Statistical Priors for Implicit Preferences: Decoupling Skill Selection as a Local Harness in Personal Agents

Anonymous ACL submission

Abstract

As Large Language Model (LLM) capabilities advance, locally deployed personal agents relying on API-based remote models and external skills have emerged as a novel paradigm. With the rapid expansion of available skills, enabling personal agents to learn and adapt to implicit user preferences becomes a critical challenge. However, local deployment constraints preclude complex centralized selection algorithms, creating an urgent need for a lightweight local preference harness. This paper explores the implementation of such a harness through a novel architecture that strictly decouples statistical preference learning from semantic intent parsing. Specifically, we leverage localized statistical results to influence and modulate the selection decisions of the remote LLM. Extensive evaluations demonstrate that our decoupled approach achieves the lowest cumulative regret and highest test accuracy, significantly outperforming traditional memory-augmented agents. We open-source our code via an anonymous GitHub repository at https://anonymous.4open.science/r/Personalized-Skill-Selection-BC62.

1 Introduction

With the continuous advancement of Large Language Models (LLMs), the research paradigm has increasingly shifted from merely scaling model capabilities to transforming these powerful models into robust productivity tools. Against this backdrop, LLM-based agents have emerged as a prominent research frontier, giving rise to numerous commercial products capable of complex reasoning and task execution, such as Gemini's deep research (Google, 2025), Manus (Monica, 2025), and Perplexity (Perplexity AI, 2025). Simultaneously, the growing demand for highly customizable, privacy-preserving, and deeply integrated digital assistants has catalyzed the rapid development of locally deployed personal agents (e.g., Claude Code (Anthropic, 2025), Codex (OpenAI, 2025a), OpenClaw (Peter Steinberger, 2026), HERMES Agent (Nous Research, 2026), and Pi Agent (Mario Zechner, 2026)). These personal agents represent a novel interaction paradigm, intimately embedding themselves into users' daily workflows to provide highly individualized assistance.

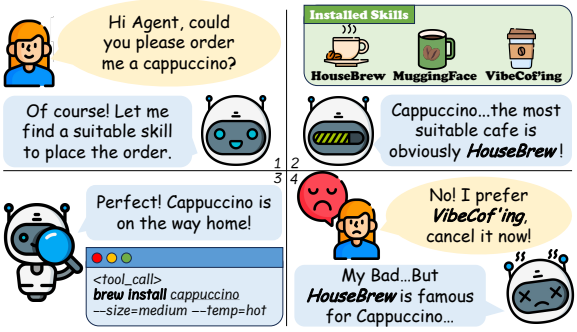


Figure 1: **A typical failure when personal agents are handling tasks with implicit user preference.** The agent selected a skill based on the user's instruction and its own knowledge, but ignored the user's implicit preferences, resulting in an unsatisfactory choice.

Distinct from traditional frameworks, personal agents heavily rely on API-based remote foundation models for complex semantic reasoning while operating within locally deployed execution environments. To extend their functional boundaries, a rapidly proliferating ecosystem of external skills and tools is being integrated into these local setups (Qin et al., 2023; Schick et al., 2023; Patil et al., 2024). As the number of available skills expands, dynamically selecting the most appropriate skill tailored to individual user preferences has emerged as a critical challenge (Lin et al., 2025). As illustrated in Figure 1, when a user asks the agent to order a cappuccino, multiple functionally viable skills exist. The optimal choice is not determined by the explicit semantics of the user's query, but rather by their implicit, personal preference. Failing to accurately capture this preference

inevitably leads to frustrating interactions.

In real-world scenarios, user preferences for specific skills are rarely stated explicitly in every prompt; instead, they manifest through stochastic, high-frequency daily feedback and repetitive habits. For instance, a developer might consistently favor a specific linting skill for code inspection, or a user might habitually rely on a particular weather API. Consequently, the skill selection process in personal agents inherently encompasses two fundamentally different cognitive dimensions: **semantic understanding**, which parses explicit, ad-hoc intents from natural language instructions, and **statistical preference learning**, which identifies latent habits from historical interactions.

However, effectively managing statistical preference learning poses a unique challenge for personal agents. Due to their localized deployment and strict privacy constraints, adopting computationally heavy or complex centralized recommendation algorithms is highly impractical, creating an urgent need for lightweight, on-device preference management. Currently, the prevailing solution relies on prompt-injected memory structures, forcing a single remote LLM to simultaneously handle both historical frequency tracking and semantic reasoning. This conflation leads to severe systemic failures. Beyond introducing high API latency and context window overflow, LLMs frequently get the logic lost in multi-turn conversations (Laban et al., 2026). Consequently, memory-based approaches fail to robustly capture fine-grained statistical priors and severely lack mathematical interpretability.

To resolve this architectural bottleneck, we propose the LOCAL HARNESS, a novel framework that enforces a strict physical and logical decoupling between statistical preference learning and semantic intent parsing. Specifically, we delegate the probabilistic credit-assignment problem of modeling implicit user habits to a locally deployed, highly efficient statistical primitive. This local module natively manages the exploration-exploitation tradeoff and serves as the primary decision-maker. Simultaneously, the high-latency remote LLM is entirely removed from the high-frequency execution critical path, strictly reserved as a semantic exception handler to process explicit lexical overrides. The main contributions of this paper are threefold: (1) We identify the fundamental architectural flaw of conflating statistical preference learning with semantic reasoning in modern memory-augmented personal agents. (2) We propose the LOCAL HARNESS, a lightweight, decoupled architecture that synergizes a local statistical estimator with a remote LLM exception handler to accurately model user preferences. (3) Since preference-driven skill selection is a newly identified problem with no existing simulation environment, we construct TOOLBENCH-60, a dedicated sandbox, and conduct extensive empirical evaluations across diverse foundation models on it, demonstrating that our decoupled approach uniquely achieves both the lowest cumulative regret and highest test accuracy.

The remainder of this paper is organized as follows. Section 2 reviews related work on LLM agents and memory mechanisms. Section 3 formalizes the preference-driven skill selection problem and details the LOCAL HARNESS architecture. Section 4 presents theoretical analyses. Section 5 discusses our extensive experimental setup and results. Finally, Section 6 concludes the paper.

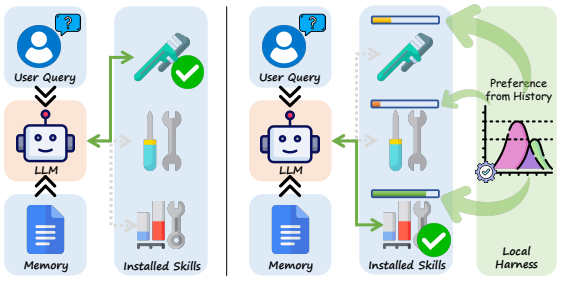
2 Related Work

2.1 LLM Agents and Tool Use

Empowering language models with external action execution has emerged as a cornerstone for building autonomous agentic systems (Qin et al., 2023; Yao et al., 2022; Zhou et al., 2026; Gan et al., 2026). Early methodologies established structured tool utilization by mapping textual intents to discrete API parameters (Schick et al., 2023; Patil et al., 2024). To scale these capabilities, subsequent frameworks optimized execution paths across dense multi-tool environments and complex multi-turn planning loops (Lin et al., 2025). Wang et al. (2026) further achieves interactive policy alignment by optimizing tool invocation through direct conversational feedback. While these tool-utilization paradigms excel at resolving explicit semantic tasks, they fundamentally formulate selection as an intent parsing problem, limiting the capability of capturing long-term user routines.

2.2 Harness Engineering

The concept of designing specialized harnesses or orchestrators around LLMs has gained traction to mitigate the unreliability and high latency of raw model inference (Lopopolo, 2026). A group of researches emphasize formalizing the programmatic architecture surrounding LLMs into systematic harness engineering rather than relying on ad-hoc scaffolding (Li et al., 2026; Zhou et al., 2026). Subsequent frameworks have advanced these archi-



(a) Skill-Selection of Personal Agents (b) Skill-Selection of Personal Agents by User Preference

Figure 2: **Overview of the personalized skill selection problem and proposed architecture.** (a) A classic skill selection process based on user query. (b) Our LOCAL HARNESS architecture improves this by using a local statistical estimator for preference learning.

tectures by introducing structured execution runtimes (Huang et al., 2026), flexible natural language interfaces (Pan et al., 2026a), and automated evolution mechanisms (Lin et al., 2026). More recently, the community has also begun exploring end-to-end optimization of model harnesses (Lee et al., 2026). While these orchestration frameworks excel at structuring explicit logic and execution infrastructure, their heavy engineering footprints make them impractical for lightweight, locally deployed personal agents.

2.3 Memory-Augmented LLMs

Recent advancements emphasize treating memory as a first-class cognitive primitive rather than a simple retrieval mechanism (Hu et al., 2026). Foundational frameworks such as Generative Agents (Park et al., 2023) pioneered experiential memory by simulating human-like episodic recording. Further refining methods construct dynamic memory for robust agent systems (Zhong et al., 2024; Shinn et al., 2024). Zhang et al. (2026) further achieves memory management in a generative manner. More recently, the community has also begun exploring self-evolving memory systems (Pan et al., 2026b). While these prompt-injected memory systems excel at retrieving explicit semantic facts, they frequently suffer from context overflow with dense logs, underscoring the necessity of our decoupled local harness to handle implicit statistical preferences.

3 Method

In this section, we formalize the personalized skill selection problem (Section 3.1) and present the LOCAL HARNESS architecture, detailing its statistical priors (Section 3.2) and end-to-end decision procedure (Section 3.3).

3.1 Task Formulation

Standard skill-selection workflows typically treat the decision process as a direct semantic mapping from a natural language query to a specific skill, as illustrated in Figure 2(a). In this conventional paradigm, the text of the query is assumed to contain sufficient surface-level intent to uniquely identify the target skill. However, in real-world personal agent interactions driven by stochastic and highly repetitive daily habits, user queries are frequently skill-agnostic, rendering purely text-based intent parsing fundamentally insufficient.

To address this, we formalize personalized skill selection as a sequential decision-making process under implicit user preference. At each interaction round t, a user u issues a query q_t. A shared domain classifier first predicts the target domain $\hat{d}(q_t) \in \mathcal{D}$, which restricts the available choices to a candidate skill inventory $\mathcal{T}_d$ containing K skills. Crucially, the ground-truth optimal skill $a_t^* \in \mathcal{T}_d$ is governed by the user's latent preference distribution $\pi_u : \mathcal{D} \to \Delta(\mathcal{T}_d)$. For a standard query, a_t^* is sampled directly from $\pi_u(\hat{d})$. Upon selecting skill a_t, the system receives a binary reward $r_t = \mathbb{1}\{a_t = a_t^*\}$ from user's subsequent feedback, which is considered an easily accessible reward signal (Wang et al., 2026). The primary objective is to maximize the cumulative expected reward $\sum_{t=1}^{T} r_t$ over a horizon of T rounds by implicitly learning π_u from historical feedback while maintaining robust adaptation to explicit semantic exceptions.

3.2 Decouple Skill Selection as Local Harness

To resolve the fundamental conflict of forcing a single remote LLM to simultaneously manage statistical preference optimization and semantic parsing, we introduce the LOCAL HARNESS architecture, illustrated in Figure 2(b). Concretely, a locally running, lightweight statistical component operates as the primary, default decision-maker. This component efficiently maintains the state space of the user's stochastic preferences locally and handles the exploration-exploitation trade-off mathematically, which we call a *statistical prior*. By delegating historical credit assignment to the local harness, and reserving the remote LLM for lexical overrides, the architecture achieves a robust balance between statistical personalization efficiency and zero-shot reasoning capabilities. We introduce the following two statistical priors implemented in this paper.

Frequency Prior. The simplest realization of the LOCAL HARNESS is a per-user empirical success-rate table. For every triple (u, d, k) comprising a user u, a domain d, and a candidate skill $k \in \mathcal{T}_d$, this component maintains two scalars after t rounds: the number of attempts $N_{u,d,k}$ and the number of successes $S_{u,d,k}$ (the count of rounds in which skill k was selected and received a positive reward). Its preference distribution over the inferred domain's candidate set $\mathcal{T}_{\hat{d}(q_t)}$ is the normalized success rate:

$$\hat{p}^{\text{freq}}_{u,d}(k) \;=\; \frac{S_{u,d,k}/\max(N_{u,d,k},\,1)}{\sum_{k' \in \mathcal{T}_d} S_{u,d,k'}/\max(N_{u,d,k'},\,1)}, \tag{1}$$

and its default action is the corresponding $\arg\max$. The frequency prior makes no use of the query text and performs no explicit exploration. It therefore characterizes the elementary local statistical primitive, and serves as a controlled point of comparison for richer harnesses.

Bandit Prior. A more principled realization casts each (u, d, k) as a separate contextual-bandit arm. We instantiate this with LINUCB (Li et al., 2010): every arm maintains a D-dimensional parameter $\boldsymbol{\theta}_{u,d,k}$ together with a regularized Gram matrix $\mathbf{A}_{u,d,k} \in \mathbb{R}^{D \times D}$ and a moment vector $\mathbf{b}_{u,d,k} \in \mathbb{R}^D$, updated online for each (u, d, k) as:

$$\begin{aligned}
&\mathbf{A} \leftarrow \mathbf{A} + \boldsymbol{\phi}(q,k,d)\boldsymbol{\phi}(q,k,d)^\top,\\
&\mathbf{b} \leftarrow \mathbf{b} + r\,\boldsymbol{\phi}(q,k,d),\\
&\boldsymbol{\theta} = \mathbf{A}^{-1}\mathbf{b}.
\end{aligned}$$

Here $r = \{0, 1\}$ represents the binary reward. The context vector $\boldsymbol{\phi}(q, k, d) \in \mathbb{R}^D$ is constructed by feature-hashing (Weinberger et al., 2009) the query, the skill name, and the inferred domain, producing a deterministic, vocabulary-free representation that generalizes across queries with related surface forms. At decision time the harness computes an upper-confidence-bound (UCB) score:

$$\begin{aligned}
s_{u,d,k}(q) \;&=\; \boldsymbol{\theta}^\top_{u,d,k}\boldsymbol{\phi}(q,k,d)\\
&+\; \alpha_{\text{ucb}}\sqrt{\boldsymbol{\phi}(q,k,d)^\top \mathbf{A}^{-1}_{u,d,k}\boldsymbol{\phi}(q,k,d)}.
\end{aligned} \tag{2}$$

The default action is the corresponding $\arg\max_k s_{u,d,k}(q)$. Additionally, the UCB scores can be exposed as a tempered-softmax preference distribution over $\mathcal{T}_{\hat{d}(q_t)}$:

$$\hat{p}^{\text{bandit}}_{u,d}(k) \;=\; \frac{\exp\big(\beta\, s_{u,d,k}(q)\big)}{\sum_{k' \in \mathcal{T}_{\hat{d}}} \exp\big(\beta\, s_{u,d,k'}(q)\big)}, \tag{3}$$

where β is the softmax temperature. The exploration coefficient α_{ucb} controls the balance between exploiting the current point estimate and probing under-sampled arms. Compared with the frequency prior, the bandit prior is sensitive to query context (through $\boldsymbol{\phi}$), and, more importantly, maintains *calibrated uncertainty*: the second term of $s_{u,d,k}(q)$ grows with epistemic uncertainty and shrinks as data accumulate. Both properties become consequential when the user's underlying preference distribution $\pi_u(d)$ is stochastic.

We emphasize that neither prior is itself a contribution: both are well-established estimators, and alternatives such as Thompson sampling or kernel bandits are equally compatible with the architecture. Our claim is that *any* consistent local harness, when paired with the override channel described next, suffices for personalized skill selection; comparing two instantiations characterizes how the strength of the statistical primitive interacts with the rest of the system.

3.3 Skill Selection via Statistical Priors

We now describe the end-to-end decision procedure that the LOCAL HARNESS architecture induces, given a chosen statistical prior $h \in \{\text{freq}, \text{bandit}\}$. The procedure separates each round of personalized skill selection into the following three steps.

Step 1: Shared domain classification. Upon receiving a query q_t, the system performs a single LLM call to infer the target domain $\hat{d}(q_t) \in \mathcal{D}$. The resulting label $\hat{d}(q_t)$ restricts the candidate set to $\mathcal{T}_{\hat{d}(q_t)} \subset \mathcal{T}$ for all downstream operations.

Step 2: Local statistical default. The local harness consumes the inferred domain together with the user's state and returns a candidate action:

$$\tilde{a}_t \;=\; \arg\max_{k \in \mathcal{T}_{\hat{d}(q_t)}} \hat{p}^{\,h}_{u,\hat{d}(q_t)}(k),$$

where $\hat{p}^{\,h}$ is either the success-rate distribution of Equation (1) or the bandit-derived distribution implied by Equation (2). This step is fully local, deterministic, and conditional on the user state.

Step 3: Semantic override probe. The remote LLM is engaged only to check whether the user's query contains an explicit lexical instruction that supersedes the user's habitual preference. We issue a single binary probe (details in Appendix I):

$$o_t,\, \tau_t \;=\; \text{LLM}_{\text{override}}\Big(q_t,\, \mathcal{T}_{\hat{d}(q_t)}\Big),$$

where $o_t \in \{0, 1\}$ indicates whether the query explicitly names a specific skill and $\tau_t \in \mathcal{T}_{\hat{d}(q_t)}$ is the named skill when $o_t = 1$.

Illustrative example. Consider the cappuccino scenario in Figure 1. The user, who habitually prefers VIBECOF'ING, issues the query "*Order me a cappuccino*". In Step 1, the shared domain classifier maps the query to the `eCommerce` domain, restricting the candidate set to the six coffee-ordering skills. In Step 2, the local harness consults its accumulated statistics for this user and returns $\tilde{a}_t =$ VIBECOF'ING as the default action (since the user has historically rewarded this skill most often in this domain). In Step 3, the override probe inspects the query text: it contains no explicit skill name, so $o_t = 0$. The final action $a_t = \tilde{a}_t =$ VIBECOF'ING is executed. Had the user instead said "*Order me a cappuccino from HouseBrew*", Step 3 would have returned $o_t = 1$, $\tau_t =$ HOUSEBREW, and the override would have superseded the habitual default.

Action and update. The action executed at round t is then

$$a_t \;=\; \begin{cases} \tau_t & \text{if } o_t = 1, \\ \tilde{a}_t & \text{otherwise.} \end{cases}$$

The environment returns the binary reward $r_t = \mathbb{1}\{a_t = a_t^*\}$, after which the local harness alone is updated with the tuple $(q_t, \hat{d}(q_t), a_t, r_t)$.

4 Theoretical Analysis

We now show that, under mild assumptions automatically satisfied by the priors of Section 3, the LOCAL HARNESS architecture provably brings the agent's expected per-round regret strictly closer to the user-optimal level.

Setup. Fix a user u. At round t, the shared domain classifier reduces the candidate set to $\mathcal{T}_{\hat{d}}$ of size K. Write $Q(\cdot \mid q) := \pi_u(\hat{d}(q)) \in \Delta(\mathcal{T}_{\hat{d}})$ for the user's latent preference distribution; the ground-truth optimal skill on a standard query satisfies $a^\star \sim Q(\cdot \mid q)$. Any agent induces a (stochastic) policy with conditional distribution $\tilde{P}(\cdot \mid q) \in \Delta(\mathcal{T}_{\hat{d}})$, and the binary loss $\ell(a, a^\star) = \mathbb{1}\{a \neq a^\star\}$ gives the expected per-round regret:

$$R(\tilde{P}) \;:=\; 1 - \mathbb{E}_q\big\langle \tilde{P}(\cdot \mid q),\, Q(\cdot \mid q)\big\rangle. \qquad (4)$$

We distinguish three distributions in $\Delta(\mathcal{T}_{\hat{d}})$: P denotes the LLM's selection distribution; h_t denotes the distribution induced by the local statistical prior $\hat{p}^h_{u,\hat{d}}$ of Section 3 after t rounds; and P'_t denotes the policy of LOCAL HARNESS. The override mechanism makes P'_t a convex combination:

$$P'_t \;=\; (1-\lambda)\, h_t \;+\; \lambda\, P, \qquad \lambda \in [0, 1], \qquad (5)$$

with $\lambda = \rho_e$ for BANDIT-AS-OVERRIDE and FREQ-AS-OVERRIDE, since standard queries (fraction $1 - \rho_e$) are routed through h_t and explicit queries (fraction ρ_e) are routed through P. Note that $R(\tilde{P})$ is linear in $\tilde{P}$. We then present the main result as follows.

Theorem 4.1 (Risk improvement under LOCAL HARNESS)**.** *Assume (i) the local statistical prior is consistent, i.e.* $\mathbb{E}[\mathrm{TV}(h_t, Q)] \to 0$ *as* $t \to \infty$*, and (ii) the personalization problem is non-trivial, i.e.* $|R(P) - R(Q)| > 0$*. Then there exists* $T_0 < \infty$ *such that for every* $t \geq T_0$*,*

$$\mathbb{E}\big|R(P'_t) - R(Q)\big| \;<\; \big|R(P) - R(Q)\big|. \qquad (6)$$

Discussion. Theorem 4.1 precisely characterizes the override architecture but is *not* available to memory-augmented agents that funnel both statistical and semantic signals through a single LLM call. A quantitative refinement,

$$\begin{aligned} \mathbb{E}\big|R(P'_t) - R(Q)\big| \;\leq\; & \lambda\, |R(P) - R(Q)| \\ & + 2(1-\lambda)\, C_{\text{prior}}\, t^{-1/2}, \end{aligned} \qquad (7)$$

exposes a two-phase convergence: an early phase dominated by the $O(t^{-1/2})$ exploration term, and a long-horizon plateau at $\lambda|R(P) - R(Q)|$, i.e. a strict $(1-\lambda)\times$ reduction over the LLM-only baseline. The full proof is deferred to Appendix B.

5 Experiment

We conduct extensive empirical evaluations to validate the effectiveness of the proposed LOCAL HARNESS architecture across diverse language model backbones and preference regimes.

5.1 Experimental Setup

This subsection outlines the experimental setup, detailing the sandbox construction, evaluation metrics, model configurations, and the nine evaluated agents grouped by four different design families. We kindly refer the readers to the detailed implementations in Appendix A.

5.1.1 Sandbox

Since preference-driven skill selection under implicit user preference is a newly identified problem, no existing benchmark provides synthetic users with controllable preference distributions over a realistic skill inventory. To evaluate personalized skill selection, we construct a simulation environment, TOOLBENCH-60, comprising 60 skills across 10 domains curated from TOOLBENCH (Qin et al., 2023). We model synthetic users by assigning each a latent preference distribution π_u (ranging from deterministic one-hot to stochastic Dirichlet) to govern their habitual choices. These users are evaluated against a mixed query pool: *standard queries* that omit specific skill names to mandate preference recovery, and *explicit queries* that directly name a skill to test zero-shot semantic override capabilities. Full configuration details are provided in Appendix A.1.

5.1.2 Evaluation Protocol and Backbones

Evaluation Metrics. We systematically evaluate the agents' performance and preference alignment using four principal metrics: (1) **Cumulative Regret (Regret):** The standard online learning metric, representing the cumulative cost of suboptimal skill selections over the T interaction rounds. (2) **Test Accuracy (Acc.):** The accuracy evaluated on the held-out test pool at the end of the interaction horizon, measuring the agent's ability to generalize learned preferences to unseen standard and explicit queries without further state updates. (3) **Recovery Rate (R.R.):** Exclusively reported in the *One-hot* preference regime, this metric computes the top-1 hit rate, indicating whether the argmax of the agent's learned probability distribution $\hat{p}_{u,d}$ correctly matches the ground-truth preferred skill $a^*_{u,d}$ across all user-domain pairs in all domains $\mathcal{D}$:

$$\text{R.R.} = \frac{1}{N_u \cdot |\mathcal{D}|} \sum_{u,d} \mathbb{1}\Big[\arg\max_k \hat{p}_{u,d}(k) = a^*_{u,d}\Big],$$

where $a^*_{u,d} := \arg\max_{k \in \mathcal{T}_d} \pi_u(d)(k)$ denotes user u's preferred skill in domain d. (4) **Spearman Rank Correlation (SRC):** Exclusively reported in the *Soft* preference regime, this metric assesses the holistic alignment between the predicted and actual preference distributions by measuring their rank correlation:

$$\text{SRC} = \frac{1}{N_u \cdot |\mathcal{D}|} \sum_{u,d} \rho\big(\text{rank}(\hat{p}_{u,d}),\ \text{rank}(p^*_{u,d})\big),$$

where ρ is the Spearman correlation coefficient.

Backbones. To ensure architectural robustness across varying model capabilities, we evaluate all agents using 3 diverse LLM backbones: GPT-5.2 (OpenAI, 2025b), DeepSeek-V4-Flash (DeepSeek-AI, 2026), and Qwen3-30B-Instruct (Yang et al., 2025). The experimental setup involves $N_u = 50$ users interacting over $T = 500$ rounds, with results averaged across $S = 3$ independent seeds. For details about the hyperparameter configurations and API settings, please refer to Appendix A.2 and Appendix A.3, respectively.

5.1.3 Agents Under Evaluation

We instantiate *nine* agents grouped into four design families to isolate the contribution of each architectural component. Agents within a family share a common decision primitive and differ only in incidental implementation choices; agents across families are intended to exhibit qualitatively different trade-offs. Detailed implementation for these agents is provided in Appendix A.4.

Family I: No learning. **(1) RANDOM** selects a skill uniformly at random from the full skill inventory. **(2) ZEROSHOT-LLM** prompts an LLM to select a skill using only the user's query and the skill descriptions, without any historical user state.

Family II: Statistical only. These agents do not invoke the LLM; they decide purely from statistical priors. **(1) FREQ-GREEDY** greedily selects the skill with the *frequency prior*. **(2) PURE-BANDIT** uses the *bandit prior* to calculate and directly select the skill with the highest UCB score.

Family III: LLM with Memory. These agents use the LLM as the sole decision-maker, differing only in the amount of per-user state exposed to the prompt. **(1) INCONTEXT-MEMORY** prompts an LLM to select a skill by appending the user's five most recent successful skill selections to the standard prompt. **(2) PROFILE-MEMORY** prompts an LLM to select a skill by injecting a complete, serialized log of the user's past successes and attempts for every skill directly into the context.

Family IV: LLM with Statistical Prior. These agents combine a learned statistical estimator with an LLM to balance historical priors with semantic reasoning. **(1) BANDIT-AS-CONTEXT** prompts an LLM to make the final selection by providing the user's query alongside the probability distribution outputted by the *bandit prior*. **(2) FREQ-AS-OVERRIDE** operates as the procedure

Qwen3-30B-Instruct		one-hot			soft-0.3		
		Regret(↓)	Acc.(%, ↑)	R.R.(%, ↑)	Regret(↓)	Acc.(%, ↑)	SRC(↑)
No Learning	Random	492.1 (± 0.2)	1.7 (± 0.1)	15.9	491.2 (± 0.8)	1.7 (± 0.3)	0.000
	ZeroShot-LLM	377.2 (± 3.0)	23.9 (± 0.2)	15.9	377.5 (± 3.6)	24.3 (± 0.6)	0.000
Statistical	Freq-Greedy	167.6 (± 6.0)	72.1 (± 1.1)	86.4	328.5 (± 1.2)	33.3 (± 1.0)	0.399
	Pure-Bandit	140.2 (± 1.9)	80.4 (± 0.6)	99.9	282.0 (± 2.2)	39.5 (± 1.7)	**0.539**
LLM with Memory	InCotext-Memory	363.9 (± 3.8)	27.2 (± 0.2)	62.5	372.7 (± 3.7)	25.4 (± 0.7)	0.282
	Profile-Memory	269.5 (± 4.5)	53.4 (± 0.6)	70.9	344.2 (± 4.3)	32.9 (± 0.7)	0.271
LLM with Statistical Prior	Bandit-as-Context	344.2 (± 1.5)	34.1 (± 0.3)	82.7	369.6 (± 2.9)	26.4 (± 0.6)	0.373
	Freq-as-Override	**126.3 (± 4.5)**	82.5 (± 1.2)	92.5	295.3 (± 2.9)	41.6 (± 0.6)	0.288
	Bandit-as-Override	135.7 (± 0.7)	**84.3 (± 0.7)**	**100.0**	**264.8 (± 2.4)**	**46.2 (± 0.6)**	**0.539**

Table 1: **Aggregate performance of nine skill-selection agents evaluated across varying user preference regimes on Qwen3-30B-Instruct.** Performance is measured by Cumulative Regret (Regret, ↓), Test Accuracy on the held-out pool (Acc., ↑), Recovery Rate (R.R., ↑), and Spearman Rank Correlation (SRC, ↑).

described in Section 3.3 with *frequency prior*. **(3)** **BANDIT-AS-OVERRIDE** operates as the procedure described in Section 3.3 with *bandit prior*.

5.2 Main Results

Table 1 presents the aggregate performance on Qwen3-30B-Instruct; results on DeepSeek-V4-Flash and GPT-5.2 appear in Appendix C. We synthesize three primary findings:

Neither Semantics Nor Statistics Alone Suffices. The NO LEARNING family consistently exhibits the highest cumulative regret, confirming that zero-shot prompting cannot deduce latent habits without historical signals. Conversely, purely statistical agents (FREQ-GREEDY, PURE-BANDIT) attain competitive Regret and R.R. but are bottlenecked on Accuracy by their inability to process explicit overrides. Both dimensions are indispensable.

Exploration Outperforms Frequency Counting. Bandit-based primitives systematically outperform frequency counting in R.R. and SRC, with the gap widening under the *Soft* regime that better reflects stochastic real-world behaviors. Principled exploration is necessary for unearthing nuanced preferences within limited interaction rounds.

Decoupling Beats Prompt-Injected Memory. The LLM WITH STATISTICAL PRIOR family consistently achieves the lowest Regret and highest Test Accuracy across all backbones, while memory-augmented baselines (INCONTEXT-MEMORY, PROFILE-MEMORY), the prevailing paradigm in commercial agents, incur significantly higher regret. The difference is not in *what* is remembered but in *what makes the decision*.

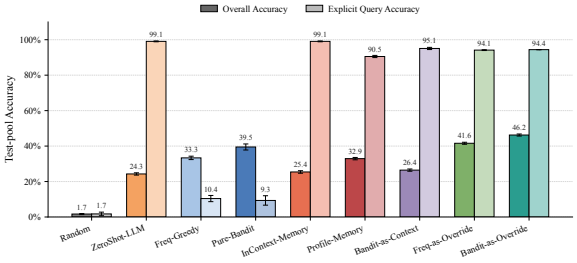


Figure 3: **Performance breakdown on explicit queries using the Qwen3-30B-Instruct backbone (Soft-0.3 regime).** Unlike purely statistical agents that fail on zero-shot instructions, our LOCAL HARNESS achieves near-perfect execution by reserving the LLM exclusively as a semantic exception handler.

5.3 Fine-Grained Evaluation

Beyond the following results, we also present analysis of cumulative regret in Appendix E, rolling accuracy in Appendix F, per-domain accuracy in Appendix G, and preference recovery in Appendix H.

5.3.1 Detailed Accuracy Analysis

To provide a detailed accuracy analysis, we present a performance breakdown on explicit queries evaluated on Qwen3-30B-Instruct in Figure 3. We also present the results on DeepSeek-V4-Flash and GPT-5.2 via Figure 5 and Figure 6 in Appendix D, respectively. These results highlight a critical vulnerability in purely statistical agents (FREQ-GREEDY, PURE-BANDIT): while adept at modeling latent habits, their accuracy drops precipitously on explicit queries because they cannot process zero-shot semantic instructions. Furthermore, memory-augmented LLMs struggle to balance dense statistical tracking with these overrides. Our LOCAL HARNESS architecture elegantly resolves this by delegating historical priors to local and reserving the LLM strictly as a semantic exception handler,

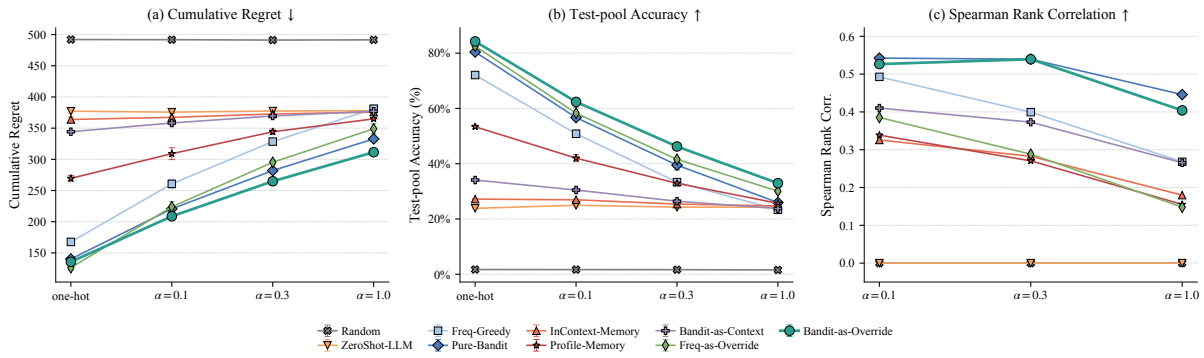


Figure 4: **Performance of evaluated agents across varying levels of user preference evenness (α) using the Qwen3-30B-Instruct backbone.** The Dirichlet concentration parameter α sweeps from a deterministic *one-hot* preference to a uniform distribution at $\alpha = 1.0$. The subfigures report (a) Cumulative Regret (↓), (b) Test-pool Accuracy (↑), and (c) Spearman Rank Correlation (↑). Across all evenness levels, BANDIT-AS-OVERRIDE uniquely and consistently attains the lowest regret and highest test-pool accuracy.

achieving near-perfect execution on explicit instructions and validating our decoupled design.

5.3.2 Evenness of User Preference

To evaluate the impact of user preference evenness, we sweep the Dirichlet concentration parameter $\alpha \in \{\text{one-hot}, 0.1, 0.3, 1.0\}$. Our analysis yields three primary findings. First, BANDIT-AS-OVERRIDE consistently maintains the lowest cumulative regret and highest test accuracy across all preference distributions. Second, the performance advantage of BANDIT-AS-OVERRIDE over the exploration-free FREQ-AS-OVERRIDE widens monotonically as preferences transition from deterministic to highly stochastic (increasing α). This underscores that principled exploration is critical for mitigating the underestimation bias inherent in greedy frequency counting under stochastic regimes. Finally, the learned posterior of BANDIT-AS-OVERRIDE closely tracks that of PURE-BANDIT, verifying that our decoupled LLM semantic override channel effectively preserves the integrity of the statistical learning signal.

5.4 Correspondence with the Theory

Theorem 4.1 makes three qualitative predictions that align with our empirical findings. **(i) Mixture decomposition is necessary, not incidental.** The proof relies on the explicit convex-combination form $P'_t = (1-\lambda)h_t + \lambda P$ both for the convexity argument of Lemma B.2 and for the linearity argument that combines the two lemmas. Agents that route the statistical signal and the semantic signal through a single LLM call (BANDIT-AS-CONTEXT, PROFILE-MEMORY, INCONTEXT-MEMORY) do not admit such a decomposition, and the bound does not cover them, which is consistent with their markedly higher regret across all three backbones. **(ii) Backbone-invariance of the ranking.** Because Theorem 4.1 depends on the LLM only through the constant $|R(P) - R(Q)|$ and not through any model-specific quantity, it predicts that the relative ordering of methods should be preserved across backbones, while absolute regret levels may shift. This is precisely what we observe across varying backbones in the main results. **(iii) Sensitivity to preference evenness.** The frequency prior is purely exploitative and therefore more sensitive than bandit to the evenness of Q: as the preference distribution flattens, distinguishing the optimal skill from its near-ties requires increasingly fine-grained estimates, inflating C_{prior}. The bound therefore predicts a widening gap between BANDIT-AS-OVERRIDE and FREQ-AS-OVERRIDE as α increases. Figure 4 confirms this: Δ_{regret} grows monotonically from 9.4 at one-hot to 38 at $\alpha = 1.0$. The theory thus accounts for both the architectural ranking and its directional sensitivity to preference structure.

6 Conclusion

We introduced LOCAL HARNESS, an architecture for preference-driven skill selection in locally deployed personal agents that strictly decouples statistical preference learning from semantic intent parsing. By delegating user habit modeling to a lightweight local estimator and reserving the remote LLM as a semantic exception handler, our framework synergizes statistical personalization with semantic reasoning, establishing a robust paradigm for personalized agent designs.

Limitations

While the LOCAL HARNESS architecture successfully demonstrates the advantages of decoupling statistical preference learning from semantic intent parsing, our study has several limitations. First, since preference-driven skill selection is a newly identified problem with no existing simulation environment, we constructed TOOLBENCH-60 as a first dedicated sandbox to enable principled evaluation. While this benchmark already captures the core dynamics of implicit preferences across diverse domains and preference regimes, it currently models stationary user profiles; extending it to capture non-stationary temporal shifts and richer contextual dependencies, as well as broadening to additional benchmarks, is a natural next step that our open-sourced framework readily supports. Second, the current sequential decision-making formulation assumes immediate, explicit binary reward signals, though practical user feedback is frequently sparse, delayed, or noisy. Third, to ensure the local harness remains lightweight and free of external model dependencies, we employ deterministic feature hashing, which may lack the representational capacity of dense neural text embeddings to capture highly nuanced syntactic variations. Finally, while we removed the remote LLM from the high-frequency execution critical path, the framework still relies on capable remote foundation models to process the semantic override probe; evaluating its viability when paired with smaller on-device models remains an important area for future exploration.

References

Anthropic. 2025. Claude code: Anthropic's agentic coding system.

DeepSeek-AI. 2026. Deepseek-v4: Towards highly efficient million-token context intelligence.

Zeyu Gan, Ruifeng Ren, Wei Yao, Xiaolin Hu, Gengze Xu, Chen Qian, Huayi Tang, Zixuan Gong, Xinhao Yao, Pengwei Tang, and 1 others. 2026. Beyond the black box: Theory and mechanism of large language models. *arXiv preprint arXiv:2601.02907*.

Google. 2025. Gemini deep research — your personal research assistant.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, Senjie Jin, Jiejun Tan, Yanbin Yin, Jiongnan Liu, Zeyu Zhang, Zhongxiang Sun, Yutao Zhu, Hao Sun, Boci Peng, and 28 others. 2026. Memory in the age of ai agents. *Preprint*, arXiv:2512.13564.

Haojian Huang, Jiahao Shi, Yinchuan Li, and Yingcong Chen. 2026. Affordance agent harness: Verification-gated skill orchestration. *Preprint*, arXiv:2605.00663.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2026. LLMs get lost in multi-turn conversation. In *The Fourteenth International Conference on Learning Representations*.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. 2026. Meta-harness: End-to-end optimization of model harnesses. *Preprint*, arXiv:2603.28052.

Junjie Li, Xi Xiao, Yunbei Zhang, Chen Liu, Lin Zhao, Xiaoying Liao, Yingrui Ji, Janet Wang, Jianyang Gu, Yingqiang Ge, Weijie Xu, Xi Fang, Xiang Xu, Tianchen Zhao, Youngeun Kim, Tianyang Wang, Jihun Hamm, Smita Krishnaswamy, Jun Huan, and Chandan Reddy. 2026. Agent harness engineering: A survey.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, and Yu-Gang Jiang. 2026. Agentic harness engineering: Observability-driven automatic evolution of coding-agent harnesses. *Preprint*, arXiv:2604.25850.

Jianghao Lin, Xinyuan Wang, Xinyi Dai, Menghui Zhu, Bo Chen, Ruiming Tang, Yong Yu, and Weinan Zhang. 2025. Masstool: A multi-task search-based tool retrieval framework for large language models. *Preprint*, arXiv:2507.00487.

Ryan Lopopolo. 2026. Harness engineering: leveraging codex in an agent-first world.

Mario Zechner. 2026. Pi coding agent.

Monica. 2025. Manus: Hands on ai.

Nous Research. 2026. Hermes agent — the agent that grows with you.

OpenAI. 2025a. Codex: Ai coding partner from openai.

OpenAI. 2025b. Introducing gpt-5.2.

Linyue Pan, Lexiao Zou, Shuo Guo, Jingchen Ni, and Hai-Tao Zheng. 2026a. Natural-language agent harnesses. *Preprint*, arXiv:2603.25723.

Wenbo Pan, Shujie Liu, Xiangyang Zhou, Shiwei Zhang, Wanlu Shi, Mirror Xu, and Xiaohua Jia. 2026b. M*: Every task deserves its own memory harness. *Preprint*, arXiv:2604.11811.

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

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2024. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565.

Perplexity AI. 2025. Introducing perplexity deep research.

Peter Steinberger. 2026. Openclaw — personal ai assistant.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *Preprint*, arXiv:2307.16789.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning, 2023. *URL https://arxiv.org/abs/2303.11366*, 8.

Yinjie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. 2026. Openclaw-rl: Train any agent simply by talking. *Preprint*, arXiv:2603.10165.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 1113–1120.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Guibin Zhang, Muxin Fu, and Shuicheng YAN. 2026. Memgen: Weaving generative latent memory for self-evolving agents. In *The Fourteenth International Conference on Learning Representations*.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 19724–19731.

Chenyu Zhou, Huacan Chai, Wenteng Chen, Zihan Guo, Rong Shan, Yuanyi Song, Tianyi Xu, Yingxuan Yang, Aofan Yu, Weiming Zhang, Congming Zheng, Jiachen Zhu, Zeyu Zheng, Zhuosheng Zhang, Xingyu Lou, Changwang Zhang, Zhihui Fu, Jun Wang, Weiwen Liu, and 2 others. 2026. Externalization in llm agents: A unified review of memory, skills, protocols and harness engineering. *Preprint*, arXiv:2604.08224.

A Experimental Setup Details

A.1 Sandbox Construction

Skill Inventory. To ground the evaluation in realistic ecosystems, we derive our skill inventory from TOOLBENCH (Qin et al., 2023), treating a tool as a simulation of a skill. From this corpus we curate a balanced subset of 60 skills spanning 10 domains (*Finance*, *Sports*, *Travel*, *Entertainment*, *Gaming*, *Education*, *Communication*, *Location*, *eCommerce*, *Social*), with $K = 6$ skills per domain. We name this sandbox TOOLBENCH-60 in the later contents. Each skill is paired with a short natural-language description, exposed to language-model agents as part of the candidate description.

Query Templates. For each domain, we use an LLM (GPT-5.2 (OpenAI, 2025b)) to author two complementary template banks: **(1) Standard templates**: natural-language user queries that describe an information need *without* naming any specific skill. The correct skill in a domain for such a query is determined entirely by the user's latent preference. Standard templates are shared across users in the same domain; the correct answer differs across users. **(2) Explicit templates**: queries that *explicitly* name a specific skill by quoting it inside the request (e.g., "Use *Twelve Data* to pull the last five years of daily prices for EUR/USD…"). An explicit query establishes a ground-truth target that supersedes the user's habitual choice. These queries enable us to measure whether an agent can break out of a learned habit when the user demands a non-default behavior. A detailed query generation process is presented in Appendix I.

Synthetic Users A user persona is defined by a preference function $\pi_u : \mathcal{D} \to \Delta(\mathcal{T}_d)$ that maps each domain to a distribution over its K skills. We instantiate two preference regimes: **(1) One-hot**

preference. $\pi_u(d)$ places unit mass on a single preferred skill, sampled uniformly. This models a user with strong, deterministic habits and is the simpler benchmark setting. **(2) Soft preference** (Dirichlet). $\pi_u(d) \sim \mathrm{Dir}(\alpha \mathbf{1}_K)$ with concentration α. Smaller α yields more peaked distributions (approaching one-hot), larger α yields flatter distributions. This models a user whose preference is graded rather than absolute. We use $\alpha = 0.3$ throughout the main experiments. We maintain a training pool of 200 queries and a held-out test pool of 50 per user, with a fixed explicit query ratio of $\rho_e = 0.10$.

Per-user query pool. For each user we construct two disjoint pools: a *training pool* of size $|\mathcal{P}_u^{\mathrm{tr}}| = 200$ and a held-out *test pool* of size $|\mathcal{P}_u^{\mathrm{te}}| = 50$, both sampled from the same template banks with an explicit query ratio of $\rho_{\mathrm{e}} = 0.10$. Concretely, each pool contains 90% standard queries and 10% override queries:

- For a *standard* query, we uniformly draw a domain d and a template from the standard bank for d. The ground-truth skill is sampled from $\pi_u(d)$ (one-hot mode collapses this to the preferred skill; soft mode samples per query).
- For an *explicit* query, we uniformly draw a domain d, draw an override target τ from $\mathcal{T}_d$ that is not the user's mode skill $\arg\max \pi_u(d)$, and sample a template from the explicit bank associated with τ. The ground-truth skill is τ.

The two pools are sampled independently with the same procedure but different random draws. The test pool is used only at the end of training (no updates are applied to any agent), eliminating the train–test leakage that would otherwise contaminate online metrics.

Reward signal. The reward at round t is binary, $r_t = \mathbb{1}\{a_t = a_t^\star\}$, where $a_t^\star$ is the ground-truth skill.

Shared domain classifier. A naïve instantiation would expose the ground-truth domain label to agents that consume it (PURE-BANDIT, FREQ-GREEDY, the hybrid agents), giving them an unrealistic advantage. To avoid this, we prepend a *shared* domain classifier to every round: a single LLM call, using the same backbone as the agents, labels the query domain $\hat{d}(q_t)$, and *every* agent observes this label. Agents that do not require an explicit domain (e.g. RANDOM, ZEROSHOT-LLM) are unaffected. This design (i) ensures fair comparison across statistical and language-model agents, (ii) amortizes one LLM call across all agents in a round, and (iii) lets us report domain-classification accuracy as a separate diagnostic ($\approx 94.5\%$ on the TOOLBENCH-60 sandbox with GPT-5.2).

A.2 Hyperparameter Configuration

Online evaluation protocol. Unless otherwise noted, every experiment uses $N_u = 50$ users per seed, $T = 500$ interaction rounds per user, and $S = 3$ independent random seeds. Round budget is chosen so that each $(\text{user}, \text{domain})$ pair receives, in expectation, $T/|\mathcal{D}| = 50$ interactions.

Bandit hyperparameters. All bandit-based agents (PURE-BANDIT, BANDIT-AS-CONTEXT, BANDIT-AS-OVERRIDE) use LINUCB (Li et al., 2010) with exploration coefficient $\alpha_{\mathrm{ucb}} = 1.0$ and one independent arm per $(\text{user}, \text{domain}, \text{skill})$ triple. The context vector is a 96-dimensional feature-hashed representation (Weinberger et al., 2009) constructed by concatenating three sign-hashed sub-vectors: 64 dimensions for the query string, 16 for the skill name, and 16 for the (inferred) domain. Feature hashing avoids vocabulary construction and is robust to the diverse, multi-domain natural-language queries in TOOLBENCH-60. We adopt hashing rather than a learned text encoder (e.g. a sentence transformer) by design: the local harness is intended to run on commodity hardware without external model dependencies, and hashing is the canonical lightweight encoder for online linear estimators in this regime.

Bandit-to-LLM interface. When the bandit's posterior must be exposed to a language model (BANDIT-AS-CONTEXT), we convert the per-skill UCB scores $\{s_1, \ldots, s_K\}$ to a probability vector via tempered softmax, $p_i = \exp(\beta s_i)/\sum_j \exp(\beta s_j)$, with temperature $\beta = 3.0$. We chose this value empirically.

LLM decoding. All LLM calls (domain classification, agent selection, and the override yes/no probe used by BANDIT-AS-OVERRIDE and FREQ-AS-OVERRIDE) use temperature $\tau_{\mathrm{llm}} = 0.0$ for deterministic decoding. Each prompt requests a strict JSON object as output. A response that fails JSON parsing falls back to regular-expression and substring matching against the candidate skill list; on total failure, the first candidate is returned. Empirically fall-back fires on $\lesssim 0.5\%$ of calls with the

evaluated backbones.

Memory-based baselines. The recency-weighted baseline INCONTEXT-MEMORY retains the most recent $K_m = 5$ successful selections per $(\text{user}, \text{domain})$ pair. The structured baseline PROFILE-MEMORY maintains exact $(\text{successes}, \text{attempts})$ counts per $(\text{user}, \text{domain}, \text{skill})$ and renders them into the LLM prompt as a deterministic profile string.

Override probe. The override-based agents (BANDIT-AS-OVERRIDE, FREQ-AS-OVERRIDE) issue a single binary probe to the LLM per round, structured to elicit a strict JSON answer of the form `{"override": true, "tool": "<name>"}` or `{"override": false}`. The probe lists the domain's skill names but omits their descriptions.

A.3 LLM Call and API

Language-model backbones. A central methodological commitment of this work is that any architectural claim must be validated across LLM backbones of differing strength. We therefore evaluate all agents on the following three models, spanning capability tiers:

- GPT-5.2 (OpenAI, 2025b) (OpenAI; frontier-tier general-purpose model)
- DEEPSEEK-V4-FLASH (DeepSeek-AI, 2026) (DeepSeek; mid-tier cost-optimized instruction-tuned model)
- QWEN3-30B-A3B-INSTRUCT-2507 (Yang et al., 2025) (Alibaba; mid-tier instruction-tuned MoE model)

This selection covers both general-purpose and reasoning-specialized models at different capability tiers, demonstrating that the proposed architecture's advantage holds across all four, which is essential to our methodological claim. The same prompts and parsing logic are used across different LLM backbones.

Intra-round parallelism. Within a single round, up to seven LLM calls are required: one for the shared domain classifier and one each for the up-to-six language-model agents. We dispatch these in parallel via a per-simulation thread pool, preserving deterministic semantics by (i) keeping all non-LLM agents on the main thread to fix the global random-number-generator interleaving, (ii) submitting each LLM agent's `select_tool` call as an independent future, and (iii) collecting results and applying agent state updates in the original agent order after every future has resolved.

Reproducibility. Each experiment is fully described by the tuple (benchmark file, backbone name, seed list, preference mode, and the remaining hyperparameters listed in Section A.2). All sources of stochasticity (user generation, query sampling, LinUCB context hashing, and the LLM decoder) are either explicitly seeded or deterministic. A two-level checkpointing scheme (per-seed and per-user-within-seed) lets an interrupted run resume without recomputation; resumed runs are byte-identical to their uninterrupted counterparts. Code, benchmark JSON files, and a reference run script are released with the paper.

Compute budget. A single main run ($N_u = 50$, $T = 500$, $S = 3$) issues approximately 3.75×10^5 LLM calls per backbone per preference regime (one for domain classification plus six for the LLM-using agents, across $N_u \cdot T \cdot S$ rounds, plus $\sim 1.5 \times 10^4$ calls for the held-out test evaluation). At typical hosted-API latencies our parallelized implementation completes one such run in 6–10 hours of wall-clock time per backbone. The full set of main experiments and ablations reported in this paper consumes approximately 200 hours of cumulative API time.

A.4 Agents Under Evaluation

We instantiate *nine* agents grouped into four design families to isolate the contribution of each architectural component. Agents within a family share a common decision primitive and differ only in incidental implementation choices; agents across families are intended to exhibit qualitatively different trade-offs.

Family I: No learning.

- **RANDOM** selects a skill uniformly at random from the full inventory $\mathcal{T}$. It serves as a basic baseline.
- **ZEROSHOT-LLM** prompts the LLM with the full skill inventory (names plus their textual descriptions) and the user's query, with no per-user state. It is the natural lower bound for any LLM-based personalized agent.

Family II: Statistical only. These agents do not invoke the LLM at decision time; they decide

purely from the (action, reward) stream accumulated for the current user.

- **FREQ-GREEDY** maintains a per-$(\text{user}, \text{domain}, \text{skill})$ success-rate table and greedily selects the skill with the highest empirical rate; ties and untried skills are broken by a single round of round-robin initialization. It tests whether naïve frequency counting suffices in the absence of exploration.

- **PURE-BANDIT** runs LINUCB as described in Section A.2. At each round, it computes a UCB score for each skill in the (inferred) domain and selects the argmax. The contrast with FREQ-GREEDY isolates the value of principled exploration; the contrast with the hybrid agents below isolates the value of LLM-driven override.

Family III: LLM with Memory. These agents use the LLM as the primary (and sole) decision-maker; they differ only in how much per-user state they expose to the prompt. This family is included specifically to represent the architectural choices behind current production AI agents.

- **INCONTEXT-MEMORY** additionally appends the $K_m = 5$ most recent successful selections for the current $(\text{user}, \text{domain})$ to the prompt, modelling a recency-weighted personalization signal of the kind that emerges in long chat sessions.

- **PROFILE-MEMORY** maintains an exact bookkeeping of $(\text{successes}, \text{attempts})$ per $(\text{user}, \text{domain}, \text{skill})$ across the entire horizon and serializes it into the prompt as a structured profile, e.g. "*Tool X: 8/10 successful (80%).*" This is, to our knowledge, the most common LLM-with-memory baseline currently feasible: it discards no information and is supplied to the model in a parsed, lossless form. It is also the most direct analogue of production memory-augmented agents such as OpenClaw (Peter Steinberger, 2026) and Claude Code (Anthropic, 2025), in which structured user facts are written to a persistent store and re-injected into the LLM context at decision time.

Family IV: LLM with Statistical Prior. These agents combine a learned statistical estimator with an LLM in different ways; they are the locus of our methodological contribution.

- **BANDIT-AS-CONTEXT** exposes the bandit's posterior to the LLM as part of its selection prompt. Concretely, for each skill in the inferred domain, the prompt lists the skill name together with its tempered-softmax probability under the current LINUCB posterior (Section A.2). The LLM is asked to integrate this prior with the query semantics and emit a final selection. BANDIT-AS-CONTEXT is the canonical "LLM-as-Bayesian-reasoner" instantiation of the hybrid idea and represents the natural *a priori* expectation for how a bandit and an LLM should be combined.

- **FREQ-AS-OVERRIDE** defaults to operate the same greedy frequency rule as FREQ-GREEDY. A separate, single LLM call asks a binary question: "*does the query explicitly name a tool by name?*"; if so, the LLM returns the named skill and overrides the frequency default. By holding the override mechanism constant and replacing LINUCB with frequency counting, this agent isolates the contribution of *bandit exploration*.

- **BANDIT-AS-OVERRIDE** is identical in structure to FREQ-AS-OVERRIDE except that the default selection is produced by LINUCB. The LLM is asked only the same binary override question, never to perform the primary selection. This realizes our central design hypothesis: the statistical estimator handles personalization (a credit-assignment problem) and the LLM is invoked only as a narrow exception handler for queries whose surface form supersedes the learned preference.

Common interface. Every agent implements the same two-method interface: $\texttt{select_tool}(q, \hat{d}, u, \mathcal{T}) \rightarrow a$ and $\texttt{update}(q, \hat{d}, u, a, r)$. State updates are performed only inside `update`; selection is read-only with respect to agent state. This shared contract makes every comparison in Section 5 an apples-to-apples swap of the decision primitive while holding the surrounding pipeline, domain classifier, query distribution, and evaluation protocol fixed.

B Proofs for Section 4

We use the notation introduced in Section 4. All expectations $\mathbb{E}[\cdot]$ are over the randomness of the

interaction history $\{(q_\tau, a_\tau, r_\tau)\}_{\tau<t}$ that generates h_t, unless otherwise specified. The proof of Theorem 4.1 proceeds in two steps. Lemma B.1 bounds the regret gap between any two policies by their total-variation distance; Lemma B.2 exploits the mixture structure of P'_t to show that this distance is in expectation strictly below $\mathrm{TV}(P, Q)$. The strict separation in regret then follows from the linearity of R.

B.1 Lemma 1: Regret–TV Transfer

Lemma B.1 (Regret–TV transfer)**.** *For any* $\tilde{P}(\cdot \mid q) \in \Delta(\mathcal{T}_{\hat{d}})$,

$$\big|R(\tilde{P}) - R(Q)\big| \;\le\; 2\,\mathbb{E}_q\big[\mathrm{TV}(\tilde{P}(\cdot \mid q),\, Q(\cdot \mid q))\big]. \tag{8}$$

Proof. By the linearity of R in Equation (4),

$$R(\tilde{P}) - R(Q) \;=\; \mathbb{E}_q\big\langle Q - \tilde{P},\, Q\big\rangle.$$

Fix q and apply Hölder's inequality:

$$\begin{aligned} \big|\langle Q - \tilde{P},\, Q\rangle\big| \;&\le\; \|Q - \tilde{P}\|_1 \cdot \|Q\|_\infty \\ &\le\; \|Q - \tilde{P}\|_1 \\ &=\; 2\,\mathrm{TV}(\tilde{P}, Q), \end{aligned} \tag{9}$$

where the second inequality uses $\|Q\|_\infty \le 1$ (since Q is a probability vector) and the final equality is the standard L^1 representation $\mathrm{TV}(P_1, P_2) = \frac{1}{2}\|P_1 - P_2\|_1$. Taking expectation over q yields Equation (8). □

This lemma is the discrete, $0/1$-loss specialization of the IPM-based (Integral Probability Metric) loss-transfer principle: with the function class $\mathcal{H} = \{h : \|h\|_\infty \le 1\}$ the integral probability metric reduces to TV, and the corresponding Lipschitz constant of the loss is at most 2.

B.2 Lemma 2: Mixing Reduces TV

Lemma B.2 (Mixing reduces TV and convergence rate)**.** *Let* $P'_t = (1-\lambda)\, h_t + \lambda\, P$ *with* $\lambda \in [0, 1]$. *Then:*

(a) ***(Mixing reduces TV.)*** *For every* q,

$$\mathrm{TV}(P'_t, Q) \;\le\; (1-\lambda)\,\mathrm{TV}(h_t, Q) + \lambda\,\mathrm{TV}(P, Q). \tag{10}$$

Consequently, if $\mathbb{E}[\mathrm{TV}(h_t, Q)] \to 0$ *and* $\mathrm{TV}(P, Q) > 0$, *there exists* T_0 *such that*

$$\mathbb{E}\big[\mathrm{TV}(P'_t, Q)\big] < \mathrm{TV}(P, Q) \qquad \forall\, t \ge T_0. \tag{11}$$

(b) ***(Parametric convergence rate.)*** *Both statistical priors of Section 3 satisfy*

$$\mathbb{E}[\mathrm{TV}(h_t, Q)] \;\le\; C_{\mathrm{prior}}\, t^{-1/2} \tag{12}$$

for a problem-dependent constant C_{prior}. *In particular, this implies* $\mathbb{E}[\mathrm{TV}(h_t, Q)] \to 0$ *as* $t \to \infty$.

Proof. **Part (a).** The map $\tilde{P} \mapsto \mathrm{TV}(\tilde{P}, Q) = \frac{1}{2}\|\tilde{P} - Q\|_1$ is convex, since the L^1 norm is convex and the affine map $\tilde{P} \mapsto \tilde{P} - Q$ preserves convexity. Applying Jensen's inequality to the convex combination $P'_t = (1-\lambda)h_t + \lambda P$ yields the first inequality. Taking expectation over the randomness of h_t and subtracting $\mathrm{TV}(P, Q)$ from both sides,

$$\begin{aligned} &\mathbb{E}[\mathrm{TV}(P'_t, Q)] - \mathrm{TV}(P, Q) \;\le \\ &(1-\lambda)\big(\mathbb{E}[\mathrm{TV}(h_t, Q)] - \mathrm{TV}(P, Q)\big). \end{aligned} \tag{13}$$

By the consistency assumption (which follows from Part (b)), $\mathbb{E}[\mathrm{TV}(h_t, Q)] < \mathrm{TV}(P, Q)$ for all sufficiently large t, making the right-hand side strictly negative.

Part (b). We verify the rate for both priors:

- *Frequency prior.* Each (u, d, k') cell aggregates Bernoulli reward signals with mean $Q(k')$; Hoeffding's inequality combined with a union bound over the K skills yields $\mathbb{E}[\mathrm{TV}(h_t^{\mathrm{freq}}, Q)] = O(\sqrt{K/t})$.
- *Bandit prior.* The standard LinUCB regret bound (Li et al., 2010) gives cumulative regret $O(d\sqrt{t \log t})$, which translates into a per-round sub-optimality gap of $\tilde{O}(\sqrt{d/t})$. A standard argument relating action sub-optimality to TV distance yields $\mathbb{E}[\mathrm{TV}(h_t^{\mathrm{bandit}}, Q)] = \tilde{O}(\sqrt{d/t})$.

Both bounds are summarized by Equation (12). □

B.3 Proof of Theorem 4.1

Proof. By the linearity of R in Equation (4) and the mixture form (5),

$$\begin{aligned} R(P'_t) \;&=\; R\big((1-\lambda)h_t + \lambda P\big) \\ &=\; (1-\lambda)\, R(h_t) + \lambda\, R(P). \end{aligned} \tag{14}$$

Subtracting $R(Q)$ from both sides and rearranging,

$$R(P'_t) - R(Q) \;=\; (1-\lambda)\big(R(h_t) - R(Q)\big) \;+\; \lambda\big(R(P) - R(Q)\big).$$

Taking absolute value and applying the triangle inequality,

$$\begin{aligned} &\big|R(P'_t) - R(Q)\big| \;\le \\ &(1-\lambda)\,\big|R(h_t) - R(Q)\big| \;+\; \lambda\,\big|R(P) - R(Q)\big|. \end{aligned} \tag{15}$$

Taking expectation over h_t and applying Lemma B.1 to $\tilde{P} = h_t$,

$$\mathbb{E}\left|R(h_t) - R(Q)\right| \;\le\; 2\,\mathbb{E}\big[\mathrm{TV}(h_t, Q)\big].$$

By Lemma B.2 (b), $\mathbb{E}[\mathrm{TV}(h_t, Q)] \to 0$, so given the non-triviality assumption $|R(P) - R(Q)| > 0$ there exists $T_0 < \infty$ such that

$$2\,\mathbb{E}[\mathrm{TV}(h_t, Q)] \;<\; |R(P) - R(Q)| \qquad \forall\, t \ge T_0. \tag{16}$$

Taking expectation over h_t on both sides of Equation (15), we obtain:

$$\begin{aligned}
&\mathbb{E}\left|R(P'_t) - R(Q)\right| \\
&\quad\le\; 2(1-\lambda)\,\mathbb{E}[\mathrm{TV}(h_t, Q)] \;+\; \lambda\,|R(P) - R(Q)| \\
&\quad<\; (1-\lambda)\,|R(P) - R(Q)| \;+\; \lambda\,|R(P) - R(Q)| \\
&\quad=\; |R(P) - R(Q)|,
\end{aligned}$$

where the second inequality uses Equation (16). This is precisely Equation (6). □

Quantitative bound. Substituting the explicit rate $\mathbb{E}[\mathrm{TV}(h_t, Q)] \le C_{\text{prior}}\, t^{-1/2}$ into Equation (15) and using Lemma B.1 on the first term yields Equation (7). The threshold T_0 scales as $T_0 = O\big(C_{\text{prior}}^2 / |R(P) - R(Q)|^2\big)$: the larger the LLM's miscalibration to the user, the fewer interactions are needed for LOCAL HARNESS to overtake.

Remark. The proof uses convexity of TV (Lemma B.2) and linearity of R (Equation (4)); both follow from the structural fact that P'_t is a convex combination of h_t and P. Architectures that route both the statistical prior and the semantic signal through a single LLM call (e.g. BANDIT-AS-CONTEXT or PROFILE-MEMORY) do *not* produce a distribution of this form, because the LLM's non-linear processing destroys the convex combination. Theorem 4.1 therefore identifies the *mixture decomposition* as the structural property that distinguishes the LLM-WITH-STATISTICAL-PRIOR family from its memory-augmented counterpart.

C More Main Results on Varying Backbones

To ensure the architectural robustness of our framework across varying model capabilities, we present the aggregate performance evaluated on DeepSeek-V4-Flash and GPT-5.2 in Table 2 and Table 3, respectively. Consistent with the primary findings, the results demonstrate that the BANDIT-AS-OVERRIDE agent consistently achieves the lowest cumulative regret and the highest test accuracy across all backbones and preference regimes.

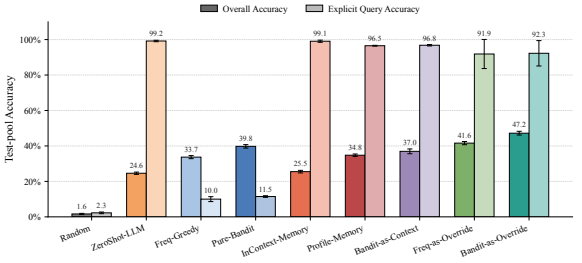


Figure 5: Performance breakdown on explicit queries evaluated using the DeepSeek-V4-Flash backbone under the Soft-0.3 preference regime.

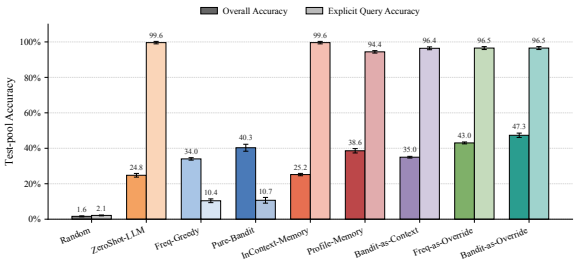


Figure 6: Performance breakdown on explicit queries evaluated using the GPT-5.2 backbone under the Soft-0.3 preference regime.

Purely statistical agents establish strong preference alignment but remain bottlenecked in test accuracy due to their inability to process explicit semantic overrides. Similarly, memory-augmented LLMs yield significantly higher regret and lower accuracy compared to our decoupled approach. These empirical results reinforce the core conclusion: delegating probabilistic credit assignment to a local statistical prior while reserving the remote LLM strictly for complex intent parsing avoids the systemic failures of forcing a single model to manage both tasks simultaneously.

D More Experiments for Test-Pool Accuracy

Figure 5 and Figure 6 provide a detailed accuracy breakdown on explicit queries utilizing the DeepSeek-V4-Flash and GPT-5.2 backbones under the Soft-0.3 preference regime. These fine-grained evaluations highlight a critical vulnerability in purely statistical agents, which suffer a precipitous drop in accuracy on explicit queries because they cannot process zero-shot semantic instructions. Furthermore, memory-augmented LLMs consistently struggle to balance dense statistical tracking with these semantic overrides. The LOCAL HARNESS architecture elegantly resolves this limitation across diverse foundational models; by delegating historical priors to a local bandit and reserving

DeepSeek-V4-Flash		one-hot			soft-0.3		
		Regret(↓)	Acc.(%, ↑)	R.R.(%, ↑)	Regret(↓)	Acc.(%, ↑)	SRC(↑)
No Learning	Random	492.0 (± 0.3)	1.7 (± 0.2)	15.9	491.7 (± 0.5)	1.6 (± 0.3)	0.000
	ZeroShot-LLM	377.1 (± 2.6)	23.8 (± 0.6)	15.9	376.7 (± 4.1)	24.6 (± 0.6)	0.000
Statistical	Freq-Greedy	166.0 (± 3.0)	73.0 (± 0.5)	86.9	328.0 (± 2.0)	33.7 (± 0.9)	0.402
	Pure-Bandit	139.6 (± 2.0)	80.3 (± 0.5)	99.8	280.7 (± 2.2)	39.8 (± 1.0)	**0.549**
LLM with Memory	InCotext-Memory	365.4 (± 2.6)	26.8 (± 0.0)	62.6	372.5 (± 3.8)	25.5 (± 0.7)	0.294
	Profile-Memory	245.1 (± 5.2)	57.1 (± 1.5)	70.9	327.4 (± 4.8)	34.8 (± 0.7)	0.251
LLM with Statistical Prior	Bandit-as-Context	216.6 (± 4.6)	62.9 (± 1.4)	87.9	309.6 (± 5.2)	37.0 (± 1.4)	0.465
	Freq-as-Override	**103.3 (± 1.8)**	87.0 (± 1.0)	94.1	288.0 (± 3.9)	41.6 (± 0.9)	0.273
	Bandit-as-Override	118.9 (± 1.0)	**87.4 (± 0.4)**	**100.0**	**255.9 (± 1.6)**	**47.2 (± 1.1)**	0.523

Table 2: **Aggregate performance of nine skill-selection agents evaluated across varying user preference regimes on DeepSeek-V4-Flash.** Performance is measured by Cumulative Regret (Regret, ↓), Test Accuracy on the held-out pool (Acc., ↑), Recovery Rate (R.R., ↑), and Spearman Rank Correlation (SRC, ↑).

GPT-5.2		one-hot			soft-0.3		
		Regret(↓)	Acc.(%, ↑)	R.R.(%, ↑)	Regret(↓)	Acc.(%, ↑)	SRC(↑)
No Learning	Random	491.8 (± 0.5)	1.9 (± 0.3)	15.9	491.5 (± 0.5)	1.6 (± 0.3)	0.000
	ZeroShot-LLM	376.0 (± 2.6)	24.3 (± 0.6)	15.9	375.5 (± 3.4)	24.8 (± 1.0)	0.000
Statistical	Freq-Greedy	162.4 (± 3.2)	73.6 (± 1.1)	87.3	327.2 (± 2.1)	34.0 (± 0.7)	0.404
	Pure-Bandit	137.1 (± 1.0)	81.0 (± 0.6)	99.9	279.2 (± 1.4)	40.3 (± 2.0)	**0.554**
LLM with Memory	InCotext-Memory	371.5 (± 2.2)	25.4 (± 0.2)	63.5	373.5 (± 4.0)	25.2 (± 0.7)	0.304
	Profile-Memory	208.7 (± 9.6)	64.8 (± 2.2)	67.5	316.0 (± 3.6)	38.6 (± 1.2)	0.193
LLM with Statistical Prior	Bandit-as-Context	266.0 (± 3.5)	53.1 (± 4.9)	94.1	327.1 (± 3.8)	35.0 (± 0.5)	0.473
	Freq-as-Override	**107.2 (± 3.1)**	86.8 (± 0.9)	94.2	286.5 (± 1.9)	43.0 (± 0.5)	0.286
	Bandit-as-Override	122.4 (± 1.6)	**86.9 (± 1.8)**	**100.0**	**256.8 (± 1.4)**	**47.3 (± 1.2)**	0.535

Table 3: **Aggregate performance of nine skill-selection agents evaluated across varying user preference regimes on GPT-5.2.** Performance is measured by Cumulative Regret (Regret, ↓), Test Accuracy on the held-out pool (Acc., ↑), Recovery Rate (R.R., ↑), and Spearman Rank Correlation (SRC, ↑).

the LLM strictly as a semantic exception handler, BANDIT-AS-OVERRIDE achieves near-perfect execution on explicit instructions, thereby thoroughly validating our decoupled design.

E Regret Analysis

Figure 7 reports the full cumulative-regret trajectories. Across all six panels, BANDIT-AS-OVERRIDE and FREQ-AS-OVERRIDE attain the lowest curve throughout training. The two memory-augmented LLM baselines (PROFILE-MEMORY and INCONTEXT-MEMORY) accumulate regret at a near-linear rate, indicating that prompt-injected per-user statistics do not substitute for a locally-fitted estimator. Consistent with the evenness analysis in Section 5.3.2, the gap between BANDIT-AS-OVERRIDE and FREQ-AS-OVERRIDE visibly widens between the top row (one-hot) and the bottom row ($\alpha = 0.3$), confirming that the marginal value of exploration grows as user preference becomes stochastic.

F Rolling Accuracy

Figure 8 traces the per-round convergence dynamics with a sliding window of $W = 50$ rounds. The dashed Oracle line marks the accuracy ceiling of *any* static policy that commits to a single skill per (u, d) pair; under one-hot preferences it equals $1 - \rho_{\mathrm{e}} = 0.9$ by construction, while in the soft regime it is determined by the mode of each user's Dirichlet preference. Two observations stand out. First, the statistical-only agents (FREQ-GREEDY and PURE-BANDIT) plateau *at* the Oracle line, exactly as predicted: lacking access to query semantics, they can at best recover the static optimum. Second, the override-based agents visibly cross above this ceiling, because their LLM probe can re-route explicit-query rounds to the named skill. The curves also show that BANDIT-AS-OVERRIDE converges faster than every alternative across all panels, an advantage that is most pronounced in the early rounds where the preference signal is scarce.

G Per-Domain Accuracy Analysis

Figure 9 decomposes the accuracy along the ten TOOLBENCH-60 domains, displayed as a 9×10

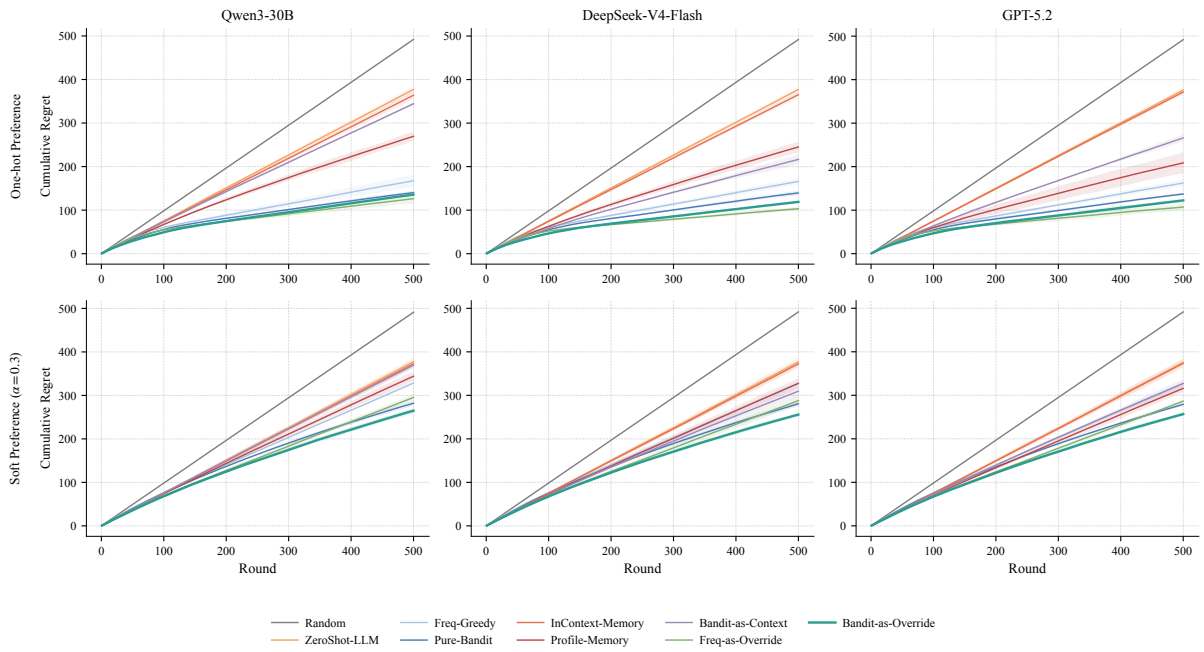


Figure 7: **Cumulative regret over the interaction horizon for the six main experiments.** Rows correspond to the preference regime (top: one-hot; bottom: soft preference with $\alpha = 0.3$); columns correspond to the LLM backbone (QWEN3-30B, DEEPSEEK-V4-FLASH, GPT-5.2). Each panel plots cumulative regret $\sum_{\tau \le t}(1 - r_\tau)$, averaged across $N_u = 50$ users and $S = 3$ independent seeds; shaded bands denote 95% confidence intervals. Lower curves indicate faster convergence and lower aggregate cost. Across all three backbones and both regimes, BANDIT-AS-OVERRIDE (deep teal) attains the lowest regret throughout training, and its advantage over FREQ-AS-OVERRIDE becomes more pronounced under the soft regime.

heatmap per experiment. The decomposition is intended to rule out the hypothesis that the proposed architecture's advantage is driven by one or two particularly easy categories. In every one of the six panels, the bottom two rows of the heatmap (FREQ-AS-OVERRIDE and BANDIT-AS-OVERRIDE) are uniformly the darkest. By contrast, PROFILE-MEMORY and INCONTEXT-MEMORY exhibit substantial cross-domain variability, with several light cells indicating domains where memory-injection is particularly noisy.

H Preference Recovery for Soft-Setting Results

Figure 10 evaluates how well each agent's *learned* preference distribution $\hat{p}_{u,d}(k)$ matches the user's ground-truth Dirichlet preference $\pi_u(d)$ under the soft regime, along three complementary metrics. The most informative comparison is between PURE-BANDIT and BANDIT-AS-OVERRIDE: their bars are indistinguishable on all three metrics and across all three backbones (e.g., Spearman ≈ 0.53 on every panel), demonstrating that interleaving the LLM override channel *does not corrupt the bandit's posterior*. By contrast, FREQ-AS-OVERRIDE attains a competitive Spearman score but catastrophic KL divergence, because frequency counting can recover the *mode* of $\pi_u(d)$ but not its full shape. Memory-augmented LLM baselines show the same pathology in even more pronounced form. Taken together, these results support the architectural separation we propose: the local statistical primitive is responsible for posterior estimation, while the LLM is restricted to a narrow override channel that, by construction, leaves the learned posterior intact.

I Query Templates

This appendix documents the textual prompts used throughout our benchmark. We organize them into three parts: (i) the prompts we feed to an LLM to generate the query template banks, before any experiment is run; (ii) representative examples of the resulting *standard* and *explicit* queries on a subset of domains; and (iii) the prompts that each agent issues to a language model at inference time during the online evaluation.

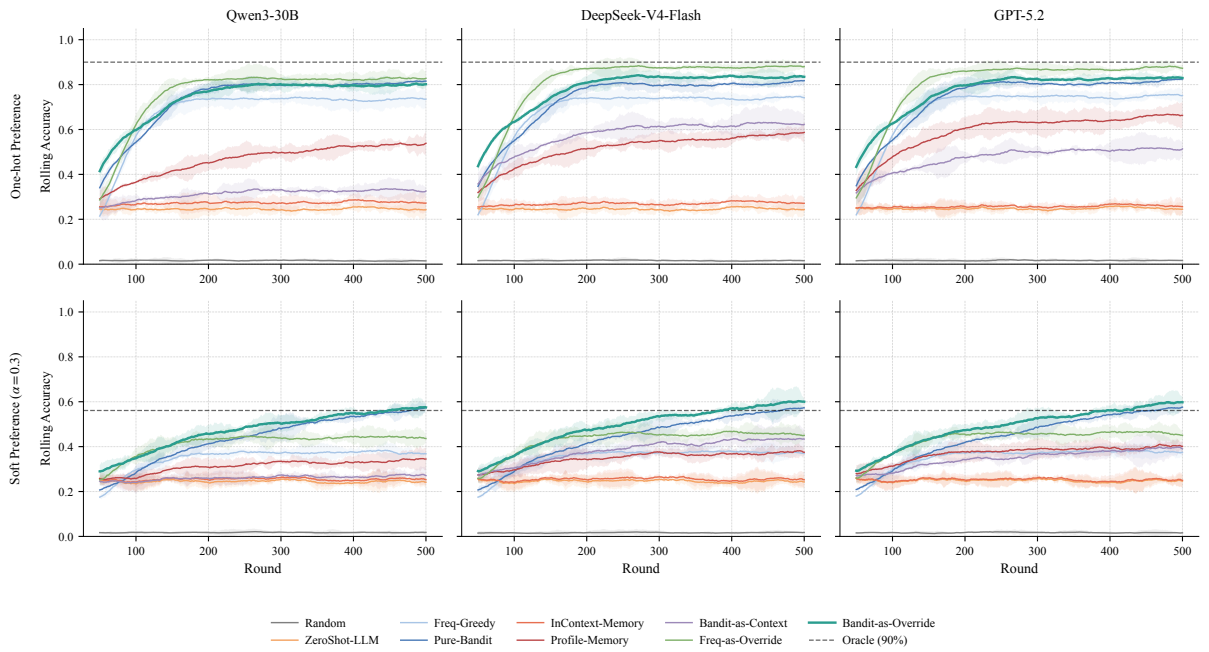


Figure 8: **Rolling per-round accuracy along the interaction horizon for the six main experiments.** For each round t, we plot the mean reward over a sliding window of the previous $W = 50$ rounds, averaged across $N_u = 50$ users and $S = 3$ seeds; shaded bands denote 95% confidence intervals. The dashed Oracle line marks the accuracy ceiling of *any* static policy that commits to a single skill per (u, d) pair; under one-hot preferences it equals $1 - \rho_{\text{ood}} = 0.9$ by construction, while in the soft regime it is determined by the mode of each user's Dirichlet preference. Within each panel, BANDIT-AS-OVERRIDE converges fastest and plateaus closest to the Oracle line; the gap to memory-augmented LLM baselines (PROFILE-MEMORY, INCONTEXT-MEMORY) is large under one-hot preferences and remains substantial under the soft regime.

I.1 Query Template Generation

We construct the benchmark's query bank in a single offline pass using GPT-5.2 (OpenAI, 2025b). For each of the ten TOOLBENCH-60 domains, we issue two distinct prompts. The *standard* prompt produces queries that describe a user's information need in natural language but *deliberately avoid naming any specific tool (skill)*; such queries can be answered correctly only if the agent has inferred the user's underlying preference. The *explicit* prompt produces queries that explicitly name one particular tool (skill) from the domain's six options, so that a competent language model can recover the target tool from the text alone. We generate 20 standard templates per domain and 1-5 explicit templates per tool; the exact counts and the resulting raw text are released alongside the code.

Standard query generation prompt

```
Generate {n_per_domain} diverse user queries
that someone might ask when needing a tool/API
from the "{domain}" category.
Tools in this category:
  - {tool_1}: {description_1}
  - {tool_2}: {description_2}
  . . .
IMPORTANT RULES:
  - Do NOT mention any specific tool name in
the queries
  - Each query should be a natural user request
(1-2 sentences)
  - Cover diverse use cases within this
category
  - Write in English
Return a numbered list (1. query, 2. query,
. . . ).
```

Explicit query generation prompt

```
Generate {n_per_tool} user queries that
explicitly request using "{tool}" (a {domain}
tool: {description}).
IMPORTANT RULES:
  - Each query MUST mention "{tool}" by name
  - The query should be a natural user request
(1-2 sentences)
  - Write in English
Return a numbered list.
```

The dual constraint "do not mention any tool" for standard queries versus "must mention this tool by name" for explicit queries creates a clean binary partition of the benchmark's stimuli along the axis our method is designed to handle: standard queries are decidable only via *personalised* statistical inference, while explicit queries are decidable

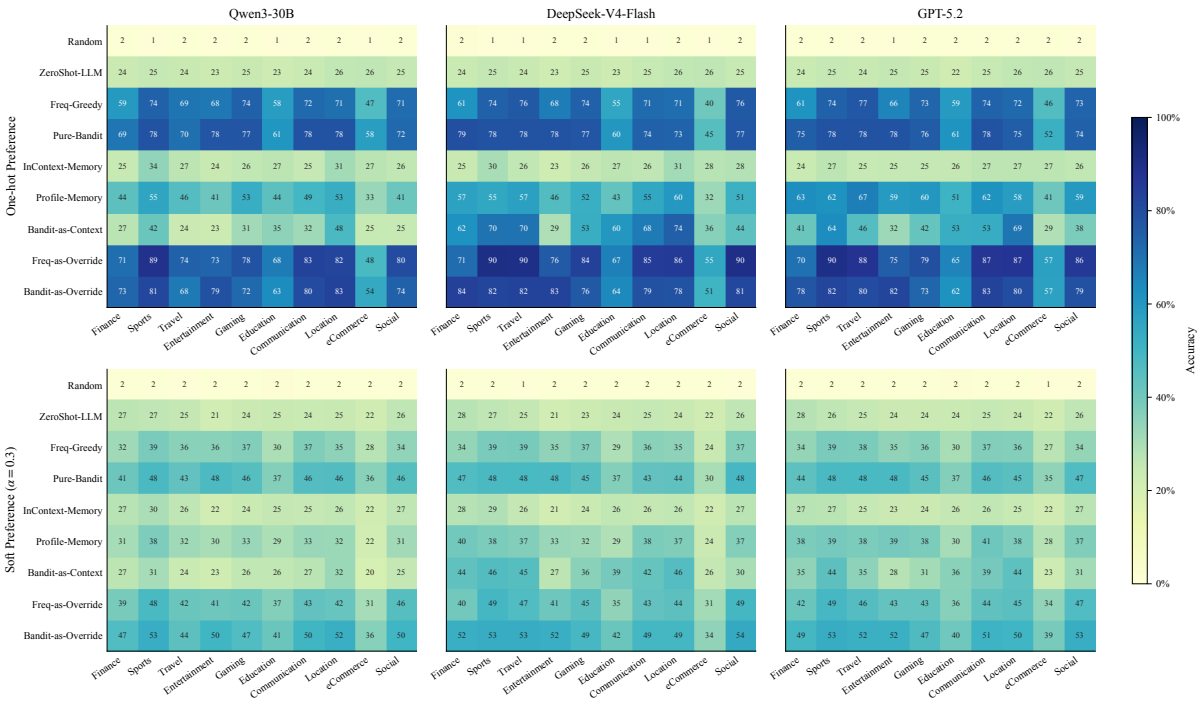


Figure 9: **Per-domain accuracy heatmaps for the six main experiments.** Rows of the figure correspond to the preference regime (top: one-hot; bottom: soft with $\alpha = 0.3$); columns correspond to the LLM backbone. Within each panel, rows are the nine agents and columns are the ten TOOLBENCH-60 domains. Cells report the mean training-round accuracy for that (agent, domain) pair, averaged across $N_u = 50$ users and $S = 3$ seeds, with darker shades denoting higher accuracy (colour scale shown on the right). The override-based family (FREQ-AS-OVERRIDE and BANDIT-AS-OVERRIDE, last two rows of every panel) maintains the highest accuracy uniformly across all ten domains, indicating that the architectural advantage is not concentrated in a particular topical category.

from the text alone. The standard-versus-explicit dichotomy is the structural property that the two-bar test-accuracy decomposition in the main paper makes visible.

I.2 Example Queries by Domain

To give the reader a concrete sense of the linguistic style and difficulty of the synthetic stimuli, we list three illustrative domains from TOOLBENCH-60 below. For each domain we show three standard queries (which require user-preference inference) and three explicit queries that each name a distinct tool (which require only text-level recognition). The full banks of 200 standard and 60 explicit templates across all ten domains are released with the benchmark configuration.

Finance domain

Standard queries:

- I need end-of-day data for major indices and ETFs, plus a way to backfill missing market days automatically.
- Give me daily and historical metrics for thousands of cryptocurrencies, including market cap and volume, with simple pagination and caching support.
- Pull real-time and 10-year historical daily prices for a list of US stocks, and also include splits and dividends where available.

Explicit queries:

- Can you run an *"Extended ACH and Check Prescreen"* on this routing and account number to confirm the account is open and in good standing before we accept an ACH payment?
- Use *Twelve Data* to pull real-time and 1-minute intraday prices for AAPL for today, and return the latest quote plus OHLCV data.
- I want to integrate the *ChangeNOW crypto exchange* into my checkout so customers can buy and exchange crypto; can you outline the steps and what I need to get started?

Travel domain

Standard queries:

- I need to search and compare round-trip flight options from NYC to Tokyo for flexible dates next month, including price trends and the cheapest day to fly.
- Can you find the lowest fares for a multi-city

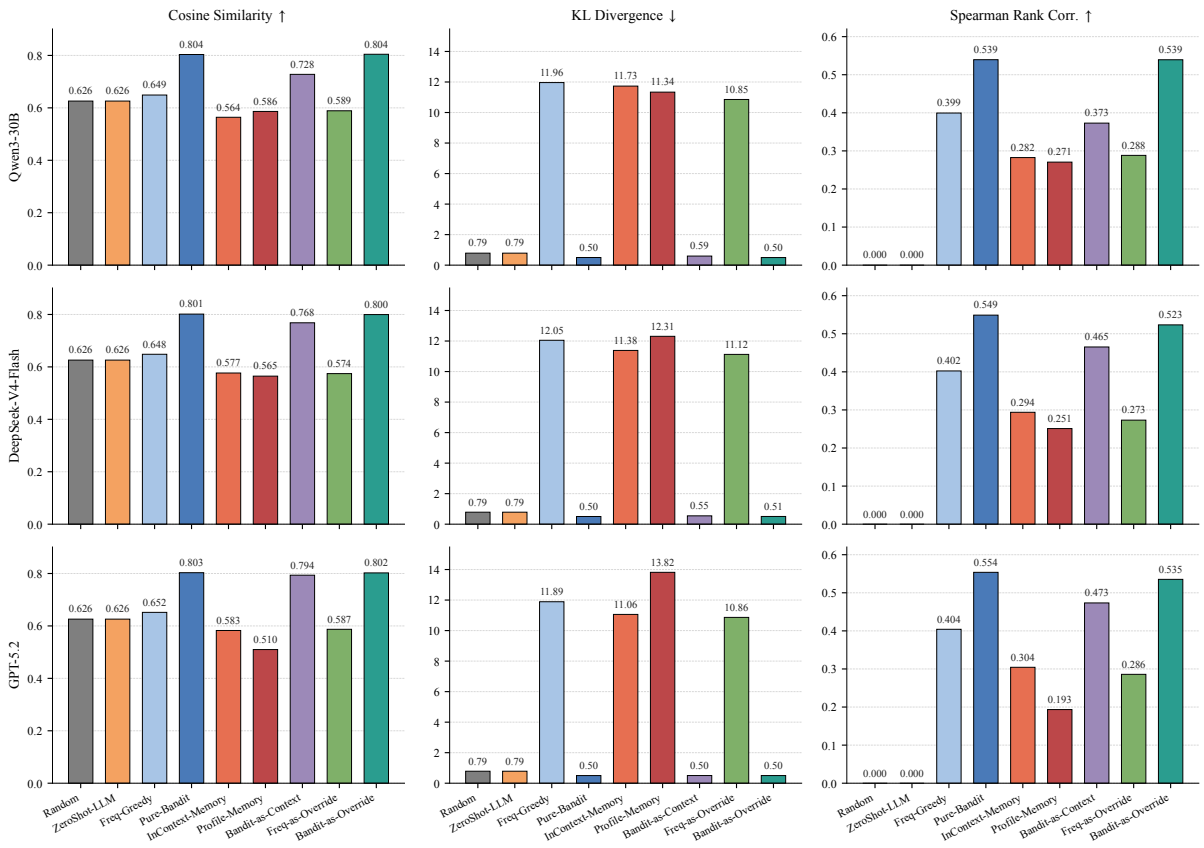


Figure 10: **Preference-distribution recovery for each agent under the soft preference regime ($\alpha = 0.3$).** Rows correspond to the three LLM backbones; columns correspond to the three alignment metrics between each agent's learned per-(u, d) preference $\hat{p}_{u,d}$ and the user's ground-truth Dirichlet distribution $\pi_u(d)$: cosine similarity, KL divergence, and Spearman rank correlation. Arrows in column headers indicate the direction of improvement. All values are aggregated over $N_u \times |\mathcal{D}| = 500$ user–domain pairs per seed across $S = 3$ seeds. PURE-BANDIT and BANDIT-AS-OVERRIDE are indistinguishable on all three metrics across all three backbones, confirming that the LLM override channel of our method does not corrupt the bandit's learned posterior.

itinerary (London → Rome → Athens → London) and return results with cabin class, baggage rules, and cancellation terms?

- I want to monitor a specific route (SFO to LAS) and alert me when the price drops below a threshold within the next 60 days.

Explicit queries:

- Use the *Skyscanner API* to find the cheapest round-trip flights from New York (JFK) to London (LHR) for flexible dates in the next two months, and return the top 10 options.
- Use *Airbnb listings* to pull all active listings in the Lisbon market, including nightly price, cleaning fee, minimum stay, and availability for the next 60 days.
- Use *webcams.travel* to find nearby landscape webcams around my current location, and show the top 10 with a quick preview and distance for each.

Social domain

Standard queries:

- I need to fetch a user's public profile info and most recent posts, then compute engagement metrics and growth over time.
- Can you help me set up phone-number verification via SMS with a single API call, and automatically expire codes after five minutes?
- I want to automate posting to my social account on a schedule, rotate proxies, and avoid triggering anti-bot blocks.

Explicit queries:

- Use *Twitter AIO* to monitor tweets mentioning my brand name in real time and summarize the key themes every hour. Include links, author handles, and engagement stats for the top posts.
- Can you integrate *Branch Metrics* in our iOS and Android apps to track sign-up and purchase events and tie them back to the original referral source?

- Can you use *TikTok Video No Watermark_v2* to download this TikTok and give me the direct no-watermark video link: https://www.tiktok.com/@user/video/1234567890?

A few features are worth pointing out. Standard queries are *intentionally skill-agnostic*: a competent agent that has not learned the user's preference cannot do meaningfully better than uniform random among the six in-domain tools. Explicit queries, by contrast, quote the target skill name (in our shown examples, we render the quoted phrase in italics for readability), so a language model that can read the query can identify the target skill without any history. The two query types together cover the two sub-problems: personalization and explicit override, which motivate our architectural split.

I.3 Inference-Time Prompts

At inference time, every agent that involves an LLM call issues one or two structured prompts per round, each constrained to return a strict JSON object. We list below the three prompts that drive the comparisons reported in the main paper: the shared domain classifier (used by *all* agents that require an inferred domain, including PURE-BANDIT), the binary override probe used by BANDIT-AS-OVERRIDE and FREQ-AS-OVERRIDE, and the chain-of-thought selection prompt used by BANDIT-AS-CONTEXT. For brevity we omit the prompts of the LLM-only baselines (ZEROSHOT-LLM, INCONTEXT-MEMORY, PROFILE-MEMORY), which differ only in what user-history context they prepend to a similar selection instruction.

Shared domain classifier. This prompt is issued exactly once per query and its inferred-domain output is broadcast to every domain-aware agent. The placeholder `{domains_block}` expands to a multi-line listing of the ten domains together with the first four tool names of each. Using a shared classifier guarantees that all agents operate over the same domain signal, removing an otherwise unfair advantage that statistical agents would receive from ground-truth domain access.

Shared domain classifier

```
Which domain does this query belong to?
Domains and their tools:
{domains_block}
Query: "{query}"
Reply with only JSON: {"domain": "<exact
domain name>"}
```

Override probe (BANDIT-AS-OVERRIDE and FREQ-AS-OVERRIDE). The defining design choice of BANDIT-AS-OVERRIDE agent and FREQ-AS-OVERRIDE is to ask the LLM a *single binary question*: does the query explicitly name a tool? The probe deliberately omits tool descriptions and includes only the tool *names* for the agent's inferred domain, so the LLM's task reduces to lexical recognition rather than open-ended selection. This narrow interface is what allows the LLM to act as an exception handler without contaminating the bandit's default selection on the ∼90% of standard queries.

Override probe

```
Does this query explicitly request a specific
tool by name?
Available tools: {tool_1}, {tool_2}, . . . ,
{tool_K}
Query: "{query}"
If the query mentions a specific tool name,
reply:
  {"override": true, "tool": "<tool name>"}
If not, reply:
  {"override": false}
```

Selection-with-prior (BANDIT-AS-CONTEXT). For comparison, the BANDIT-AS-CONTEXT baseline exposes the bandit's posterior to the LLM as a tempered-softmax prior over the domain's six tools, and asks the LLM to perform the *full selection* itself rather than answering an override question. The placeholder `{tools_block}` expands to lines of the form "`- {tool_name}: 38%`" sorted by descending prior.

Selection-with-prior prompt

```
Select the best tool for this query.
Available tools (with learned user preference
%):
{tools_block}
Query: "{query}"
If the query explicitly names a tool, use it.
Otherwise, prefer the tool with the highest
user preference.
Reply with only JSON: {"tool": "<tool name>"}
```

In all three prompts we set the decoding temperature to 0.0 and parse the response as JSON. On parse failure (empty body, malformed JSON, or a tool name that does not appear in the domain's tool list), the client first attempts a regular-expression and substring match against the candidate tools, and only on total failure falls back to the default action (the first listed domain for the classifier, the bandit's selection for the override probe, and the first tool for BANDIT-AS-CONTEXT). Empirically

the fallback path fires on $\lesssim 0.5\%$ of calls with the backbones reported in Appendix A.3.

J License of Scientific Artifacts

Our constructed simulation sandbox, TOOLBENCH-60, is derived from the original TOOLBENCH (Qin et al., 2023) dataset. We acknowledge the creators of ToolBench, which is distributed under the Apache-2.0 License. The newly curated TOOLBENCH-60 benchmark is open-sourced and released under the MIT License to facilitate further research in personalized agent interactions.

K Use of AI Assistant

During the preparation of this work, we utilized AI assistants to enhance productivity in both software development and manuscript preparation. Specifically, Claude Code was employed to assist in drafting, structuring, and debugging the underlying experimental framework and execution pipeline. Additionally, Google's Gemini was utilized to help refine, draft, and polish the textual content of this manuscript. In all cases, we carefully reviewed, verified, and take full responsibility for the accuracy and originality of the submitted work.