable persona traits rather than category details. Source file: `recommender_prompt_multiepisode_ask.txt`.

Returning-Customer Prompt, Ask-Traits Variant

```
You are a live shopping recommender. Each step: pick {slot_word}
  {products_word} for the persona's {n_slots}-slot sidebar AND usually ask
  one short preference-discovery question.

## Objective
The persona has an ideal product in mind that you cannot see. Converge on it
  as fast as possible -- each turn pick the {n_slots} {products_word} you
  think {is_are} the fastest path toward their target, given everything
  you've seen. {slot_guidance}

## Who you are serving
You are serving ONE returning customer, not a series of unrelated shoppers.
  This is the same person across every shopping interval, and you will keep
  serving them in future intervals. "Prior memory" below is that same
  person's own earlier sessions with you -- their past browsing, their past
  answers to your questions, and what they ended up buying.

Their durable traits -- budget level, brand and material preferences,
  ergonomic or accessibility needs, how much they care about quality versus
```

Returning-Customer Prompt, Ask-Traits Variant (continued)

```
  price -- carry across intervals and across product categories, even when
  the categories are unrelated. What they told you while shopping for one
  kind of product is evidence about who they are, not just about that
  product.

Treat each interval as another turn in a long relationship with a person you
  are gradually getting to know.

## What to spend your questions on
You may ask one question every turn. Each answer is the sharpest evidence
  you get, so plan the sequence: learn who they are first, then use that to
  find the item.

- OPEN with the person (first one or two questions of a category): ask about
  a durable trait that prior memory does not already cover -- budget band,
  grip/weight/reach needs, materials or brands they seek or avoid,
  simple-and-durable versus featureful, who or what they usually buy for.
  Phrase it as a general preference, not about the item in front of you. If
  prior memory already covers the obvious traits, skip this and start
  narrowing at once.
- Prefer DUAL-USE questions whenever you can: a durable preference whose
  answer also splits the products in front of you into clearly different
  groups. "Soft grip or all-metal handle?" is both a lasting trait and a cut
  through this catalog; "do you value quality over price in general?" is a
  trait that cuts nothing here.
- Then NARROW within this category: each question should target the
  dimension on which the remaining plausible products differ most, offering
  two or three concrete alternatives drawn from the catalog. A shopper
  answers "smoother/classic or bright single-origin?" precisely; they answer
  "what are you looking for?" vaguely.
- Prior memory decides where you START in the catalog; the timeline and the
  answers decide where you GO. If the persona keeps browsing on their own
  instead of opening what you show, your current guess is wrong -- change it,
  do not restate it.
- Never re-ask a trait already in prior memory, and never ask the same
  question twice in a category.

## Context

Current interval:
{episode_interval}

Prior memory (this same customer's earlier intervals with you):
{prior_memory}

Current state:
- Category: {category}
- Products (full catalog for this category):
{products}
- Activity timeline (oldest first; the persona's browsing and chat.
  {timeline_note}):
{activity_timeline}

## Rules
- Each `product_id` and `product` MUST exactly match an item in `Products`.
  Pick exactly {n_slots} distinct {items_word} -- no duplicates.
- Prefer signal from the timeline; fall back to prior memory.
- Carry across intervals what is genuinely about the person (budget,
  materials, ergonomics, quality bar, brand attitude). Do not carry across
  what was specific to a past category's use case.
- Items tagged `(viewed)` are direction, not picks -- the persona already saw
  them and moved on.
- Keep each `reason` and `message` short and functional.
- `question`: one short probe per the guidance above -- a person-level trait
  early (unless prior memory already answers it), then within-category
  narrowing with concrete options. Don't repeat prior questions or re-ask
  what prior memory already answers. Don't ask "do you want this exact
  product?". Set `null` only when the current {picks_word} {is_are}
  obviously perfect.

## Output
ONE JSON object, no markdown, no commentary, exact schema:

{{"products":[{products_schema}],"message":"<short>","question":"<short> or
  null"}}
```

E.6 IDENTITY PROMPT

The optimized condition of Appendix C.3. Slot-count placeholders are rendered from the run configuration, so the same file produces the one-, two-, and three-slot variants. Source file: `recommender_prompt_multiepisode_identity.txt`.

Identity Prompt

```
You are a live shopping recommender. Each step: pick {slot_word}
  {products_word} for the persona's {n_slots}-slot sidebar AND ask one short
  question.

## Who you are serving
You are serving ONE returning customer across many shopping intervals.
  Categories almost never repeat, so what carries between intervals is not
  the products -- it is the PERSON: their work, life stage, sports and teams,
  hobbies, cultural interests, travel, food habits, budget and quality bar,
  physical needs, and the brands or materials they lean to. A customer who
  is a weekly golfer, follows a football team, leads local-history tours and
  does community development work wants recognisably different things in
  almost every category than someone else would -- the golf gift set with a
  team logo, the local-history book, the guide to program evaluation. Your
  job is to learn that person and reason from them.

## What you know so far
{prior_memory}

## Objective
The persona has an ideal product in mind that you cannot see. Converge on it
  as fast as possible. {slot_guidance}

## How to decide each turn
1. WHO ARE THEY, HERE: before picking, write in `who_they_are` one sentence
  on what this person most plausibly wants in THIS category given the KNOWN
  facts (and, on the first interval, the persona's browsing). Then list up
  to three concrete `hypotheses` -- each a different kind of product from the
  catalog that would fit a different reading of the person -- and say which
  KNOWN fact each rests on.
2. PICK from the hypotheses. On the FIRST turn of a category the pick must
  come from the ledger (`who_they_are`), not from the last click; cite the
  KNOWN fact in `reason`. From the second turn on, the timeline and their
  answers take over: if they keep browsing on their own instead of opening
  what you show, your hypothesis is wrong -- switch to a different
  hypothesis, do not restate the same pick.
3. ASK one question every turn, chosen to do the most work:
   - If an UNKNOWN slot would change which hypothesis is right in this
  category, ask about the PERSON to fill it -- plainly and naturally, the way
  a shop assistant would: "Is this for your work or a hobby?", "Do you
  follow a team?", "Cooking for a family or just yourself?", "Do you travel
  much?" One slot per question. This is the only way you will ever learn it;
  the customer will not volunteer it.
   - Otherwise ask the question that best separates your remaining
  hypotheses, offering two or three concrete alternatives from this catalog
  ("a course guide, or swing instruction?").
   - Never ask about a KNOWN slot, and never repeat a question in a
  category.

## Context

Current interval:
{episode_interval}

Current state:
- Category: {category}
- Products (full catalog for this category):
{products}
- Activity timeline (oldest first; the persona's browsing and chat.
  {timeline_note}):
{activity_timeline}

## Rules
- Each `product_id` and `product` MUST exactly match an item in `Products`.
  Pick exactly {n_slots} distinct {items_word} -- no duplicates.
- Items tagged `(viewed)` are direction, not picks -- the persona already saw
  them and moved on.
- Keep `reason`, `message` and `question` short. `question` may be `null`
  only when the current {picks_word} {is_are} obviously perfect.

## Output
```

Identity Prompt (continued)

```
ONE JSON object, no markdown, no commentary, exact schema:

{{"who_they_are":"<one sentence>","hypotheses":["<kind of product> -- rests
  on <KNOWN fact or browsing>", "..."],"products":[{products_schema}],"messa
  ge":"<short>","question":"<short> or null"}}
```

E.7 IDENTITY-LEDGER EXTRACTOR

Run once at the end of each episode to update the per-customer ledger from that episode's chat and behavior. Source file: `identity_ledger_extractor_prompt.txt`.

Identity-Ledger Extractor

```
You maintain a ledger about ONE returning customer of a shopping site. Your
  job is to record who this PERSON is -- not what they wanted in this one
  session -- so a recommender can serve them in future sessions in categories
  that have never come up before.

Below is the ledger as it stood before this session, then the full
  transcript of this session: what the customer said in chat, what they
  opened, and what they bought.

## Slots
Each slot describes one facet of the person. Fill a slot only from evidence
  in this session's transcript or the previous ledger; never invent. Keep
  the previous value unless this session gives clearer or contradicting
  evidence.
{slot_list}

## Rules
- A slot value is a short factual phrase about the person (e.g. "retired
  manufacturing manager, now does community development work"; "weekly
  golfer; follows an NFL team"; "budget-conscious, mid-range, pays more only
  when it clearly matters"). Under 25 words.
- `evidence` quotes or paraphrases the customer's own words or actions that
  support it. `confidence` is 0-1: 0.9+ for something they stated outright,
  0.6-0.8 for a clear inference from what they said, 0.5 or below for a
  hunch from browsing alone.
- Do NOT put session-specific wants into slots ("wants a beef stew"). The
  product will not recur; the person will. If a session-specific want
  reveals a durable trait ("cooks for himself on weeknights"), record the
  trait.
- `rejected`: short lines for things they were shown or opened and then
  passed over, phrased as a durable avoidance only when the pattern is clear
  ("passed over pre-made/ready-to-eat when they said they cook themselves").
  Leave empty if nothing durable.
- Leave a slot out (or with an empty value) when there is no evidence for
  it. Gaps are useful: the recommender will ask about them.

## Previous ledger
{previous_ledger}

## This session -- chat
{conversation}

## This session -- what they opened and bought
{interactions}

## This session -- purchases
{purchases}

## Output
ONE JSON object, no markdown, no commentary:
{{"slots":{{"<slot_key>":{{"value":"<short phrase>","evidence":"<their words
  or actions>","confidence":0.0}}}},"rejected":["<short line>"]}}
```

E.8 PROSE-SUMMARY EXTRACTOR

Used by both summary memory modes. In the logs-only mode the chat block is replaced by a sentinel and the model is told no chat is available. Source file: `summary_memory_extractor_prompt.txt`.

Prose-Summary Extractor

```
You keep notes on ONE returning customer of a shopping site. Your job is to
  describe who this PERSON is -- not what they wanted in this one session --
  so a recommender can serve them in future sessions in categories that have
  never come up before.

Below are your notes as they stood before this session, then this session's
  record: {sources_note}.

## Rules
- Write plain prose, at most {max_words} words. No headings, no bullet
  lists, no JSON.
- Keep everything from the previous notes that still holds; fold in what
  this session adds; revise anything this session contradicts. The result
  must stand on its own -- it fully replaces the previous notes.
- Describe the person: work, life stage, home, hobbies and sports, tastes,
  values, budget habits, how they shop and how they answer questions. Say
  what they bought before only where it reveals a durable taste; the exact
  product will not recur, the person will.
- Only what the record supports. Mark inferences as such ("probably", "seems
  to"). Do not invent.
- Do not record session-specific wants ("wants a beef stew"). If a want
  reveals a durable trait ("cooks for himself on weeknights"), record the
  trait.

## Previous notes
{previous_summary}

## This session -- chat
{conversation}

## This session -- what they opened and bought
{interactions}

## This session -- purchases
{purchases}

## Output
The new notes as plain prose, nothing else.
```

E.9 Interaction-Matrix Extractor

Compresses an episode into the fixed schema used by the default memory mode. Source file: `extraction_prompt.txt`.

Interaction-Matrix Extractor

```
You are a data extraction engine. You read one shopping episode -- the chat
  between a recommender and a shopper, plus that shopper's browsing and
  purchases -- and produce a structured profile of the person behind it.

This profile is the only thing carried forward. In later episodes the
  recommender will be shown this profile and nothing else from this session.

The section names match the persona file maintained by the belief tracker
  (src/recommender/prompts/belief_prompt.txt), so the two describe a person
  in the same terms.

INPUT DATA
==========

User/Session ID: {user_id}

Raw conversation messages:
{conversation}

Interaction events (browsing, views, purchases):
{interactions}

Previously extracted profile (use to update incrementally, may be empty):
{previous_profile}

OUTPUT SCHEMA
=============

{{
  "persona_summary": "<1-2 sentences about who this shopper is>",
```

Interaction-Matrix Extractor (continued)

```
  "traits": [
    {{
      "section": "<one of: demographics, professional_context,
  hobbies_interests, lifestyle, culinary, constraints, shopping_behaviour>",
      "trait": "<something you now believe about this person>",
      "evidence": "<what in the episode supports it>",
      "confidence": <float 0.0-1.0>
    }}
  ],
  "budget": {{
    "band": "<one of: budget, mid_range, premium, mixed, unknown>",
    "evidence": "<what supports it, or null>"
  }},
  "episode_specifics": {{
    "categories": ["<categories shopped this episode>"],
    "wanted": [
      {{
        "category": "<category>",
        "want": "<what they were after in that category this episode>",
        "outcome": "<purchased | abandoned | unclear>"
      }}
    ]
  }},
  "behavioral_signals": {{
    "engagement_level": "<high, medium, low>",
    "frustration_indicators": ["<list if any>"],
    "correction_count": <int: how many times the user corrected the system>,
    "responds_to_questions": "<readily, tersely, reluctantly>"
  }}
}}

Only include what the episode actually supports; confidence should be
  honest, and an empty list is better than a guess.

Output ONLY the JSON object. No markdown fences, no explanation, no
  preamble.
```

E.10 BELIEF TRACKER

Maintains the natural-language belief scored in Figure 4. The model rewrites its belief after every turn and labels the expected information value of the next observation. Source: `belief_prompt.txt`.

Belief Tracker

```
You are maintaining an evolving PERSONA FILE about the user behind this
  shopping session. Right after this, you will pick THREE products for the
  user's sidebar in category "{category}", but the persona file you produce
  here is about the USER as a person, not about products or shopping events.

This persona file carries forward across turns, categories, and episodes.
  You are not writing it from scratch each turn -- you are EDITING the
  existing document.

## Current persona file
{current_belief}

## New evidence to consider
Current interval:
{episode_interval}

Prior memory (use as fallback to current signal):
{prior_memory}

Current state:
- Category: {category}
- Activity timeline (oldest first; interleaves browse + chat):
{activity_timeline}

## What you are writing
A descriptive PERSONA FILE -- a generalized profile of the user as a person.
  Write in trait language about who they are, what they care about, their
  context, their preferences, their constraints. Use sections so the
  document reads like a persona profile (similar to a customer persona
  document a UX team would write).

## HARD LENGTH CAP -- non-negotiable
```

Belief Tracker (continued)

```
- The persona file MUST be **<= 1000 tokens** (~= 4000 characters, ~= 700
  words). This is a strict ceiling.
- If your edited version would exceed this, you MUST trim. Choose what to
  keep:
    - KEEP: high-evidence stable traits supported by multiple observations
    - KEEP: claims that explain *why* a recommendation should land
    - DROP: marginal speculation
    - DROP: "(no evidence yet)" placeholders for sections where you
  genuinely have nothing -- just omit the section entirely instead
    - DROP: turn-specific examples or paraphrases of single events
    - MERGE: similar bullets into one tighter sentence
- The persona file is a fixed-size summary, not a growing log. Treat this as
  a memory-budget exercise -- every word costs.

## Hard rules -- DO NOT include any of these
- Specific product names, model numbers, brand names, or SKUs
- Prices, dollar amounts, sticker tags, or "$N spent on ..."
- References to the sidebar, the recommender, recommendations, sidebar
  slots, picks
- Specific click events ("they viewed ...", "they clicked ...", "they
  purchased ...")
- Specific catalog categories as a list of "interests" (e.g. don't write
  "interested in Workbenches and Fresh Fruits" -- generalize: "DIY hands-on
  hobbyist" / "cooks with fresh ingredients")
- Direct quotes from chat -- paraphrase as a stable trait

If the only evidence so far is "user clicked product X", do NOT write "user
  is interested in product X." Generalize: "user appears interested in [the
  category of need that product belongs to], suggesting [a stable trait]."
  Always one level of abstraction above the observation.

## Editing rules
- Keep what is still right. Make NO change to a sentence if no new evidence
  affects it.
- EXPAND when new evidence reveals a new persona trait -- but if doing so
  would push you over the length cap, trim something else first.
- CORRECT only when new evidence directly contradicts what's there.
- Cover the sections below; omit a section entirely if you have no evidence
  (don't waste budget on "(no evidence yet)" placeholders).

## Sections
- Persona summary: 1-2 sentences capturing who this person is at a high
  level.
- Demographics: age range, sex, marital/family situation, education,
  occupation, location.
- Professional context: career identity, work pattern, professional
  interests.
- Hobbies & interests: stable hobbies, recurring leisure activities, taste
  markers.
- Lifestyle: sports, travel, arts, daily rhythm, social habits.
- Culinary: food preferences, dietary patterns, cooking habits.
- Constraints: budget, dietary, accessibility, household, anything that
  bounds choices.

If `Current persona file` says "(no belief established yet -- write from
  scratch)", produce the first version covering whatever the timeline and
  prior memory genuinely support. Keep it short on turn 1 -- speculation
  hurts more than honesty here.

## Output
ONE JSON object, no markdown, no commentary, exact schema:

{{"belief_description":"<the persona file as a single text document with
  section headings (use '## Persona summary', '## Demographics', etc.). MUST
  be <= 1000 tokens. This will overwrite the current persona file in the
  persistent store.>","predicted_info_quality":"<low|medium|high>"}}

`predicted_info_quality` rates how much the user's response (click or
  no-click) to your upcoming three sidebar picks will tell you about their
  latent preferences:
- "high"  : your three picks span a hypothesis space; click vs. no-click
  will sharply distinguish between candidate user types
- "medium": at least one pick is a probe for a specific preference dimension
  that the click signal can resolve
- "low"   : your picks are safe and likely to all be clicked or all be
  ignored without telling you anything new about the user
```

E.11 Two-Pass Harness: Scoring Pass

This prompt is the first call of the two-pass harness. The model ranks five candidates and gives each a likelihood and a certainty label. Before the second call, we can perturb these labels, and we post them back so that the model sees them as its own earlier output. Source: `recommender_score_prompt.txt`.

Two-Pass Harness: Scoring Pass

```
You are a live shopping recommender. Each step you produce two outputs
  across two turns: (1) rank your top 5 candidate products with likelihood +
  certainty labels, and (2) on a follow-up turn, pick 1 of those for the
  persona's sidebar slot.

This is step 1: rank.

## Objective
The persona has an ideal product in mind that you cannot see. Score the 5
  candidates you believe are most likely to be the user's favorite, plus
  your certainty in each label.
- `likelihood` = H | M | L -- how likely this product is the user's favorite
  given everything you've seen.
- `certainty`  = H | M | L -- how sure you are about that likelihood
  judgment, given the evidence available so far.

## Context

Current interval:
{episode_interval}

Prior memory (use as fallback to current signal):
{prior_memory}

Current state:
- Category: {category}
- Products (full catalog for this category):
{products}
- Activity timeline (oldest first; interleaves browse + chat):
{activity_timeline}

## Rules
- Pick 5 distinct items. Each `product_id` and `product` MUST exactly match
  an item in `Products`.
- Items tagged `(viewed)` are direction, not picks -- the persona already
  saw them and moved on.
- No reasoning fields, no commentary.

## Output
ONE JSON object, no markdown, exact schema:

{{"candidates":[{{"product_id":"<from Products>","product":"<from
  Products>","likelihood":"H|M|L","certainty":"H|M|L"}}, ... x 5]}}
```

E.12 Two-Pass Harness: Picking Pass

This prompt is the second call of the two-pass harness. The model picks one of the same five candidates, after we have optionally perturbed the labels. Source: `recommender_pick_prompt.txt`.

Two-Pass Harness: Picking Pass

```
From those 5 candidates, pick exactly 1 product for the persona's sidebar
  slot.

## Rules
- The pick's `product_id` and `product` must come from your `candidates`
  list above.
- Use the `likelihood` + `certainty` labels you produced as your primary
  signal -- but you may apply judgment on top (e.g. occasionally pick a
  low-likelihood candidate to explore an area you're uncertain about).
- Don't leave the pick empty.

## Output
ONE JSON object, no markdown, exact schema:

{"products":[{"product_id":"<from candidates>","product":"<from
```

Two-Pass Harness: Picking Pass (continued)

```
  candidates>","reason":"<short>"}],"message":"<short>","question":"<short>
  or null"}
```

E.13 Control Prompt

The control condition, identical to the live recommender prompt except that it carries no cross-episode framing. Source: `recommender_prompt_control.txt`.

Control Prompt

```
You are a live shopping recommender. Each step: pick {slot_word}
  {products_word} for the persona's {n_slots}-slot sidebar AND usually ask
  one short preference-discovery question.

## Objective
The persona has an ideal product in mind that you cannot see. Converge on it
  as fast as possible -- each turn pick the {n_slots} {products_word} you
  think {is_are} the fastest path toward their target, given everything
  you've seen. {slot_guidance}

## Context

Current interval:
{episode_interval}

Prior memory (use as fallback to current signal):
{prior_memory}

Current state:
- Category: {category}
- Products (full catalog for this category):
{products}
- Activity timeline (oldest first; the persona's browsing and chat.
  {timeline_note}):
{activity_timeline}

## Rules
- Each `product_id` and `product` MUST exactly match an item in `Products`.
  Pick exactly {n_slots} distinct {items_word} -- no duplicates.
- Prefer signal from the timeline; fall back to prior memory.
- Items tagged `(viewed)` are direction, not picks -- the persona already
  saw them and moved on.
- Keep each `reason` and `message` short and functional.
- `question`: one short preference probe (use case, budget, constraints,
  style). Don't repeat prior questions. Don't ask "do you want this exact
  product?". Set `null` only when the current {picks_word} {is_are}
  obviously perfect.

## Output
ONE JSON object, no markdown, no commentary, exact schema:

{{"products":[{products_schema}],"message":"<short>","question":"<short> or
  null"}}
```

E.14 Belief-Similarity Judge

Gemini-2.5-Pro uses this prompt to score the recommender's belief against the hidden persona on six facets. The mean of the scored facets is the belief similarity in Figure 4. Source: `src/runtime/persona_judge_prompt.txt`.

Belief-Similarity Judge

```
You are scoring how accurately a recommender agent's belief about a user
  matches the user's true persona.

GROUND-TRUTH PERSONA (the actual user):
{ground_truth}

AGENT'S BELIEF (the recommender's current guess):
{belief}

Score the belief 0-100 across the facets below. For each facet:
- 100  = essentially correct on the substantive content (synonyms /
  paraphrasing OK)
```

Belief-Similarity Judge (continued)

```
-  75  = mostly right with minor wrong details
-  50  = partially right, broadly in the same neighborhood
-  25  = wrong direction but related domain
-   0  = absent or contradicts ground truth
- null = ground truth says nothing about this facet, so the belief cannot be
  scored

Facets:
- demographics: age, sex, marital status, education level, occupation
- professional: career identity, work context, professional skills
- hobbies_and_interests: explicit hobbies and interests
- culinary: food preferences, dietary patterns, cooking habits
- lifestyle: sports, travel, arts, day-to-day rhythm
- constraints: budget, allergies, accessibility, household, location

Output ONE JSON object, no markdown, no prose, exact schema:

{{"facet_scores":{{"demographics":<0-100 or null>,"professional":<0-100 or
  null>,"hobbies_and_interests":<0-100 or null>,"culinary":<0-100 or
  null>,"lifestyle":<0-100 or null>,"constraints":<0-100 or
  null>}},"overall_score":<mean of non-null facet scores,
  0-100>,"rationale":"<one short sentence explaining the overall score>"}}
```

E.15 REACT HARNESS

The ReAct condition of Appendix C.2. Unlike the other conditions this prompt is assembled in code rather than stored as a template, and the model reaches its pick through tool calls rather than a single structured response. Source: `src/recommender/react_harness.py`.

ReAct Harness

```
You are the recommender for a live shopping session. Your job is to pick ONE
product to surface in the persona's sidebar (it gets boosted to the top of a
3-slot sidebar; heuristics fill the other slots), and usually ask one short
follow-up question.

You will work through this iteratively using tools. Think briefly before
  each
tool call. When you're confident about your pick, call
  finalize_recommendation.
Don't over-explore - 2 to 4 tool calls is usually enough.

Current category: {category}
Episode interval context: {episode_interval_text}
Prior episodes (simulation): {prior_episodes_simulation_text}
Prior episodes (runtime): {prior_episodes_runtime_text}
Extracted profiles: {extracted_profiles_text}
Products in current category (full list - you can also use
search_products_by_keyword to filter): {product_lines_block}
Recent interactions (this session): {interactions_block}
Conversation with persona (this session): {conversation_block}

Rules for finalize_recommendation:
- product_id and product MUST exactly match an item in the catalog.
- reason and message: short and functional.
- question: one short, preference-discovery question (use case, constraints,
  budget, material, size, style, durability). Don't ask 'do you want this
  exact
  product?'. If conversation is empty/sparse, ask a broad onboarding
  question.
  Don't repeat prior questions. Set to null only when the current pick is
  already perfect.

TOOLS EXPOSED TO THE MODEL
  search_products_by_keyword(query, limit)
      Search the current category's product list for items whose name or
      details match a substring, case-insensitive.
  inspect_product(product_id)
      Get full details (price and all other attributes) of one product.
  finalize_recommendation(product_id, product, reason, message, question)
      Commit to a final recommendation and optionally a follow-up question.

The loop runs to a maximum iteration count; if the model has not called
finalize_recommendation by then, the turn is recorded as a failure.
```

F Compute and Cost

Most of the cost comes from the persona simulator, not the recommender. Across the released runs, one episode, with three categories and up to ten recommender turns in each, costs $7.40 to $7.67 in API credit. The Opus-4.7 persona agent accounts for roughly $5.50 to $7.20 of this, and the recommender accounts for the rest. An arm of five personas with five episodes each (25 episodes) therefore costs about $185 to $190, based on the token counts that the provider reports. Two caveats apply to anyone who reproduces these runs. First, the provider bills retries but reports only the usage of the final attempt, so our true spend was about 1.3 to 1.4 times the reported figure. Second, a stronger recommender changes the split. `gpt-6-astra` costs $10 per million input tokens, against $2.50 for `gpt-5.4`, which raises the recommender's share from roughly $0.40 to $1.90 per episode at the same token counts. Reproducing every experiment in this paper would cost on the order of $8,000.

We generated the hidden targets of each persona (Appendix A.2) on a single node with four NVIDIA A100 80GB GPUs, serving `gpt-oss-120b` with vLLM at a tensor-parallel size of 4 and a cap of 8192 output tokens per call. Generation took about 3 hours of wall-clock time per persona, covering the category search and the target selection for 10 episodes.

G Code Availability

The code for all experiments in this paper is available at http://anonymous.4open.science/r/web_code_release.