Rational Clarification by Assistive Agents via Value-of-Information Reasoning

Anonymous authors
Paper under double-blind review

Abstract

Users of language-based assistive agents often make ambiguous requests. In response, an assistant can either directly act on its interpretation of the request — risking misalignment with the user — or ask a clarifying question. Which option is the most safe and helpful? A common approach is to ask questions that minimize uncertainty about the user's intent until a threshold is reached. However, this neglects the impact of uncertainty reduction on downstream performance, the costs of asking versus acting immediately, and the possibility that users may provide corrections without being asked. To navigate these trade-offs, we introduce *Rational Enquiry via Value-of-Information Reasoning* (REVOIR). REVOIR makes clarification decisions via inference-time reasoning about the *value-of-information* (VoI) of a question, which captures the expected improvement in task reward due to the answer received. In two assistive tasks — ambiguous question answering (CondAmbigQA) and preference-aligned household task planning (ADAPT) — we show that REVOIR achieves greater success with fewer questions than approaches based on prompting, chain-of-thought, fine-tuning, or information gain, improving preference satisfaction on ADAPT by 13–15% over a fine-tuned clarification policy while requiring no training and asking five times fewer questions. Furthermore, when the assistant can receive cheap user corrections after acting, REVOIR naturally infers that asking questions is not always efficient, demonstrating the adaptivity of our approach. In contrast, we find that vanilla reasoning agents fail to adaptively clarify user requests, and request *fewer* clarifications as reasoning effort increases.

1 Introduction

People often express their intentions to AI agents in under-specified ways. Faced with such ambiguity, how should an assistive agent respond? Agents that directly execute the request might misinterpret the user, resulting in misaligned actions with potentially irreversible consequences (Ming, 2025). Instead, an agent might ask clarifying questions to determine the right interpretation, thereby avoiding costly mistakes. However, asking questions may not always be necessary. If the assistant is certain enough about the user's intent, or if the user can easily correct the assistant afterwards, it may be best to act on the request directly.

In this paper, we take a *rational* approach to addressing these trade-offs in assistive agents based on large language models (LLMs):

1. We formulate clarificatory assistance as a cooperative game (Hadfield-Menell et al., 2016), where the assistant starts off uncertain about the user's intent, but can learn by asking clarifying questions or (possibly) receiving corrections after acting upon the user's request — a form of learning neglected by past work on multi-turn clarification.
2. To solve these assistance problems, we introduce **Rational Enquiry via Value-of-Information Reasoning** (**REVOIR**, Figure 1), a decision-theoretic assistant that (a) updates its beliefs about the user's intentions in natural language; (b) estimates the *value-of-information* (VoI) (Howard, 1966; Raiffa & Schlaifer, 1961) of clarificatory actions — i.e., the expected improvement in task reward due to the answer received; (c) uses this to make rational decisions about asking questions vs. acting immediately.

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

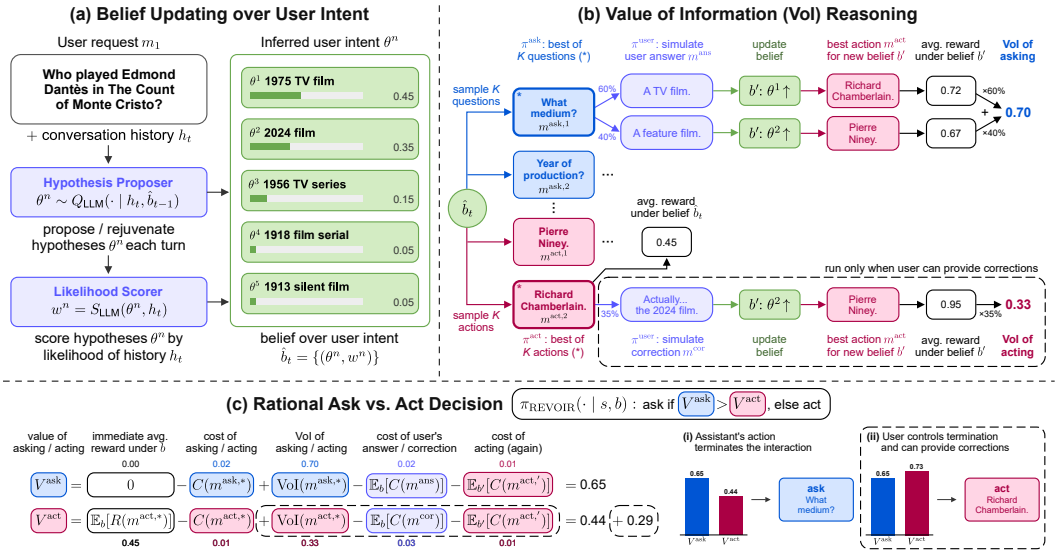


Figure 1: Overview of REVOIR. (a) Given an ambiguous user request, REVOIR infers and updates a belief over the user's intent θ. **(b)** REVOIR computes the value-of-information (VoI) of asking (vs. acting) by simulating the expected benefit of acting after a clarifying question (vs. a user correction). **(c)** REVOIR decides between asking or acting by maximizing the cost-adjusted value of asking V^{ask} vs. acting V^{act}, rationally adapting to cases where: (i) the assistant's action is terminal; (ii) the user can give corrections.

We evaluate REVOIR across ambiguous question answering (CondAmbigQA, Li et al., 2025) and preference-aligned household task planning (ADAPT, Patel et al., 2025). We compare REVOIR against prompting baselines, generic inference-time reasoning, fine-tuning for clarifying questions, and using expected information gain, and find that our approach achieves greater task success and preference alignment with fewer questions. On ADAPT, REVOIR reaches preference satisfaction rates of 56–59% using only in-context examples and no training, exceeding a clarification policy fine-tuned for the task (44%) by 13–15% while asking roughly 5× fewer questions, and surpassing even an "always ask" baseline by around 6%. REVOIR also adapts rationally to a variant of the QA task where users can provide post-hoc corrections — a setting ignored by prior clarification methods (Andukuri et al., 2024; Zhang et al., 2025) — successfully inferring that asking a question often provides no benefit beyond acting now and being corrected later. In contrast, generic reasoners (ReAct + reasoning-trained LLMs) fall short of REVOIR at reasoning about whether further clarifications are helpful, and indeed generally engage in *less* clarifications as reasoning effort is increased.

Related Work. REVOIR is distinct from prior work in several ways. Unlike generic inference-time reasoning (e.g. chain-of-thought Wei et al., 2022; Deng et al., 2023, or ReAct Yao et al., 2022), which lack a normative standard for clarification decisions, REVOIR builds upon the theory of assistance games (Hadfield-Menell et al., 2016; Zhi-Xuan et al., 2024; Ma et al., 2025; Laidlaw et al., 2025), showing how rational assistance under uncertainty can be applied to LLM agents while remaining tractable. Finetuning methods optimize for clarification by maximizing the quality of subsequent assistant responses (Andukuri et al., 2024; Zhang et al., 2025; Patel et al., 2025; Wu et al., 2025; Chi et al., 2024), but typically assume fixed user costs, and that conversations terminate after the assistant commits to an answer (Andukuri et al., 2024; Zhang et al., 2025). In comparison, REVOIR can accommodate user-specified costs for questions, and naturally adapts to interactions where users can choose whether to accept the assistant's response or provide further corrections. Finally, our approach optimizes the cost-adjusted VoI of a question rather than its expected information gain (EIG) (Lindley, 1956; MacKay, 1992), a metric often used to select uncertainty-reducing questions until a fixed budget or confidence threshold is reached (Hu et al., 2024; Handa et al., 2024; Piriyakulkij et al., 2023; Grand et al., 2025). Apart from neglecting the variable costs of questions, EIG is insensitive to downstream task reward, and may score questions highly even if they do not provide task-relevant information.

2 Clarificatory Assistance as a Cooperative Game

How should an assistive agent rationally decide whether to clarify user requests? REVOIR builds on the framework of *assistance games* (Hadfield-Menell et al., 2016), which formalizes how a user-aligned agent should provide assistance under uncertainty about what the user wants. We adapt this framework to the setting of LLM-based assistive agents, which interact with a user via language while (optionally) taking actions in an external environment:

Definition 2.1. A *conversational assistance game* is a tuple $\langle \mathcal{S}, \{\mathcal{A}^{\text{user}}, \mathcal{A}^{\text{agent}}\}, T, \Theta, R, P_0 \rangle$. Each element has the following definitions:

States. The state space factorizes as $\mathcal{S} = \mathcal{W} \times \mathcal{H}$, where $w \in \mathcal{W}$ is a state of the external world (e.g., a physical room or code-base), and $h_t \in \mathcal{H}$ is a conversational history $h_t = (m_1, m_2, \ldots, m_t)$ composed of alternating user and agent messages m_t. A subset of $\mathcal{S}^{\text{term}} \subseteq \mathcal{S}$ are terminal states, depending on whether the final message m_t ends the conversation.

Actions. $\mathcal{A}^{\text{user}}$ and $\mathcal{A}^{\text{agent}}$ are the user and agent action spaces. Each action space $\mathcal{A}^i = \mathcal{X}^i \cup \mathcal{M}$ decomposes into external actions $\mathcal{X}^i$ and messages $\mathcal{M}$. We further assume that $\mathcal{M}$ can be categorized into user requests $\mathcal{M}^{\text{req}}$, agent questions $\mathcal{M}^{\text{ask}}$, user answers $\mathcal{M}^{\text{ans}}$, verbal actions $\mathcal{M}^{\text{act}}$, user corrections $\mathcal{M}^{\text{cor}}$, and end-of-conversation messages $\mathcal{M}^{\text{end}}$.

Transitions. The transition function $T(s_t \mid s_{t-1}, a_t^{\text{user}}, a_t^{\text{agent}})$ factorizes into world and conversation transitions. We assume players take turns acting, so $a_t^{\text{user}} = \bot$ when $a_t^{\text{agent}} \neq \bot$ and vice versa; let a_t denote the acting player's action. External actions ($a_t \in \mathcal{X}$) update the world via $T^{\text{world}}(w_t \mid w_{t-1}, a_t)$ while leaving the conversational history unchanged.

User Intents. Θ is the space of user intents $\theta \in \Theta$, which we represent in natural language. Crucially, θ is initially *unknown* to the agent, and is not fully revealed by the user's initial action. An intent θ determines the goal reward associated with a terminal state $s \in \mathcal{S}^{\text{term}}$.

Rewards and Costs. The user's reward function, shared by the agent, decomposes as $R(s_t, a_t^{\text{user}}, a_t^{\text{agent}}, \theta) := G(s_t, \theta) - C(a_t^{\text{user}}, a_t^{\text{agent}})$. Here, $G(s_t, \theta)$ is a (weakly) positive *goal reward* dependent on the hidden intent θ, realized only when $s_t \in \mathcal{S}^{\text{term}}$ is terminal. Unlike clarification tasks with multiple choice answers (Li et al., 2024; Hu et al., 2024), $G(s_t, \theta)$ can vary continuously, so maximizing reward does not reduce to discovering the user's intent θ. $C(a_t^{\text{user}}, a_t^{\text{agent}})$ is a user-configurable *cost function* known to the agent, capturing the costs to the user of reading questions, writing answers, or receiving the wrong assistive action.

Gameplay. The user's intent $\theta \sim P_0(\theta)$ and initial world state $w_0 \sim P_0(w_0)$ are drawn from their priors. Starting from the empty conversation $h_0 = ()$, the user and agent take turns according to their policies $\pi^{\text{user}}(a_t^{\text{user}} | s_{t-1}, \theta)$ and $\pi^{\text{agent}}(a_t^{\text{agent}} | s_{t-1})$, with the user going first. The game ends when a player sends a terminal message $m_t \in \mathcal{M}^{\text{end}}$.

Termination Control. Prior work on clarification generally assumes that the interaction ends once the assistant takes an action m^{act} (e.g. answering the user's query) (Andukuri et al., 2024; Zhang et al., 2025). Our formulation instead allows termination to be controlled by *either* the assistant or the user. In one case, assistant actions m^{act} are terminal. In the other case, users can decide between providing corrections m^{cor} or ending the conversation with a message m^{end} (see 4.1), capturing a wide range of more natural interactions.

Modeling the User. To solve an assistance game from the agent's perspective, we fix a user policy $\pi^{\text{user}}(a_t^{\text{user}} \mid h_t, \theta)$. This reduces the game to an *assistive partially observable Markov decision process* (A-POMDP) (Hadfield-Menell et al., 2016; Laidlaw et al., 2025). Solving this POMDP amounts to optimal assistance with respect to the user model π^{user}.

For our assistant to be helpful to *actual* users, its user model needs to be reasonably realistic. We thus make the following assumptions suited for the conversational assistance context:

- The user always initiates the conversation with a request message $m_1 \in \mathcal{M}^{\text{req}}$.
- In response to a question $m_{t-1} \in \mathcal{M}^{\text{ask}}$, the user replies with an answer $m_t \in \mathcal{M}^{\text{ans}}$.
- When the user has termination control, after the assistant acts via $m_{t-1} \in \mathcal{M}^{\text{act}}$, the user can choose to issue a correction $m_t \in \mathcal{M}^{\text{cor}}$ or end the conversation $m_t \in \mathcal{M}^{\text{end}}$.

We instantiate this in our experiments by prompting an LLM to simulate a human user, providing it with the true or hypothesized user intent θ. For increased realism, we do not give the assistant a true model of the user; instead, it has an *internal* user model that differs from the *external* user simulator in our benchmark environments. In domains with external actions (e.g., ADAPT), we also assume that users delegate all such actions to the agent, leaving dual-control environments to future work (Barres et al., 2025).

3 Rational Enquiry via Value-of-Information Reasoning

In principle, solving the assistive POMDP in Section 2 yields optimal clarification behavior. However, exact POMDP planning is intractable, and existing solvers for POMDPs (Kurniawati et al., 2008; Silver & Veness, 2010; Somani et al., 2013) and assistance games (Malik et al., 2018; Laidlaw et al., 2025) cannot be applied to natural language interactions and goal spaces. We therefore propose **Rational Enquiry via Value-of-Information Reasoning** (**REVOIR**, Figure 1), an inference-time algorithm that provides rational assistance with limited lookahead. REVOIR is a hierarchical control policy that decides between (i) an *ask* sub-policy π^{ask} that produces a clarifying question $m^{\text{ask}} \in \mathcal{M}^{\text{ask}}$; (ii) an *act* sub-policy π^{act} that produces an assistive action $a^{\text{act}} \in \mathcal{A}^{\text{act}} = \mathcal{X} \cup \mathcal{M}^{\text{act}}$. At each turn t, REVOIR:

1. Updates its belief $b_t(\theta)$ over the user's intent given the conversational history h_t;
2. Evaluates the VoI of π^{ask} under belief b_t, and (if corrections are possible) the VoI of π^{act};
3. Decides between asking with π^{ask} vs. acting with π^{act} based on their cost-adjusted values.

We unpack each step of REVOIR below.

3.1 Belief Updating over User Intent

To estimate how much decision quality improves with new information, the agent needs to both update and simulate future changes in its beliefs about rewarding outcomes, which in turn depend on its belief b_t about the user's intent θ. This belief is given via Bayes rule as:

$$b_t(\theta) \propto b_0(\theta) \prod_{\tau=1}^{t} \pi^{\text{user}}(a_\tau^{\text{user}} \mid s_{\tau-1}, \theta) = b_{t-1}(\theta) \cdot \pi^{\text{user}}(a_t^{\text{user}} \mid s_{t-1}, \theta), \tag{1}$$

where $b_0(\theta) := P_0(\theta)$ is the agent's prior over intents, and π^{user} is the user model.

Representing and updating $b_t(\theta)$ is challenging in practice: The space of (language-represented) intents Θ is intractable to enumerate over, and we may not have trustworthy models of the prior $b_0(\theta)$ or robust estimates of the likelihood $\pi^{\text{user}}(a_t^{\text{user}} \mid s_{t-1}, \theta)$[1]. We address this by approximating b_t with a weighted particle set $\hat{b}_t = \{(\theta^n, w^n)\}_{n=1}^N$, where:

- Intent hypotheses θ^n are proposed by an LLM $Q_{\text{LLM}}(\theta | h_t, \hat{b}_{t-1})$ given the history h_t and past belief $\hat{b}_{t-1}$ (if available), with the option of retaining or pruning particles in $\hat{b}_{t-1}$;
- Each θ^n is assigned a weight $w^n = S_{\text{LLM}}(\theta^n, h_t)$ by an LLM that verbally scores the consistency of θ^n with h_t, serving as a "semantic" log-likelihood of conversation h_t.

This scheme can be viewed as a form of variational particle approximation (Saeedi et al., 2017; Afshar et al., 2024), which approximates the full posterior belief by weighting a small set of particles by their (log) probabilities. In our experiments, we explore several variants that improve performance, including incremental updating of w^n as each user message a_t^{user} is received, and factorizing θ into independently-updated components (see Appendix I).

3.2 Evaluating the Value of Information (VoI)

Assessing the value of a question involves determining how it will improve later decisions. When a question is answered, this updates the belief b and changes the expected reward of addressing the user's request with an action a^{act}. This reward is computed as follows.

[1] LLM log-probabilities can be used to score the prior and the likelihood, since θ and a^{user} are in language, but these are often sensitive to semantically-irrelevant features like typos or text length.

Expected Goal Reward. The expected goal reward of $a^{\text{act}} \in \mathcal{A}^{\text{act}}$ under belief b is:

$$\bar{G}(s, b, a^{\text{act}}) = \mathbb{E}_{\theta \sim b, s' \sim T(\cdot|s, a^{\text{act}})}\big[G(s', \theta)\big], \tag{2}$$

where s' is the state reached by executing a^{act} from s. In practice, the true goal reward G is not available to the agent. Instead we use an internal LLM-based reward model $\hat{G}_{\text{LLM}}(s', \theta)$ to score a^{act} against each intent hypothesis θ^k, and estimate $\bar{G}$ as the weighted average.

In REVOIR, we need to select between policy branches, not just actions. As such, we also need to compute the (cost-adjusted, immediate) reward of the *act* policy π^{act}:

Net Reward of Act Policy. The net immediate reward of an *act* policy $\pi^{\text{act}}(\cdot \mid s, b)$ is:

$$R^{\text{act}}(s, b) = \mathbb{E}_{a^{\text{act}} \sim \pi^{\text{act}}(\cdot|s,b)}\big[\bar{G}(s, b, a^{\text{act}}) - C(a^{\text{act}})\big] \tag{3}$$

For improved action selection, we use a best-of-K policy for π^{act}, proposing K action candidates $\{a^{\text{act},k}\}_{k=1}^{K} \sim \text{LLM}(\cdot \mid s, b)$ from an LLM and selecting the best. In this case, the net reward of π^{act} reduces to the maximum cost-adjusted reward over K proposed actions.

In some settings, the assistant's action is terminal and provides no information. However, when the user controls termination and may follow an action a^{act} with a correction $m^{\text{cor}} \in \mathcal{M}^{\text{cor}}$, acting itself can result in valuable information:

Value of Information for an Action. The (one-step) value of information of an action $a^{\text{act}} \in \mathcal{A}^{\text{act}}$ is the expected net reward of acting again after a user's correction m^{cor}:

$$\text{VoI}(s, b, a^{\text{act}}) = \mathbb{E}_{m^{\text{cor}}}\big[R^{\text{act}}(s', b')\big] \tag{4}$$

Here, $s' \sim T(\cdot \mid (h_t, a^{\text{act}}), m^{\text{cor}})$ and $b'(\theta) \propto b(\theta)\, \pi^{\text{user}}(m^{\text{cor}} \mid (h_t, a^{\text{act}}), \theta)$ are the updated state and belief after receiving m^{cor}. The expectation is taken by sampling a user intent $\theta \sim b(\cdot)$ from b, simulating the user's correction $m^{\text{cor}} \sim \pi^{\text{user}}(\cdot \mid (h_t, a^{\text{act}}), \theta)$ to the action a^{act}, then updating the state s' and belief b'. Note that m^{cor} can be a user message that accepts a^{act} and ends the conversation, s.t. $R^{\text{act}}(s', b') = 0$.

With this, we can define the *value function* (i.e. cumulative reward) for the *act* policy π^{act}, combining the net immediate reward with the expected future reward after a correction:

Value of Act Policy. The (one-step) cumulative reward of an *act* policy $\pi^{\text{act}}(\cdot \mid s, b)$ is:

$$V^{\text{act}}(s, b) = \mathbb{E}_{a^{\text{act}}, m^{\text{cor}}}\big[\text{VoI}(s, b, a^{\text{act}}) - C(m^{\text{cor}})\big] + R^{\text{act}}(s, b) \tag{5}$$

where $a^{\text{act}} \sim \pi^{\text{act}}(\cdot \mid s, b)$ and m^{cor} is the user's correction to a^{act}. Note that $R^{\text{act}}(s, b)$ is an expectation that accounts for the possibility that the user does not accept the action a^{act}, and that $V^{\text{act}}(s, b)$ reduces to $R^{\text{act}}(s, b)$ when assistant actions a^{act} are terminal.

Having defined the value of acting, we can now quantify the VoI gained from asking a question, revising beliefs based on the answer, and acting based on the revised beliefs.

Value of Information for a Question. The (one-step) value of information of a question $m^{\text{ask}} \in \mathcal{M}^{\text{ask}}$ is the expected cumulative reward of acting once the answer m^{ans} is received:

$$\text{VoI}(s, b, m^{\text{ask}}) = \mathbb{E}_{m^{\text{ans}}}\big[V^{\text{act}}(s', b')\big] \tag{6}$$

Similar to above, the expectation is taken by sampling an intent $\theta \sim b(\cdot)$ from b, simulating the user's answer $m^{\text{ans}} \sim \pi^{\text{user}}(\cdot \mid (h_t, m^{\text{ask}}), \theta)$ to the question m^{ask}, then updating both s' and b'. In Appendix A, we show that as long as we maximize over enough samples when acting, the VoI of a question is never worse than the net reward of acting immediately.

Finally, we compute the value function of the *ask* policy π^{ask} as its (cost-adjusted) expected future reward, assuming that the agent acts immediately after receiving an answer:

Value of Ask Policy. The (one-step) cumulative reward of an *ask* policy $\pi^{\text{ask}}(\cdot \mid s, b)$ is:

$$V^{\text{ask}}(s, b) = \mathbb{E}_{m^{\text{ask}}, m^{\text{ans}}}\big[\text{VoI}(s, b, m^{\text{ask}}) - C(m^{\text{ans}}) - C(m^{\text{ask}})\big] \tag{7}$$

where $m^{\text{ask}} \sim \pi^{\text{ask}}(\cdot \mid s, b)$ and m^{ans} is the user's simulated answer to m^{ask}. For improved performance, we use a best-of-K policy for π^{ask}, maximizing over K questions sampled from an LLM prompted with a summary of the current belief b.

3.3 Deciding Between Asking and Acting

With the value functions for π^{ask} and π^{act} defined, REVOIR asks questions if and only if V^{ask} (the cost-adjusted VoI of asking) is higher than the value V^{act} of acting immediately:

$$\pi^{\text{REVOIR}}(\cdot \mid s, b) = \begin{cases} \pi^{\text{ask}}(\cdot \mid s, b), & V^{\text{ask}}(s, b) > V^{\text{act}}(s, b), \\ \pi^{\text{act}}(\cdot \mid s, b), & \text{otherwise.} \end{cases} \tag{8}$$

We note that in practice, $V^{\text{ask}}(s, b)$ and $V^{\text{act}}(s, b)$ are expectations that we can only effectively estimate via sampling. With best-of-K policies, this amounts to estimating $V^{\text{ask}}(s, b)$ and $V^{\text{act}}(s, b)$ with the value of the best question m^{ask} or action m^{act} among the K samples.

4 Experiments

4.1 Ambiguous Question Answering (CondAmbigQA)

Our first domain is ambiguous question-answering. We use CondAmbigQA (Li et al., 2025), a dataset of 2,000 ambiguous queries drawn from AmbigNQ (Min et al., 2020). Every query in the dataset is paired with a set of *conditions*, each representing a possible latent intent $\theta \in \Theta$ behind the ambiguous query. The assistant's goal is to produce a final answer m^{act} that is semantically similar to the ground-truth answer y_θ to the user's true intent θ.

Benchmark Configuration. Assistants interact with an LLM-based user simulator π^{user} which provides clarifications or accepts/corrects answers based on the user's true intent θ. Following Li et al. (2025), we evaluate the assistant's final answer m^{act} against the ground-truth answer y_θ using a rubric-based `AnswerCorrectness`$(m^{\text{act}}, y_\theta)$ metric with the G-Eval harness (Liu et al., 2023). This metric lies in $[0, 1]$, and uses an LLM judge (GPT-4o-mini) to inspect m^{act} for contradictions, omissions, relevance, and depth. We also record the number of clarifications (questions + corrections) as a measure of clarification `Effort`. We validate both the user simulator and `AnswerCorrectness` metric with human annotators (Appendix E), finding high levels of agreement w.r.t. answer acceptance (72 out of 89) and answer quality (41 out of 57). We also test for robustness against an "inattentive" user simulator in Appendix D.3. Further details and validations are in Appendix C.

Termination Variants. We instantiate two benchmark variants: With *agent termination*, the assistant's m^{act} is terminal. With *user termination*, the user corrects m^{act} if it fails to address their intent θ; otherwise the user accepts the answer, ending the conversation.

REVOIR. REVOIR optimizes a proxy goal reward $\hat{G}_{\text{LLM}}(m^{\text{act}}, \hat{\theta}) \in [0, 1]$ which evaluates m^{act} against a hypothesized intent $\hat{\theta}$ similarly to `AnswerCorrectness`, but *without* knowing the true answer y_θ or using the G-Eval harness. We use a word-budgeted cost function $C(m^{\text{user}}, m^{\text{agent}})$ which assigns a cost of 0 to a (user/agent) message m when its word count $|m|$ is less than the respective budget $B_{\text{user/agent}}$, then scales linearly to 1 as $|m|$ grows to twice the budget. The internal user model (distinct from the benchmark's user simulator), proxy goal reward, hypothesis proposer (N=5), hypothesis scorer, question generator (K=5) and answer generator (K=1) all use the same LLM backbone. See Appendix C.2 for details.

Baselines. We compare REVOIR against: **REIGN** (Rational Enquiry via Information-Gain Net-utility): A REVOIR variant configured to maximize expected information gain (EIG) (Lindley, 1956; MacKay, 1992) instead of VoI (see Appendix B), balanced against message costs; **Direct Answer**: zero-shot, no clarification; **ReAct** (Yao et al., 2022): chain-of-thought interleaved with actions, no explicit belief or cost accounting; **SC-BoN-ReAct**: generates K ReAct samples, decides to ask or act via majority vote (a.k.a. self-consistency, Wang et al. (2023)), then selects the best of the $N \leq K$ majority samples with an LLM judge similar to REVOIR's proxy goal reward $\hat{G}_{\text{LLM}}$; **Entropy Thresholding**: same particle belief as REVOIR, but asks until $\frac{H_{\max}(b) - H(b)}{H_{\max}(b)} > \tau$, with τ fitted; **ReflectionDPO-ReAct**: our adaptation of ReflectionDPO (Patel et al., 2025) (Appendix H), which finetunes a model to ask clarifying questions in order to better mimic an oracle; and **Oracle-ReAct**: ReAct privileged with ground-truth y_θ, serving as a non-interactive oracle / topline.

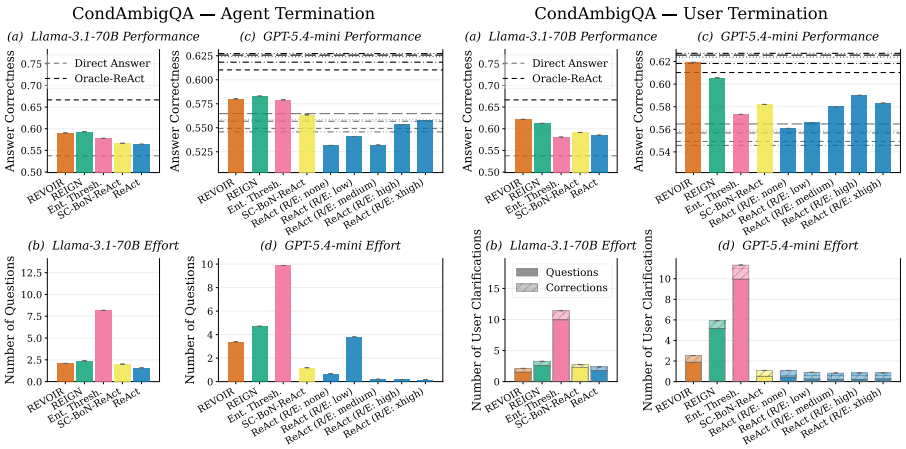


Figure 2: CondAmbigQA results for Llama-3.1-70B and GPT-5.4-mini. ReAct variants and non-interactive base/top lines are evaluated at five reasoning-effort (R/E) levels for GPT-5.4-mini. Horizontal lines correspond to baselines and toplines across R/E levels: gray lines represent the **Direct Answer** baselines, and **black lines** represent **Oracle-ReAct** toplines, with line styles indicating effort levels: (– · · –) for `none`, (— —) for `low`, (– · –) for `medium`, (· · ·) for `high`, and (– –) for `xhigh`. REVOIR and REIGN use budgets of $(B_{\text{agent}}, B_{\text{user}}) = (100, 50)$, and SC-BoN-ReAct uses $K = 5$ samples.

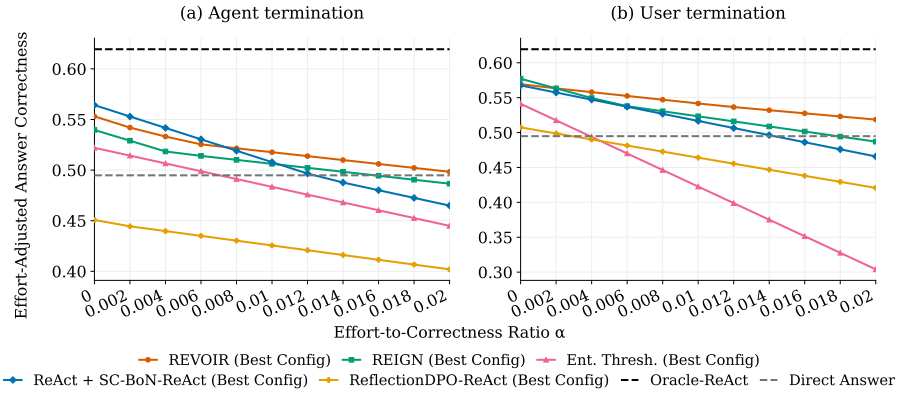


Figure 3: Effort-adjusted answer correctness on CondAmbigQA for Llama-3.1-8B as a function of the effort-to-correctness ratio α. The y-axis plots `AnswerCorrectness` $-$ $\alpha \cdot$ `Effort` for the best method configuration at each value of α. `Effort` is the number of clarifications per conversation. REVOIR dominates other methods for most values of α under both (a) agent termination ($\alpha > 0.008$) and (b) user termination ($\alpha > 0.002$).

4.1.1 Experiment Results

REVOIR achieves high correctness while minimizing clarification effort. Figure 2 compares REVOIR against baselines using Llama-3.1-70B, and GPT-5.4-mini as backbones. Under *agent termination*, REVOIR beats all ReAct variants in correctness. REIGN and Entropy Thresholding are close in correctness, with slight edge for REIGN, but at the cost of significantly more questions than REVOIR (1.3–2.8× for GPT-5.4-mini). REVOIR's advantage is even sharper under *user termination*, outperforming all baselines in correctness while using the least clarifications of non-ReAct methods. ReAct variants clarify very little in all cases, but at a substantial cost to correctness. We find similar results for Gemini 3.5 Flash-Lite and Claude Haiku 4.5 in Appendix D.2, albeit with model-specific idiosyncrasies. Appendix D.3 also shows REVOIR's robustness to having an inaccurate user model.

REVOIR best trades-off correctness vs. clarification effort for most trade-off ratios. To study how well REVOIR adapts to different preferences about correctness vs. clarification effort, we run REVOIR across many budget points $(B_{\text{agent}}, B_{\text{user}})$, using a smaller Llama-

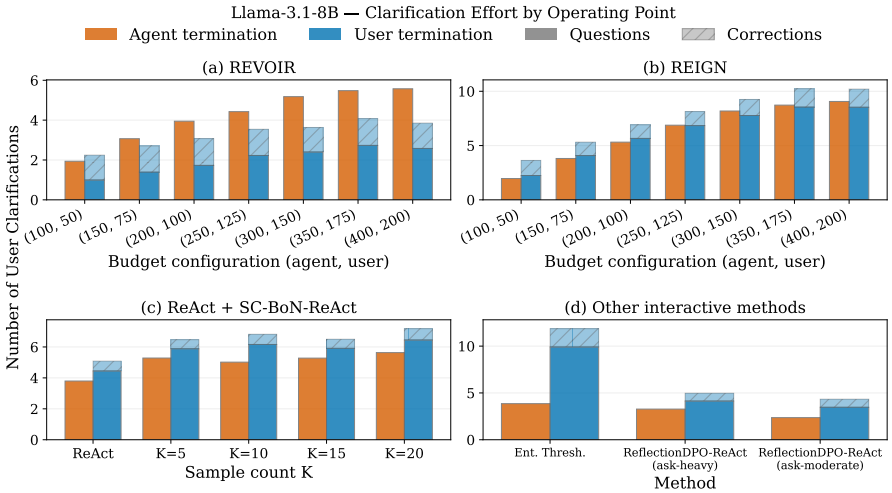


Figure 4: Clarification effort on CondAmbigQA for Llama-3.1-8B across termination variants and methods. When moving from *agent termination* to *user termination*, only REVOIR adapts to using less clarifications in total by relying on user corrections.

3.1-8B model due to compute costs. We then compute the *effort-adjusted answer correctness* $=$ `AnswerCorrectness` $-\ \alpha \cdot$ `Effort` as a measure of overall utility, where α is the effort-to-correctness trade-off ratio, and find the REVOIR budget that maximizes this metric at each value of α. Figure 3 plots this metric across a range of trade-off ratios, comparing REVOIR at each α against the best configuration for each baseline (across REIGN budgets, SC-BoN-ReAct sample sizes $K \in \{1, 5, 10, 15, 20\}$, ReflectionDPO data mixtures; see Tables D.1–D.2). REVOIR outperforms baselines across most trade-off ratios α (agent term.: $\alpha > 0.008$, user term.: $\alpha > 0.002$), and is second best even at $\alpha = 0$ (i.e. cost-free clarifications), demonstrating that REVOIR is the most adaptable of the methods to varying trade-offs.

REVOIR rationally adapts to the provision of user corrections. Figure 4 shows the clarification effort for each method on Llama-3.1-8B across termination variants. Only REVOIR adapts rationally when moving from *agent termination* to *user termination*; since user corrections provide information, REVOIR not only asks less questions, but makes less clarifications overall. All other methods increase in both questions and clarifications.

Generic inference-time scaling does not consistently improve performance or rational information-gathering. In Figure 2, increasing GPT-5.4-mini ReAct's reasoning effort yields non-monotonic and limited gains. Notably, higher reasoning effort (above `low`) results in very few clarifications. We find similar trends for other reasoning models (Section D.2), indicating that reasoning models do not spend extra reasoning tokens on whether more clarification is helpful to answer the user's query; instead tokens appear to be spent on directly formulating a better answer. We separately find that SC-BoN-ReAct does not consistently improve with more samples K, and only exceeds medium-to-high budget REVOIR by using up to 2–4× more tokens (Tables D.3 & D.1). These results suggest that neither reasoning effort nor parallel sampling substitutes for principled VoI reasoning.

4.2 Preference-Aligned Household Assistance (ADAPT)

Our second domain is preference-aligned household task planning. We build on the ADAPT benchmark (Patel et al., 2025), where an agent must complete long-horizon cooking tasks (e.g., "prepare an omelet for breakfast") while respecting a user's hidden preference set $\theta = \{\gamma_1, \ldots, \gamma_M\}$, with preferences like "use non-dairy milk" or "serve beverages first."

Benchmark Description. The agent operates in a grounded text-based simulation with over 230 objects. At each turn t it may take an external action $a_t \in \mathcal{X}$ (e.g. physical manipulation) or send a question m^{ask} to the user. The user simulator $\pi^{\text{user}}(\cdot \mid h_t, \theta)$ answers questions based on the true preferences θ, following the original benchmark exactly. Episodes are scored by the *preference satisfaction rate* $\text{PSR} = p^+/(p^+ + p^-) \in [0, 1]$, where p^+ and p^- count the preferences in θ that are satisfied or not in the final state s_t.



Figure 5: ADAPT preference satisfaction rate and question count across methods (4-fold cross-validated following ADAPT splits in Patel et al. 2025). The standard deviation is computed across split means. Baseline results are reproduced from Table 1 in Patel et al. (2025). For ICL variants, all seen personas are provided in context during belief updating. More details are in Table I.1.

Methods & Baselines. REVOIR is evaluated on a two-stage variant of ADAPT where all questions are asked before committing to a physical plan (Appendix I.1); this disadvantages it relative to the baselines, which interleave asking and acting. REVOIR uses PSR as the goal reward $G(s, \hat{\theta}) = \text{PSR} \cdot \mathbf{1}[s \in \mathcal{S}^{\text{term}}]$, where $\hat{\theta}$ is inferred. To match Patel et al. (2025), we also use Llama-3.1-70B-Instruct as our LLM, assume agent termination, and use a very low question cost $C(m^{\text{ask}}) = 5 \times 10^{-4}$ since the original baselines are not cost-aware.

We compare against finetuning methods (ReflectionDPO (Patel et al., 2025); STaR-GATE (Andukuri et al., 2024)), prompting baselines (Never-Ask; Baseline [≡ Direct Answer]; ReAct), a Teacher oracle with the true θ, and an Always-Ask baseline. While REVOIR requires no finetuning on user personas θ (unlike ReflectionDPO), we still evaluate on the benchmark's *seen* and held-out (*unseen*) persona splits, and test both a *few-shot* variant (seen personas are in-context examples for belief updating) and a *zero-shot* variant (no examples). Full details, along with belief-update variants, can be found in Appendices I–K.

4.2.1 Experiment Results

Figure 5 shows REVOIR with the best belief updating configuration (factored beliefs + incremental updating). In zero-shot, REVOIR achieves 51.5% PSR on seen personas and 49.5% PSR on unseen personas, virtually matching Always-Ask with 11× fewer questions. With ICL, REVOIR reaches 58.8% on seen personas and 56.0% on unseen personas, exceeding Always-Ask by 6 points and outperforming the fully-trained Reflection-DPO (44%) by 13–15 points with 5× fewer questions. Overall, REVOIR outperforms all finetuning and prompting baselines. More results are in Appendix I, showing that factored belief updating helps substantially, but that joint beliefs still do well with few-shot examples.

5 Conclusion

In this paper, we introduced REVOIR, an inference-time algorithm for rational clarification via VoI reasoning. Across ambiguous question-answering and preference-aligned task planning, REVOIR achieves a better accuracy-efficiency frontier than alternatives, adapts naturally to user-correctable settings, and surpasses strong fine-tuning baselines while asking substantially fewer questions. However, several limitations point towards future work: REVOIR's one-step VoI estimate is tractable but myopic, suggesting the need to develop efficient multi-step estimates of VoI. Second, REVOIR is currently not designed for dual control or full interleaving of questions and non-terminal actions; addressing this will require novel ways to efficiently estimate VoI under interleaving. Finally, while REVOIR is reasonably robust to imperfect user modeling (Appendix D.3), evaluating with real users and their idiosyncrasies is an important next step toward deployment.

AI USE STATEMENT

In this work, we used generative AI tools for augmenting a dataset (Section C.1), implementing parts of the presented methods, and implementing parts of the result visualization and interpretation. We have not used generative AI tools for generating synthetic data sets, helping develop theoretical models or conceptual frameworks, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, assisting with translation, or cleaning datasets. Formulating mathematical claims played a minimal role in this work, and AI assistance was used only for extending known results with elementary techniques, formatting, and restatement. Additionally, we used generative AI tools for writing suggestions in parts of the manuscript. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Hadi Mohasel Afshar, Gilad Francis, and Sally Cripps. Optimal particle-based approximation of discrete distributions (opad). *arXiv preprint arXiv:2412.00545*, 2024.

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

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Yizhou Chi, Jessy Lin, Kevin Lin, and Dan Klein. Clarinet: Augmenting language models to ask clarification questions for retrieval. *arXiv preprint arXiv:2405.15784*, 2024.

Yang Deng, Lizi Liao, Liang Chen, Hongru Wang, Wenqiang Lei, and Tat-Seng Chua. Prompting and evaluating large language models for proactive dialogues: Clarification, target-guided, and non-collaboration. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10602–10621, 2023.

Gabriel Grand, Valerio Pepe, Jacob Andreas, and Joshua B. Tenenbaum. Shoot First, Ask Questions Later? Building Rational Agents that Explore and Act Like People, October 2025. URL http://arxiv.org/abs/2510.20886. arXiv:2510.20886 [cs].

Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. Cooperative Inverse Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://papers.nips.cc/paper_files/paper/2016/hash/c3395dd46c34fa7fd8d729d8cf88b7a8-Abstract.html.

Kunal Handa, Yarin Gal, Ellie Pavlick, Noah Goodman, Jacob Andreas, Alex Tamkin, and Belinda Z Li. Bayesian preference elicitation with language models. *arXiv preprint arXiv:2403.05534*, 2024.

Jiwoo Hong, Noah Lee, and James Thorne. Orpo: Monolithic preference optimization without reference model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 11170–11189, 2024.

Ronald A. Howard. Information Value Theory. *IEEE Transactions on Systems Science and Cybernetics*, 2(1):22–26, August 1966. ISSN 2168-2887. doi: 10.1109/TSSC.1966.300074. URL https://ieeexplore.ieee.org/abstract/document/4082064.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Zhiyuan Hu, Chumin Liu, Xidong Feng, Yilun Zhao, See-Kiong Ng, Anh Tuan Luu, Junxian He, Pang Wei Koh, and Bryan Hooi. Uncertainty of Thoughts: Uncertainty-Aware Planning Enhances Information Seeking in LLMs. November 2024. URL https://openreview.net/forum?id=CVpuVe1N22&noteId=aTI8PGpO47.

Hanna Kurniawati, David Hsu, Wee Sun Lee, et al. SARSOP: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems*, volume 2008. Zurich, Switzerland, 2008.

Cassidy Laidlaw, Eli Bronstein, Timothy Guo, Dylan Feng, Lukas Berglund, Justin Svegliato, Stuart Russell, and Anca Dragan. AssistanceZero: Scalably solving assistance games. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=b9hVMJi0t2.

Shuyue Stella Li, Vidhisha Balachandran, Shangbin Feng, Jonathan Ilgen, Emma Pierson, Pang Wei Koh, and Yulia Tsvetkov. MEDIQ: Question-Asking LLMs for Adaptive and Reliable Clinical Reasoning, June 2024. URL http://arxiv.org/abs/2406.00922. arXiv:2406.00922 [cs].

Zongxi Li, Yang Li, Haoran Xie, and S Joe Qin. CondAmbigQA: A benchmark and dataset for conditional ambiguous question answering. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 2269–2288, 2025.

D. V. Lindley. On a Measure of the Information Provided by an Experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005, December 1956. ISSN 0003-4851, 2168-8990. doi: 10.1214/aoms/1177728069. URL https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-27/issue-4/On-a-Measure-of-the-Information-Provided-by-an-Experiment/10.1214/aoms/1177728069.full.

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

Rachel Ma, Jingyi Qu, Andreea Bobu, and Dylan Hadfield-Menell. Flexible Agent Alignment with Goal Inference from Open-Ended Dialog. *arXiv preprint arXiv:2508.15119*, 2025.

David JC MacKay. Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604, 1992.

Dhruv Malik, Malayandi Palaniappan, Jaime Fisac, Dylan Hadfield-Menell, Stuart Russell, and Anca Dragan. An Efficient, Generalized Bellman Update For Cooperative Inverse Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 3394–3402. PMLR, July 2018. URL https://proceedings.mlr.press/v80/malik18a.html.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. AmbigQA: Answering Ambiguous Open-domain Questions. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5783–5797, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.466. URL https://aclanthology.org/2020.emnlp-main.466.

Lee Chong Ming. Replit's CEO apologizes after its AI agent wiped a company's code base in a test run and lied about it. *Business Insider*, 2025.

Maithili Patel, Xavier Puig, Ruta Desai, Roozbeh Mottaghi, Sonia Chernova, Joanne Truong, and Akshara Rai. ADAPT: Actively Discovering and Adapting to Preferences for any Task, April 2025. URL http://arxiv.org/abs/2504.04040. arXiv:2504.04040 [cs].

Top Piriyakulkij, Volodymyr Kuleshov, and Kevin Ellis. Asking clarifying questions using language models and probabilistic reasoning. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023. URL https://openreview.net/forum?id=2SjoG6lVz3.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. November 2023. URL https://openreview.net/forum?id=HPuSIXJaa9.

Howard Raiffa and Robert Schlaifer. *Applied Statistical Decision Theory*. Division of Research, Harvard Business School, Boston, MA, 1961.

Tom Rainforth, Adam Foster, Desi R. Ivanova, and Freddie Bickford Smith. Modern Bayesian Experimental Design. *Statistical Science*, 39(1), February 2024. ISSN 0883-4237. doi: 10.1214/23-STS915. URL https://projecteuclid.org/journals/statistical-science/volume-39/issue-1/Modern-Bayesian-Experimental-Design/10.1214/23-STS915.full.

Ardavan Saeedi, Tejas D Kulkarni, Vikash K Mansinghka, and Samuel J Gershman. Variational particle approximations. *Journal of Machine Learning Research*, 18(69):1–29, 2017.

David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. *Advances in Neural Information Processing Systems*, 23, 2010.

Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. DESPOT: Online POMDP Planning with Regularization. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/hash/c2aee86157b4a40b78132f1e71a9e6f1-Abstract.html.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*, February 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, December 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Shirley Wu, Michel Galley, Baolin Peng, Hao Cheng, Gavin Li, Yao Dou, Weixin Cai, James Zou, Jure Leskovec, and Jianfeng Gao. CollabLLM: From passive responders to active collaborators. In *International Conference on Machine Learning*, pp. 67260–67283. PMLR, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022. URL https://openreview.net/forum?id=tvI4u1ylcqs.

Michael Zhang, W Bradley Knox, and Eunsol Choi. Modeling Future Conversation Turns to Teach LLMs to Ask Clarifying Questions. In *International Conference on Learning Representations*, volume 2025, pp. 60722–60742, 2025.

Tan Zhi-Xuan, Lance Ying, Vikash Mansinghka, and Joshua B. Tenenbaum. Pragmatic Instruction Following and Goal Assistance via Cooperative Language-Guided Inverse Planning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '24, pp. 2094–2103, Richland, SC, May 2024. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 979-8-4007-0486-4.

APPENDIX

A Value of Adaptivity

Here, we develop on a well-known result about why acting after gaining new information leads to better outcomes (Howard, 1966). By employing a best-of-K approach for the π^{act} sub-policy, $R^{\text{act}}(s', b')$ effectively maximizes the cost-adjusted reward over candidate actions conditioned on the updated belief b'. Because the maximum of a set of linear functions is convex, Jensen's inequality guarantees that the expected value of this maximum across all possible user answers is greater than or equal to the maximum reward obtainable under the current belief, provided the same candidate set is available before and after clarification.

For the following results, we consider terminal act decisions and questions that change only information, leaving the feasible actions and the utility of each fixed action unchanged.

Proposition A.1 (Value of Adaptivity)**.** *For any fixed nonempty finite action set $\mathcal{A}$,*

$$\mathbb{E}_{m^{\text{ans}}}\left[\max_{a^{\text{act}}\in\mathcal{A}}\left(\bar{G}(s,b',a^{\text{act}})-C(a^{\text{act}})\right)\right]\geq\max_{a^{\text{act}}\in\mathcal{A}}\left(\bar{G}(s,b,a^{\text{act}})-C(a^{\text{act}})\right). \tag{9}$$

Proof sketch. Under Bayesian updating, with answers drawn from the corresponding predictive distribution, $\mathbb{E}_{m^{\text{ans}}}[b'] = b$ by the law of total expectation. For each fixed action, $\bar{G}(s, b, a^{\text{act}}) - C(a^{\text{act}})$ is linear in b, so its maximum over $\mathcal{A}$ is convex. Jensen's inequality gives the result. The inequality can be strict because the maximizing action may differ across the realized answers. □

Thus, for a policy that maximizes over this set, $\text{VoI}(s, b, m^{\text{ask}}) - R^{\text{act}}(s, b)$ (and similarly, $\text{VoI}(s, b, m^{\text{act}}) - R^{\text{act}}(s', b')$) is non-negative, because more information affords the agent the opportunity to pivot to a higher-reward action once its uncertainty is reduced.

When π^{act} does not adapt to the updated belief b' (i.e., $\pi^{\text{act}}(\cdot \mid b) = \pi^{\text{act}}(\cdot \mid b')$), the expected future goal reward of a fixed action, $\mathbb{E}_{m^{\text{ans}}}[\bar{G}(s, b', a^{\text{act}})]$, collapses back to the current expected reward $\bar{G}(s, b, a^{\text{act}})$ by the law of total expectation ($\mathbb{E}_{m^{\text{ans}}}[b'] = b$). Gathering more information thus provides no benefit to non-adaptive policies.

In practice, REVOIR generates candidates conditioned on its current belief b from some policy $\pi_0(a^{\text{act}} \mid b)$, so their distribution can change after the belief updates to b'. In such cases, under mild assumptions about $\pi_0(a^{\text{act}} \mid b')$, best-of-$K$ sampling after clarification can outperform even exact maximization before clarification when the expected benefit of the answer m^{ans} exceeds the sampling error.

Proposition A.2 (Value of Adaptivity with Belief-Conditioned Candidates)**.** *Let $R^{\text{act}}_*(s, b)$ be the supremum of $\bar{G}(s, b, a^{\text{act}}) - C(a^{\text{act}})$ over feasible actions. Suppose net rewards lie in a common interval of width D. For almost every answer, assume each of K independent draws from the updated proposal $\pi_0(\cdot \mid b')$ lies within $\varepsilon \in [0, D]$ of the optimal value under b' with probability at least $1 - \delta$, where $\delta \in [0, 1)$. Then the best-of-K act policy satisfies*

$$\mathbb{E}_{m^{\text{ans}}}[R^{\text{act}}(s', b')] \geq \mathbb{E}_{m^{\text{ans}}}[R^{\text{act}}_*(s', b')] - \varepsilon - (D - \varepsilon)\delta^K. \tag{10}$$

Hence, if the expected gain in optimal reward $\mathbb{E}[R^{\text{act}}_*(s', b')] - R^{\text{act}}_*(s, b)$ exceeds $\varepsilon + (D-\varepsilon)\delta^K$, then

$$\mathbb{E}_{m^{\text{ans}}}[R^{\text{act}}(s', b')] > R^{\text{act}}_*(s, b) \geq R^{\text{act}}(s, b). \tag{11}$$

In particular, $\delta^K \to 0$ as $K \to \infty$, so this inequality holds for sufficiently large K whenever the expected gain in optimal reward exceeds ε (i.e. whenever Eq. 9 is strict with an ε gap). For $K = 1$, the expected gain needs to exceed $\varepsilon + (D - \varepsilon)\delta$.

Proof sketch. Condition on the user's answer. The probability that all K candidates miss the ε-optimal set is at most δ^K. The selected action's shortfall from optimal is at most ε otherwise and at most D on a miss. Its expected shortfall is therefore at most $\varepsilon + (D - \varepsilon)\delta^K$. Averaging over answers gives Inequality (10) and comparisons complete the proof. □

B Recovering Expected Information Gain

REVOIR can be modified to follow an EIG maximization strategy as follows: If the goal reward in Section 3 is replaced with the log belief-probability $G(s, \theta) = \log b(\theta)$, then the expected goal reward $\bar{G}(s, b, a^{\text{act}})$ at belief b is just the (negative) entropy $-H(b)$, and $\text{VoI}(s, b, m^{\text{ask}})$ reduces to the (negative) expected entropy. If costs are also zero and acting gives no information, then the value difference between asking and acting is precisely the expected information gain $H(b) - H(b \mid m^{\text{ask}})$ from asking. EIG-based clarification can thus be viewed as a special case of REVOIR in which task reward is replaced with pure uncertainty reduction.

C Extended Details on CondAmbigQA

C.1 Benchmark Configuration

Dataset. CondAmbigQA (Li et al., 2025) contains 2,000 ambiguous queries drawn from AmbigNQ (Min et al., 2020), where each question is endowed with a set of explicit contextual constraints (called *conditions*) that approximate the assumptions underlying different valid interpretations (information-seeking intents) of the query. Each condition corresponds to an intent $\theta \in \Theta$ in our framework, and is supported by a set of retrieved Wikipedia fragments that serve as its source. Ground-truth answers $y \in \mathcal{Y}$ are extracted from Wikipedia fragments, and were generated via a human-LLM collaborative annotation process.

We augment each (question, condition) pair with an *explicit disambiguating question*—a question whose answer uniquely identifies the scoping condition θ. These are generated by GPT-4o-2024-08-06 prompted with the ambiguous question and the condition text. The explicit question serves two purposes: it is provided to the user simulator as a reference for what information should be revealed when answering clarification requests, and it serves as the oracle reflection question for our ReflectionDPO baseline (Appendix H). The augmented dataset is partitioned at the question level into train, validation, and test splits to ensure that no condition of a question seen during training appears in evaluation; split statistics are reported in Table C.1.

Table C.1: CondAmbigQA Data Splits

Split	Examples	Unique Questions
Train	3,062	1,600
Dev	380	200
Test	380	200

User Simulator. We simulate the user by prompting an LLM (GPT-4o-mini) with the ground-truth condition θ. At each turn, the user either responds to the assistant's clarification request m^{ask} based on the condition, or produces a dismissal if the inquiry cannot be answered (Templates 1–2). In the user termination setting, the user simulator also decides whether to accept the assistant's answer m^{act} (Templates 3–4) or respond with a correction m^{cor} (Templates 5–6). Note that REVOIR has *no access* to the true user simulator π^{user}, and instead uses its own proxy user model $\hat{\pi}^{\text{user}}$ with a different LLM backbone. We validate the user simulator against human annotators in Appendix E in terms of answer acceptance. We also test for robustness against an "inattentive" user simulator in Appendix F.2.

Evaluation Metric. We evaluate the quality of the assistant's final answer m^{act} against the ground-truth answer y_θ for the true intent θ using an LLM-based judge (Template 7). Following Li et al. (2025), we use a rubric-based `AnswerCorrectness` metric implemented in the G-Eval framework (Liu et al., 2023). The rubric performs a *contradiction check*, inspecting whether the answer contradicts established facts in y_θ; an *omission penalty*, heavily penalizing omission of critical details in y_θ; and a *relevance check*, ensuring the answer directly and concisely addresses the question. Since our assistants are not provided with access to web-retrieved resources for the user's query (unlike in Li et al. (2025)), we add an *acumen reward* that rewards relevant answers presented with more depth and detail than y_θ, thereby avoiding over-penalization of answers of higher quality than the reference answer.

We verify that our `AnswerCorrectness` metric is discriminative and rank-consistent by comparing it to topline and baseline anchors: **Oracle**, which submits the gold answers y_θ verbatim, scores 0.995 out of 1; **Oracle-ReAct**, an agent shown y_θ but answering in its

own words, scores 0.619 for Llama-3.1-8B and 0.63 for GPT-5.4-mini). **Direct Answer**, a non-interactive baseline that answers the user's query without reasoning, achieves around 0.50–0.56. Our metric thus orders the tiers as expected, and our interactive methods score between these topline and baseline anchors.

C.2 REVOIR Implementation in CondAmbigQA

Goal Reward. The proxy goal reward $\hat{G}_{\text{LLM}}(m^{\text{act}}, \theta^n)$ scores an answer m^{act} against each hypothesis θ^n using criteria analogous to the G-Eval `AnswerCorrectness` metric used by the benchmark, but omitting the contradiction check since no ground-truth answer y_θ is available at planning time (Templates 8–9):

1. Check whether the predicted answer addresses the asking intent implied by θ^n.
2. Heavily penalize omission of information-seeking targets implied by the asking intent.
3. Ensure the answer addresses the intended question θ^n without irrelevant information.

The expected reward is then $\bar{G}(s, \hat{b}_t, m^{\text{act}}) = \sum_n w^n \cdot \hat{G}_{\text{LLM}}(m^{\text{act}}, \theta^n)$.

Cost Function. We assume that users specify budgets B_{user} and B_{agent} for how many words they are willing to (respectively) write and read per message:

$$C(a_t^{\text{user}}, a_t^{\text{agent}}) = \tfrac{1}{B_{\text{user}}} \cdot \left[|a_t^{\text{user}}| - B_{\text{user}}\right]_+ + \tfrac{1}{B_{\text{agent}}} \cdot \left[|a_t^{\text{agent}}| - B_{\text{agent}}\right]_+,$$

where $|m|$ is the word count of m and $[x]_+ = \max(x, 0)$. A message m has zero cost when $|m|$ is less than the respective budget, and costs within $[0, 1]$ while less than twice the word budget. We assume costs are mostly incurred by the *user*, and that token inference costs are marginal, in line with LLM product trends where increasingly many tokens are spent to deliver user value.

Belief Updating. At each turn, REVOIR maintains a particle belief $\hat{b}_t = \{(\theta^n, w^n)\}_{n=1}^N$ over latent intents θ^n behind the ambiguous query as follows: $N = 5$ intent hypotheses are sampled from a hypothesis proposer LLM $Q_{\text{LLM}}(\theta | h_t, \hat{b}_{t-1})$ provided with the history h_t and past belief $\hat{b}_{t-1}$ (if available). This LLM is prompted to enumerate N distinct hypotheses θ^n (Templates 10–11), and may keep, or prune hypotheses present in the past belief $\hat{b}_{t-1}$.

An LLM-based hypothesis scorer S_{LLM} then computes a weight $w_t^n = S_{\text{LLM}}(\theta^n, h_t)$ for each hypothesis θ^n, reflecting how consistent θ^n is with the history h_t. The LLM is prompted to output a score between 0 and 10 (Templates 12–13), and this is then normalized to lie within $[0, 1]$ s.t. $\sum_{n=1}^N w^n = 1$. Weights are recomputed from scratch at every step t (batch rescoring), rather than being incrementally updated.

Internal User Model. REVOIR uses an internal user model $\hat{\pi}^{\text{user}}(\cdot | h_t, \theta^n)$ to forecast how a user will respond to the assistant given a hypothesized intent θ^n. We use different prompts for user answers m^{ans} to clarifying questions m^{act} (Templates 14–15) and user corrections m^{cor} to assistant actions m^{act} (Templates 5–6). $\hat{\pi}^{\text{user}}$ differs from the true user simulator π^{user} in several ways: (i) $\hat{\pi}^{\text{user}}$ has no access to the true intent θ; (ii) $\hat{\pi}^{\text{user}}$ uses a different LLM (e.g. Llama-3.1-8B) than π^{user} (GPT-4o-mini); (iii) different prompts are used. In Section D.3, we test REVOIR's robustness to user model misspecification by prompting the external simulator π^{user} to mimic an inattentive user while keeping $\hat{\pi}^{\text{user}}$ the same.

Question Generation and Selection. We use a best-of-K *ask* policy π^{ask} with $K = 5$ question candidates, which are proposed from a *question generator* $\pi_0^{\text{ask}}(\cdot | \hat{b}_t)$ prompted with the particle belief $\hat{b}_t$ (Templates 16–17). We then select among these K questions by maximizing the cost-adjusted VoI (the expectand in Eq. 7). The VoI of each question m^{ask} is computed per Eq. 6 by simulating 1 user answer m^{ans} per hypothesis θ^n in the belief $\hat{b}_t$, and taking the sample average of the expectand.

Answer Generation and Evaluation. For efficiency, the *act* policy π^{act} samples only $K = 1$ answer candidate m^{act} from an *answer generator* $\pi_0^{\text{act}}(\cdot | \hat{b}_t) \equiv \pi^{\text{act}}(\cdot | \hat{b}_t)$ prompted with the particle belief $\hat{b}_t$ (Templates 18–19). Despite not maximizing over many candidates, we note that this policy is still *adaptive* in the sense of Appendix A to the extent that the

LLM generates better answers when it is prompted with an updated belief b'. Proposition A.2 provides a sufficient condition for belief-adaptivity that results in positive value of information, including when $K = 1$. Thus, explicit selection among multiple candidates is not necessary for clarification to improve downstream decisions.

With one answer candidate m^{act}, the value of acting V^{act} (Eq. 5) is estimated by the expected cumulative reward (or Q-value) of m^{act}:

$$\begin{aligned} Q^{\text{act}}(s, b, m^{\text{act}}) &= R^{\text{act}}(s, b, m^{\text{act}}) + \mathbb{E}_{\theta\sim b, m^{\text{cor}}}\big[\, R^{\text{act}}(s', b') - C(m^{\text{cor}})\,\big] \\ \text{where} \quad R^{\text{act}}(s, b, m^{\text{act}}) &= \bar{G}(s, b, m^{\text{act}}) - C(m^{\text{act}}) \\ &= \mathbb{E}_{\theta\sim b, m^{\text{cor}}}[\hat{G}_{\text{LLM}}(m^{\text{act}}, \theta)] - C(m^{\text{act}}) \end{aligned}$$

Under *user termination*, the net immediate reward $R^{\text{act}}(s, b, m^{\text{act}})$ is an expectation over cases where: (i) the user accepts m^{act} (resulting in positive goal reward $\hat{G}_{\text{LLM}}(m^{\text{act}}, \theta) > 0$); (ii) the user's intent θ leads them to reject m^{act} and give a correction m^{cor} (resulting in zero goal reward $\hat{G}_{\text{LLM}}(m^{\text{act}}, \theta) = 0$). While the user model $\hat{\pi}^{\text{user}}$ is technically required to determine acceptance vs. correction, our implementation avoids this check and directly takes the belief-weighted average over $\hat{G}_{\text{LLM}}(m^{\text{act}}, \theta)$. This is a good approximation since $\hat{G}_{\text{LLM}}(m^{\text{act}}, \theta) \approx 0$ for any θ that would lead to a correction.

For the residual expectation $\mathbb{E}_{m^{\text{cor}}}[\, R^{\text{act}}(s', b') - C(m^{\text{cor}})]$, the expectand is only non-zero when a correction m^{cor} is issued. We can thus rewrite it as:

$$\mathbb{E}_{m^{\text{cor}}}\big[\, R^{\text{act}}(s', b') - C(m^{\text{cor}})\,\big] = \mathbb{E}_{\theta\sim b}\big[P_{\text{cor}}(\theta, m^{\text{act}})\big(\, R^{\text{act}}(s', b') - \mathbb{E}_{m^{\text{cor}}}[C(m^{\text{cor}})]\,\big)\,\big],$$

where $P_{\text{cor}}(\theta, m^{\text{act}})$ is the probability that a user with intent θ corrects the answer m^{act}. To avoid sampling $\hat{\pi}^{\text{user}}$ many times to estimate P_{cor}, we approximate it as a fixed value across all θ and m^{act}, parameterized by the uncertainty of the agent's current particle belief $b = \{(\theta^n, w^n)\}_{n=1}^N$: $P_{\text{cor}}(\theta, m^{\text{act}}) \approx 1 - \max_n w^n$. When the agent is uncertain (a near-uniform belief with low peak probability $\max_n w^n$) corrections are likely across the board, whereas a confident agent trusts that the user is largely satisfied.

Factoring P_{cor} out of the expectation gives the value of acting via m^{act} that REVOIR evaluates at decision time:

$$Q^{\text{act}}(s, b, m^{\text{act}}) \approx R^{\text{act}}(s, b, m^{\text{act}}) + \big(1 - \max_n w^n\big) \sum_{n=1}^N w^n \,\big(\, R^{\text{act}}(s', b'_n) - C(m_n^{\text{cor}})\,\big), \quad (12)$$

where m_n^{cor} is the simulated correction for intent θ^n and b'_n is the resulting updated belief. Under *agent termination*, the correction branch is absent ($P_{\text{cor}} = 0$), and so the value of acting with m^{act} reduces to $Q^{\text{act}}(s, b, m^{\text{act}}) = R^{\text{act}}(s, b, m^{\text{act}})$.

C.3 Baselines for CondAmbigQA

We compare REVOIR against EIG maximization, bare prompting, generic inference-time reasoning algorithms, and models finetuned to ask clarifying questions:

- **REIGN.** A REVOIR variant whose (proxy) goal reward is $\hat{G}(s, \theta) = \log b(\theta)$. The expected goal reward is thus the negative entropy of the belief $-H(\hat{b})$ (see Appendix B), which we normalize to $1 - H(\hat{b}_t)/H_{\max} \in [0, 1]$ to ensure a similar scale as our cost function. REIGN is thus the cost-aware counterpart of the EIG maximization strategy in Bayesian experimental design (Lindley, 1956; MacKay, 1992; Rainforth et al., 2024), and a stand-in for other EIG-style algorithms like Hu et al. (2024). Since EIG is indifferent to whether a sharper belief yields a better answer, comparing REIGN to REVOIR isolates the value of *task-grounded* VoI.
- **Direct Answer.** The LLM is minimally prompted and directly answers the ambiguous question without any clarification.
- **ReAct.** One of the most widely adopted frameworks for LLM-based agents (Yao et al., 2022), ReAct interleaves chain-of-thought reasoning with action execution. The ask-vs-act decision is entirely implicit in the LLM's generation with no explicit belief, no principled stopping criterion, and no cost accounting.

- **Self-Consistency Best-of-N ReAct (SC-BoN-ReAct).** Generates K ReAct samples, then decides to ask or act via majority vote among the samples (a.k.a. self-consistency, Wang et al. (2023)), with a fitted threshold for determining majority. Out of the $N \leq K$ majority samples, the best sample is selected with an LLM judge. This judge is similar to REVOIR's proxy goal reward $\hat{G}_{\text{LLM}}$ when the majority votes to *act*, and uses a separate question selection prompt when the majority votes to *ask*.
- **Entropy Thresholding.** Shares the same particle belief as REVOIR but replaces VoI reasoning with a fixed stopping rule: ask until the normalized negative entropy $\frac{H_{\max}(b)-H(b)}{H_{\max}(b)}$ exceeds a threshold τ tuned on the validation set – i.e., when the belief is "concentrated enough".
- **ReflectionDPO-ReAct.** An LLM finetuned via our adaptation of the ReflectionDPO algorithm (Patel et al., 2025) to CondAmbigQA, learning when to ask clarifying questions through a privileged-teacher (Oracle-ReAct) reflection mechanism (Appendix H).
- **Oracle-ReAct.** A ReAct agent given the ground-truth extractive answer y as privileged context, which rephrases it fluently without asking any questions. It serves both as the oracle teacher for ReflectionDPO training and as a non-interactive topline.

D Additional Results on CondAmbigQA

D.1 Full Results of Llama-3.1-8B on CondAmbigQA

Under *agent termination*, REVOIR best trades-off correctness against clarification costs. We first report results in the "standard" QA setting where the interaction terminates after the agent gives an answer. Figure D.1 shows results on Llama-3.1-8B. Panel (a) reveals a clear accuracy–efficiency frontier on which REVOIR Pareto-dominates REIGN: at matched word budgets, REVOIR achieves higher correctness, reaching its best correctness (0.55) in 5.6 questions, whereas REIGN keeps asking up to 9 questions yet plateaus below 0.54 in correctness. This is because REIGN's objective is insensitive to whether a sharper belief improves the answer, so it asks until uncertainty is low rather than until expected task reward stops rising. Entropy Thresholding, which ignores costs and task reward, asks about 4 questions yet achieves the lowest correctness among multi-turn methods. Panel (b) shows the correctness increase of REVOIR over each baseline with standard errors (computed from 10,000 bootstrap samples), establishing that REVOIR's advantage over most methods (considering answer correctness alone) is robust when accounting for dataset noise.

Under *user termination*, REVOIR avoids costly questioning by being correction-aware. Figure D.2 presents results in the more realistic setting where the user can correct an inaccurate answer instead of terminating the conversation. While all interactive baselines anticipate user corrections, panel (a) shows that REVOIR rationally adapts to user corrections, using only 2–4 total user clarifications (i.e., questions asked + corrections given) with comparable correctness (0.56–0.57) to the agent termination setting. Panel (b) further shows that REVOIR's word count is substantially less than the budgeted amount (negative budget differences) — it acts well within budget by rationally relying on user corrections rather than asking more questions. REIGN, in contrast, keeps asking many questions: at its best it beats REVOIR slightly in correctness (0.58 vs 0.57), but only by relying on 6.9 clarifications compared to REVOIR's 3.6. Entropy Thresholding reaches 12 clarifications yet falls below REVOIR in correctness, having no way to reason that correction is cheaper than asking.

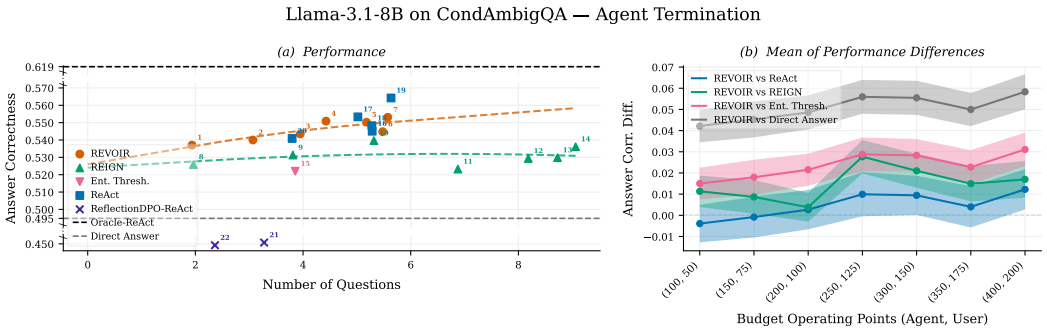


Figure D.1: CondAmbigQA results for Llama-3.1-8B under *agent* termination. **(a)** Answer correctness vs. average number of questions asked. Each numbered point corresponds to an agent/user word budget ($B_{\text{agent}}, B_{\text{user}}$) or (for ReAct) sample budget K; higher numbers map to higher values; Table D.1 lists exact values. **(b)** Mean pairwise differences in answer correctness between REVOIR and each baseline across budget points. ReAct is a single operating point broadcast across the budget axis. Shaded bands are bootstrapped standard errors, with $N_{\text{bootstrap}} = 10{,}000$.

	Method	Agent Budget	User Budget	No. Questions	Answer Correct.
1	REVOIR	100	50	1.94	0.5370
2	REVOIR	150	75	3.07	0.5400
3	REVOIR	200	100	3.95	0.5435
4	REVOIR	250	125	4.43	0.5509
5	REVOIR	300	150	5.18	0.5502
6	REVOIR	350	175	5.48	0.5448
7	REVOIR	400	200	5.57	0.5531
8	REIGN	100	50	1.96	0.5258
9	REIGN	150	75	3.81	0.5313
10	REIGN	200	100	5.32	0.5397
11	REIGN	250	125	6.88	0.5232
12	REIGN	300	150	8.18	0.5294
13	REIGN	350	175	8.73	0.5298
14	REIGN	400	200	9.06	0.5361
15	Ent. Thresh.	N/A	N/A	3.86	0.5220
16	SC-BoN-ReAct (K=5)	N/A	N/A	5.28	0.5451
17	SC-BoN-ReAct (K=10)	N/A	N/A	5.02	0.5533
18	SC-BoN-ReAct (K=15)	N/A	N/A	5.28	0.5482
19	SC-BoN-ReAct (K=20)	N/A	N/A	5.63	0.5642
20	ReAct	N/A	N/A	3.80	0.5409
21	ReflectionDPO-ReAct (ask-heavy)	N/A	N/A	3.28	0.4507
22	ReflectionDPO-ReAct (ask-moderate)	N/A	N/A	2.36	0.4492
23	Oracle-ReAct	N/A	N/A	0.00	0.6194
24	Direct Answer	N/A	N/A	0.00	0.4948

Table D.1: Details for each operating point in Figure D.1

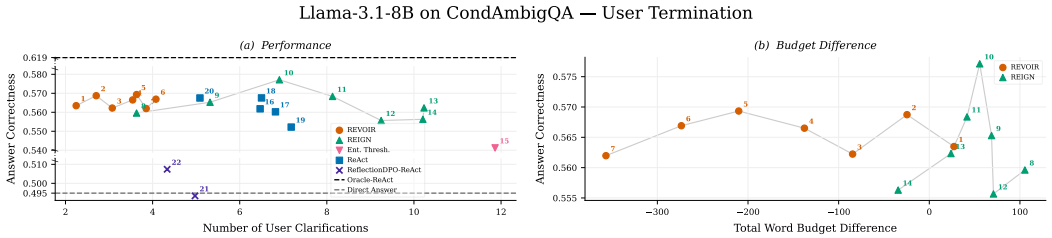


Figure D.2: CondAmbigQA results for Llama-3.1-8B under *user* termination. **(a)** Answer correctness vs. number of user clarifications (questions asked by the agent *or* corrections issued by the user). Each numbered point corresponds to an agent/user word budget ($B_{\text{agent}}, B_{\text{user}}$) or (for ReAct) sample budget K; higher numbers map to higher values; Table D.2 lists exact values. **(b)** The operating points of budget-aware methods (REVOIR and REIGN) plotted against total word budget difference (words used minus configured budget); negative values indicate the agent acted before exceeding its budget for cost-free words.

	Method	Agent Budg.	User Budg.	No. Ques.	No. Corr.	No. Clar.	Answer Correct.
1	REVOIR	100	50	1.00	1.25	2.24	0.5635
2	REVOIR	150	75	1.39	1.31	2.71	0.5687
3	REVOIR	200	100	1.73	1.34	3.07	0.5622
4	REVOIR	250	125	2.23	1.32	3.54	0.5665
5	REVOIR	300	150	2.40	1.23	3.63	0.5693
6	REVOIR	350	175	2.73	1.34	4.08	0.5669
7	REVOIR	400	200	2.58	1.27	3.85	0.5620
8	REIGN	100	50	2.24	1.39	3.63	0.5596
9	REIGN	150	75	4.08	1.24	5.32	0.5653
10	REIGN	200	100	5.66	1.25	6.91	0.5771
11	REIGN	250	125	6.84	1.29	8.13	0.5684
12	REIGN	300	150	7.77	1.48	9.25	0.5557
13	REIGN	350	175	8.55	1.68	10.23	0.5623
14	REIGN	400	200	8.53	1.67	10.21	0.5563
15	Ent. Thresh.	N/A	N/A	9.91	1.95	11.86	0.5412
16	SC-BoN-ReAct (K=5)	N/A	N/A	5.88	0.58	6.47	0.5618
17	SC-BoN-ReAct (K=10)	N/A	N/A	6.16	0.66	6.82	0.5602
18	SC-BoN-ReAct (K=15)	N/A	N/A	5.91	0.59	6.50	0.5676
19	SC-BoN-ReAct (K=20)	N/A	N/A	6.45	0.73	7.18	0.5522
20	ReAct	N/A	N/A	4.45	0.64	5.09	0.5675
21	ReflectionDPO-ReAct (ask-heavy)	N/A	N/A	4.14	0.83	4.97	0.4934
22	ReflectionDPO-ReAct (ask-moderate)	N/A	N/A	3.47	0.86	4.33	0.5074
23	Oracle-ReAct	N/A	N/A	0.00	0.00	0.00	0.6194
24	Direct Answer	N/A	N/A	0.00	0.00	0.00	0.4948

Table D.2: Details for each operating point in Figure D.2

LLAMA-3.1-8B TOKEN USAGE

Method	Input Tokens	Output Tokens	Total Tokens	Answer Correctness
REVOIR (100, 50)	7.0K	3.5K	10.6K	0.5371
REVOIR (150, 75)	9.6K	4.5K	14.2K	0.5400
REVOIR (200, 100)	11.8K	5.1K	16.8K	0.5435
REVOIR (250, 125)	12.7K	5.5K	18.2K	0.5509
REVOIR (300, 150)	14.8K	6.1K	20.9K	0.5502
REVOIR (350, 175)	15.3K	5.9K	21.3K	0.5448
REVOIR (400, 200)	15.6K	5.9K	21.5K	0.5531
ReAct	2.3K	0.3K	2.6K	0.5409
SC-BoN-ReAct (K=5)	17.3K	1.6K	18.9K	0.5451
SC-BoN-ReAct (K=10)	30.5K	3.1K	33.6K	0.5533
SC-BoN-ReAct (K=15)	47.4K	4.9K	52.3K	0.5482
SC-BoN-ReAct (K=20)	67.1K	6.9K	73.9K	0.5642

Table D.3: Token usage and answer correctness for REVOIR and SC-BoN-ReAct on Llama-3.1-8B under agent termination. SC-BoN-ReAct's token cost grows linearly with N, reaching $\sim$74K tokens at N=20 — over 3× the cost of the most expensive REVOIR operating point — yet correctness gains are marginal and non-monotonic. REVOIR at comparable token budgets achieves similar or greater correctness, demonstrating that principled ask-vs-act decisions are more token-efficient than inference-time scaling within a fixed policy.

D.2 GEMINI 3.5 FLASH-LITE AND CLAUDE HAIKU 4.5 ON CONDAMBIGQA

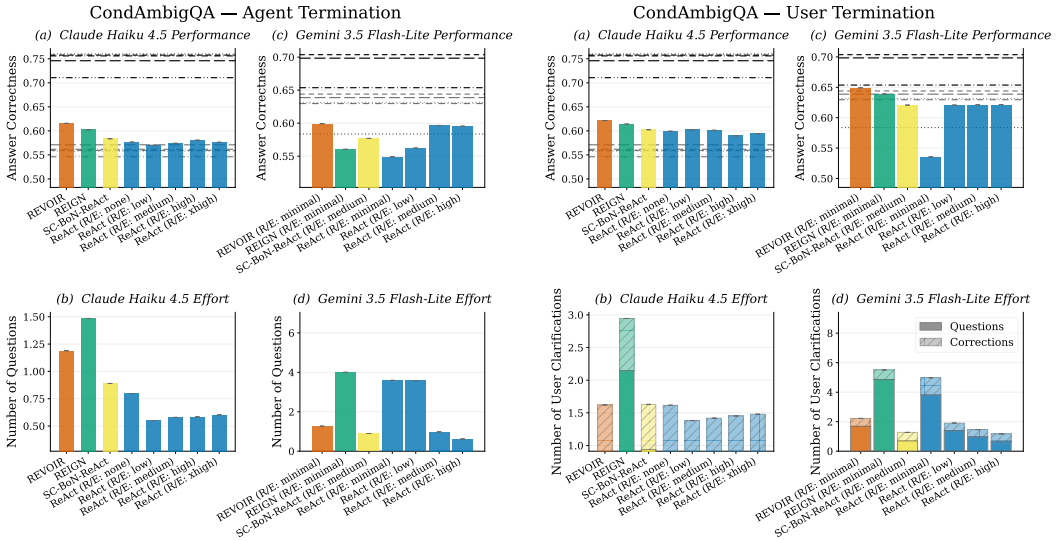


Figure D.3: CondAmbigQA results with Claude Haiku 4.5 and Gemini 3.5 Flash-Lite. ReAct variants and non-interactive base/top lines at available (Gemini 3.5 Flash-Lite) and emulated (Claude Haiku 4.5) reasoning-effort (R/E) levels. Horizontal lines in the top panels correspond to baselines and toplines across R/E levels: **gray lines** represent the **Direct Answer** baselines, and **black lines** represent **Oracle-ReAct** toplines, with line styles indicating effort levels: (– · · –) for `none` (or `minimal`), (— —) for `low`, (– · –) for `medium`, (· · ·) for `high`, and (– –) for `xhigh`.

We replicate experiments in Section 4.1 on Gemini 3.5 Flash-Lite and Claude Haiku 4.5, except for the Entropy Thresholding method due to cost constraints (Figure D.3). Since it is not possible to disable reasoning for Gemini 3.5 Flash-Lite, we evaluate REVOIR and REIGN with `minimal` reasoning. For SC-BoN-ReAct, we evaluate with `medium` reasoning

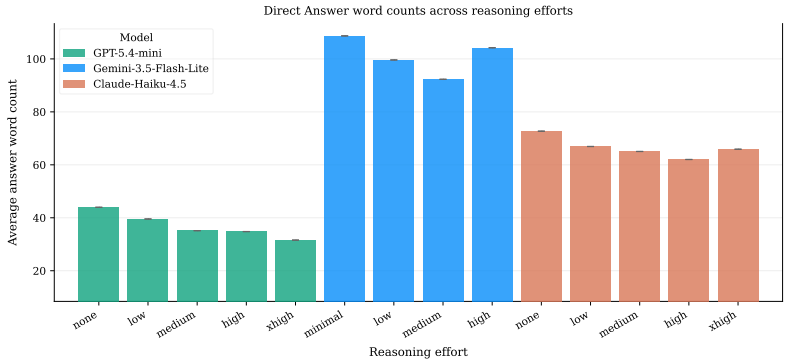


Figure D.4: Average word counts of non-interactive one-shot queries from frontier models across reasoning efforts.

level, since lower levels yield empty responses due to context-window-overflowed from reasoning. For Claude Haiku 4.5, we emulate thinking levels with the following token budgets for its *extending thinking mode*: {`low`: 1024, `medium`: 4096, `high`: 8192, `xhigh`: 12288}.

We find that across both models and termination settings, REVOIR generally achieves the highest `AnswerCorrectness` among interactive methods with minimal/no reasoning. Yet, each model exhibits different quirks. Both models outperform GPT-5.4-mini non-interactive toplines: Gemini 3.5 Flash-Lite bests at 0.70, Claude Haiku 4.5 bests at 0.76. Surpisingly, however, DirectAnswer baselines with non-`minimal` reasoning efforts with Gemini 3.5 Flash-Lite also outperform the interactive methods we evaluated — not what we expected of a baseline. Our qualitative inspections suggest that, in non-interactive settings, Gemini 3.5 Flash-Lite produces shotgun-style answers that are significantly more verbose than other frontier models (Figure D.4), and cover multiple potential information-seeking intents.

For instance, a direct answer by Gemini 3.5 Flash-Lite with `low` reasoning effort to the question "Who played Santa in Miracle on 34th St?" is as follows:

> "Multiple actors have played Santa Claus in the various versions of *Miracle on 34th Street*:
>
> - **1947 Original Movie: Edmund Gwenn** played Kris Kringle, and he won an Academy Award for his performance.
> - **1994 Remake: Richard Attenborough** played the role in the 94 feature film.
> - **1973 TV Movie: Sebastian Cabot** played Kris Kringle.
> - **1959 TV Broadcast (The Ford Star Jubilee): Macdonald Carey** played the role (though the 1955 live TV version featured **Thomas Mitchell** as Santa)."

In contrast, GPT-5.4-mini's answer to the same question with `low` reasoning effort is

> "In *Miracle on 34th Street* (1947), Santa was played by **Edmund Gwenn**. He even won an Academy Award for the role."

While Claude Haiku 4.5's answer, also with `low` reasoning effort, is

> In the classic 1947 film "Miracle on 34th Street," **Edmund Gwenn** played Santa Claus (Kris Kringle). He won an Academy Award for Best Supporting Actor for the role.
>
> In the 1994 remake, **Richard Attenborough** played the same character.

While such hedging-based answer strategies are appropriate under certain use cases *when the user is mentally prepared for an influx of information* (e.g., using a search engine), it might not always be appropriate. Regardless, our current evaluation rubrics do not discount

answers for length or intent-irrelevance as long as they stay on-topic and cover the target information-seeking intent (Appendix F.3), thus benefiting the shotgun approach.

Claude Haiku 4.5, on the other hand, uses very few questions across all interactive methods and settings. It also practically stops asking (except for REIGN) when user corrections are anticipated. These behaviors are, perhaps, reflective of different reasoning training strategies and heuristics. Nevertheless, among interactive methods, REVOIR still achieves the best performance by reasoning about the value of information associated with each action.

D.3 Results Under Inattentive User Simulator



(a) Agent termination. **(b)** User termination.

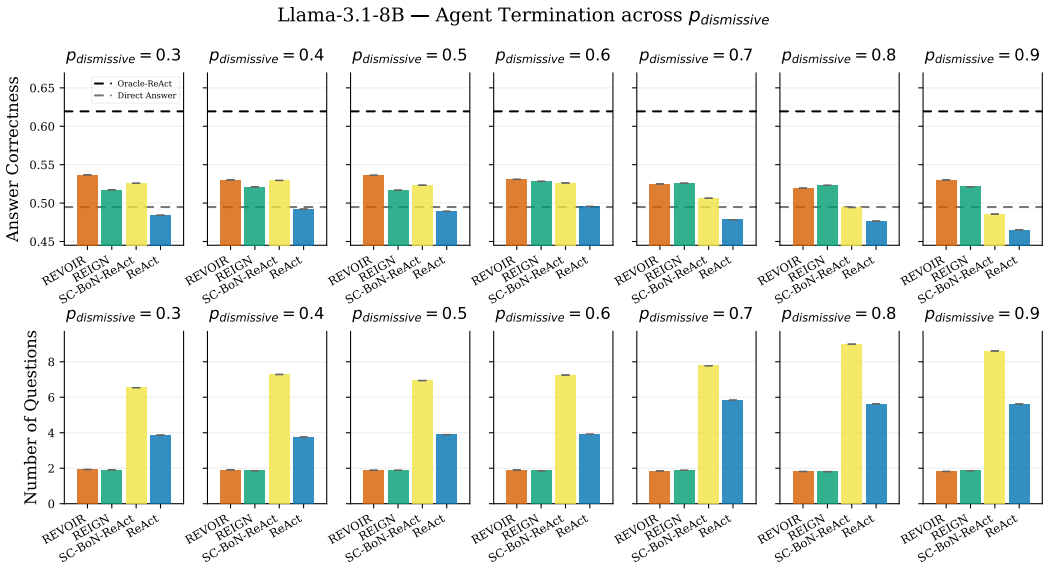
Figure D.5: CondAmbigQA results with an *inattentive user* simulation ($p_{dismissive} = 0.2$) for Llama-3.1-8B and Llama-3.1-70B.

We introduced an inattentive runtime user who sporadically gives one of the dismissive responses with probability $p_{dismissive} = 0.2$ at each clarification turn:

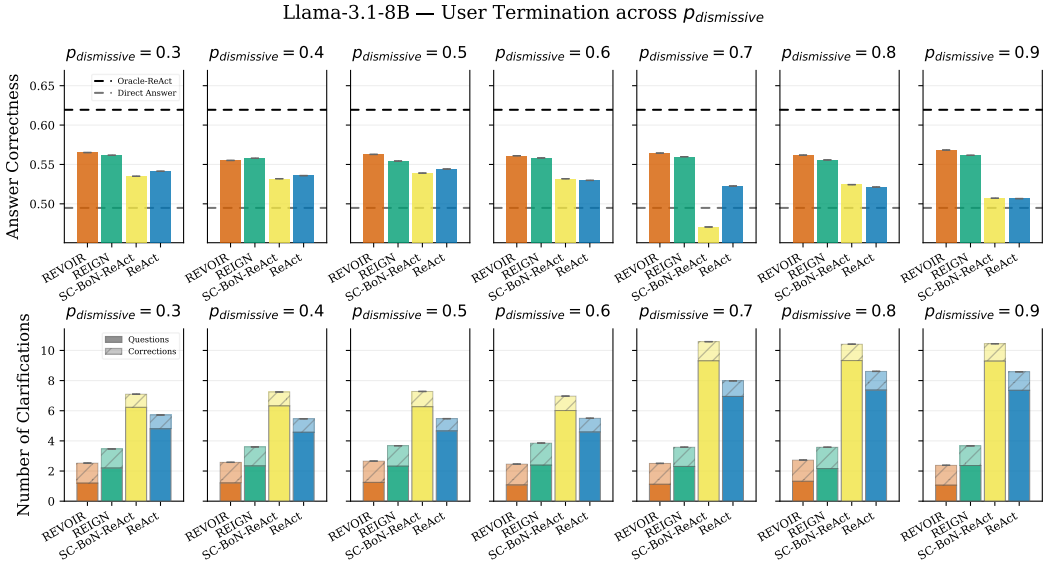
- "Not sure, whatever you think is right."
- "I don't remember exactly."
- "Just go with your best guess."
- "Hmm, I'm not really paying attention right now."

This noise is absent from the user modeled internally by REVOIR, creating a direct internal-runtime simulator mismatch. Due to compute and cost constraints, we only replicate the experiments in Section 4.1 under the inattentive user simulator for Llama-3.1-8B and Llama-3.1-70B, with planning budgets of $(B_{\text{agent}}, B_{\text{user}}) = (100, 50)$ (i.e., 100 assistant words and 50 user words). `AnswerCorrectness` remains independent of dialogue length and cost.

Our findings in Section 4.1 persist under unmodeled responses (Figure D.5). REVOIR has the highest correctness under agent termination for both backbones. Under user termination, it matches REIGN within 0.004 while asking about half as many questions for Llama-3.1-8B; for Llama-3.1-70B, it has the highest correctness and fewest total user interventions. Further results in Figure D.6 from Llama-3.1-8B with $p_{dismissive} \in \{0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ confirm the pattern. Notably, only REVOIR and REIGN are robust to dismissive noise, while ReAct baselines visibly degrade in `AnswerCorrectness` as $p_{dismissive}$ increases.



(a) Agent termination.



(b) User termination.

Figure D.6: Llama-3.1-8B CondAmbigQA results in *inattentive user* simulations with $p_{dismissive} \in \{0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$.

E Validating the LLM-simulated user and LLM judge with human annotations

We conducted a pilot conversation-replay study with human annotators to assess: (i) the user simulator's decision to terminate a conversation, (ii) the `AnswerCorrectness` rating of final answers. Annotators read logged GPT-5.4-mini conversations generated by REVOIR and ReAct (R/E: High, our strongest baseline) in the user termination setting.

Nine annotators were recruited from members of the university, and completed annotations for eight distinct item sets (with one response dropped due to an erroneous duplication of one item set between annotators). All reported intervals are two-sided 95% Wilson score intervals for binomial proportions, as Wilson intervals remain well behaved with modest samples and rates near zero or one.

E.1 Does the user simulator terminate where a person would?

For the first part of the study, each annotator is instructed to roleplay the user who asked the original ambiguous question. For every conversation each annotator is provided:

1. *The context you had in mind – but you didn't elaborate when asking*: the long-form disambiguating condition attached to the question;
2. *The unambiguous question you could have asked – but didn't*;
3. *How the conversation actually went*: the conversation transcript with interleaved turns (**Q:** the user, **A:** the assistant), ending with the assistant's final answer.

The gold answer is withheld from the annotator for this part of the study. After reading each episode, annotators answer a binary query:

> **Would the assistant's last answer have satisfied you?**
> *Yes — that would have answered me*
> *No — that would not have answered me*

Episodes are stratified by terminal outcome: *accepted on the first answer*, *accepted after clarification*, and *never accepted*. Each annotator is tasked with 20 episodes: 10 uniquely sampled episodes balanced across the two policies and 10 *anchor* episodes shared among all annotators. The anchors measure how often *people agree with each other* on this task. After per-form deduplication and removal of the eight attention-check responses, the analysis contains 160 substantive judgements: 80 judgements on 80 form-specific episodes and 80 judgements on the same 10 anchors. These judgements therefore cover 90 distinct episodes, rather than 160 distinct episodes.

For each episode, we compare one human verdict with the simulator's verdict. The form-specific episodes have one human judgement each; the anchors use the majority judgement across annotators. We do not break an evenly split vote toward either class. One anchor episode received a tied 4–4 judgments is subsequently dropped from the analysis, leaving the 89 verdicts in Table E.1. Of the remaining anchors, 5 episodes get unanimous human-simulator agreement, 1 episode gets unanimous human-simulator disagreement, and 3 episodes get majority human-simulator agreement (two 5-3 splits and one 6-2 split).

Table E.1: Human and simulated-user satisfaction verdicts.

	Simulated user		
Human verdict	**Satisfied**	**Not satisfied**	**Total**
Human: satisfied	**64**	10	74
Human: not satisfied	7	**8**	15
Total	71	18	89

The human-simulator agreement is $72/89 = 0.809$ (95% CI $[0.72, 0.88]$) across both simulator decisions. On the 71 episodes that the simulated user accepted, the human verdict also accepts 64, giving an acceptance agreement of $64/71 = 0.901$ (95% CI $[0.81, 0.95]$). Agreement is $26/29 = 0.897$ (95% CI $[0.74, 0.96]$) for episodes accepted after corrections and $38/42 = 0.905$ ($[0.78, 0.96]$) for episodes accepted on the first answer.

The disagreements are asymmetric. There are 10 episodes on which the simulator rejects an answer that the human verdict accepts, compared with 7/15 episodes on which the simulator accepts an answer that the human verdict rejects. 6 stricter simulator decisions concern answers that appear satisfactory but *conflict with the grounding condition*. Of the other five, four are genuine over-strict decisions on answers not conflicting the grounding condition, and one is due to a defective grounding condition that does not resolve ambiguity. Thus, the simulator more often errs toward continuing a conversation instead of ending it prematurely, while human is more lenient and concur on only $8/18 = 0.444$ of simulator rejections (95% CI $[0.25, 0.66]$).

E.2 Does the LLM judge track human answer-quality judgments?

In the second part, annotators are shown pairs of conversations corresponding to the same question, along with the grounding context and final answers from the two policiesin randomized order. They select the answer that they perceive as better, or mark the answers as equally good. Each retained annotator judges 12 pairs, yielding 96 comparisons across eight distinct responses.

Annotators mark 39/96 comparisons as equally good. On the 57 comparisons with an expressed preference, that preference agrees with the direction of `AnswerCorrectness` in 41 cases. Table E.2 summarizes these outcomes.

Table E.2: Agreement between human answer-quality preferences and `AnswerCorrectness`.

	Agree	**Disagree**	**Total**
Human preference vs. LLM judge ranking	**41**	16	57

On non-tied comparisons, human preferences and `AnswerCorrectness` agree at a rate of $41/57 = 0.719$ (95% CI $[0.59, 0.82]$). This provides evidence that the LLM judge tracks a human-recognizable notion of answer quality when a annotator perceives a difference.

F CondAmbigQA Prompt Library and Episode Lifecycle

This section presents the full set of prompt templates used in the ambiguous question-answering domain, together with descriptions of the episode lifecycle and user simulator. Listings show the verbatim templates with run-time values substituted by `{placeholders}`.

F.1 Episode Lifecycle

Each episode presents one ambiguous query m_1 whose latent intent θ is one of several valid scoping conditions. At each turn the agent updates its particle belief $\hat{b}_t = \{(\theta^n, w^n)\}$ over interpretations (Section 3.1) and chooses, via the cost-adjusted VoI rule of Section 3.3, between the *ask* policy π^{ask} (pose a clarifying question m^{ask}, receive a user answer m^{ans}) and the *act* policy π^{act} (commit to a final answer m^{act}). A hard budget caps the number of clarifying questions per episode. Under **agent termination**, m^{act} ends the episode; under **user termination**, the simulated user may instead issue a correction m^{cor} and let the agent answer again (Appendix F.2). The final answer is scored by the AnswerCorrectness judge (Appendix F.3).

F.2 User Simulator

The user model $\pi^{\text{user}}(a_t^{\text{user}} \mid s_{t-1}, \theta)$ is an LLM grounded in the ground-truth scoping condition θ and its retrieval context, which the agent never observes.

Answering clarifying questions. In response to a question m^{ask}, the user answers strictly from the ground-truth condition, and produces a dismissal when the question cannot be answered from it. This behaviour is shared by both termination variants.

Template 1: CondAmbigQA User Simulator System Prompt

```
You are a user who asked an ambiguous question. You know what you meant and
    will answer clarifying questions based only on the background information
     provided. Keep answers to one sentence. If the clarifying question
    cannot be answered from the background, say exactly: "That is irrelevant
    ."
```

Template 2: CondAmigQA User Simulator Message

```
My original question: "{original_question}"
What I meant: {condition}
Relevant background:
{context}

{history_block}Clarifying question from the assistant: "{clarifying_question
    }"
Answer (one sentence, or say you cannot clarify):
```

Termination and correction (user termination). Under user termination, after the agent commits to m^{act} the user first judges whether the answer matches the intended interpretation,

Template 3: CondAmbigQA User Simulator Termination System Prompt

```
You are a user who asked an ambiguous question and know exactly what you
    meant.
Given the assistant's answer, judge whether it aligns with your original
    intent.

Respond with a single word: YES or NO.
```

Template 4: CondAmbigQA User Simulator Termination Message

```
Your original question: "{original_question}"
What you actually meant: {condition}

The assistant answered: "{agent_answer}"

Does this answer align with what you intended to ask? (YES / NO)
```

and, if unsatisfied and the clarification budget remains, issues a one-sentence correction m^{cor} that steers the agent toward the correct interpretation without revealing the answer:

Template 5: CondAmbigQA User Simulator Correction System Prompt

```
You are a user who asked an ambiguous question. You know exactly what you
    meant. The assistant gave an off-topic answer. Write a single clarifying
    sentence that makes your original question clearer, pointing the
    assistant toward your intended meaning. Do NOT reveal, mention, or hint
    the answer itself. Write only the clarification sentence, nothing else.
```

Template 6: CondAmbigQA User Simulator Correction Message

```
My original question: "{original_question}"
What I meant: {condition}

{history_block}The assistant answered: "{wrong_answer}"
This answer is off-topic. Write a one-sentence clarification of my original
    question that would help the assistant understand what I meant, without
    giving away the answer:
```

F.3 Evaluation Judge and Planning-Time Reward Proxies

The evaluation metric (i.e. true goal reward) for CondAmbigQA is $G(s, \theta) = \texttt{AnswerCorrectness}(m^{\text{act}}, y_\theta)$, scored against the reference answer y_θ for the true condition θ by a rubric-based G-Eval judge. The rubric combines a *contradiction check*, an *omission penalty*, a *relevance check*, and an *acumen reward* for relevant answers that add correct depth beyond the extractive y. The judge is implemented with a DeepEval `GEval` object, producing a score in $[0, 1]$ based on criteria and evaluation steps registration.

Template 7: CondAmbigQA LLM-judge G-Eval Registration:

```
Metric Name: Answer Correctness
Criteria: Determine whether the actual answer is factually correct based on
    the question and expected answers.

Evaluation Steps:
  1. Check whether the facts in 'actual answer' contradicts any facts in '
     expected answers'.
  2. Heavily penalize omission of the question's information-seeking target
     in the answer.
  3. Ensure that the answer directly addresses the question without
     introducing irrelevant information.Reward if the 'actual answer' has
     more depth or better than the 'expected answers but still correct and
     relevant to the question
```

Planning-time reward proxy $\hat{G}_{\text{LLM}}$. At planning time no reference answer y_θ is available, so REVOIR scores a candidate answer against each hypothesised intent θ^n with the proxy $\hat{G}_{\text{LLM}}(a^{\text{act}}, \theta^n)$ used to form $\bar{G}(s, \hat{b}_t, a^{\text{act}}) = \sum_n w^n \hat{G}_{\text{LLM}}(a^{\text{act}}, \theta^n)$ (Appendix C.2). Instead of using DeepEval, which requires multiple LLM calls for each scoring, the proxy employs directly prompted verbalized scoring to produce a 0–10 score (to avoid complications with decimal tokens) that gets normalized to $[0, 1]$. It mirrors the answer-correctness rubric but drops the contradiction check (which requires y). Template 29

Template 8: CondAmbigQA LLM-judge Proxy Reward System Prompt

```
You are an evaluation judge. Your task is to determine whether a predicted
    answer correctly addresses an ambiguous question given a specific
    interpretation/condition.

Evaluation criteria: Determine whether the predicted answer is factually
    correct and complete given the asking intent.

Follow these evaluation steps:
1. Check whether the facts in 'predicted answer' addresses the asking intent.
2. Heavily penalize omission of information-seeking targets implied by the
    asking intent.
3. Ensure that the answer directly addresses the question under the asking
    intent without introducing irrelevant information.

Score interpretation:
  10 = answer fully and correctly addresses the asking intent
   5 = answer partially addresses the asking intent
   0 = answer is incorrect or misses the asking intent entirely

Use the full continuous range 0-10 (e.g. 3, 7, 8.5); do not limit yourself to
     only 0, 5, or 10.
Respond with ONLY a JSON object: {"score": <number 0-10>, "reason": "<one
    sentence>"}
```

Template 9: CondAmbigQA LLM-judge Proxy Reward Message

```
Input (ambiguous question): {question}

asking intent (true interpretation): {reference_condition}

Actual output (predicted answer): {predicted_answer}
```

F.4 REVOIR and REIGN Prompt Templates

REVOIR, REIGN, and Entropy Thresholding share the templates below; they differ only in the reward used in the decision rule — task reward $\hat{G}_{\text{LLM}}$ for REVOIR, belief concentration (negative entropy) for REIGN, and Entropy Thresholding — not in their prompts. Each template realizes a component of the algorithm in Appendix C.2.

Hypothesis proposer $Q_{\text{LLM}}(\theta \mid h_t, \hat{b}_{t-1})$. Proposes the particle set of interpretation hypotheses, conditioned on the dialogue and (after the first turn) the current belief, so that high-probability hypotheses are refined and unlikely ones pruned (Section 3.1).

Template 10: CondAmbigQA Hypothesis Proposer System Prompt

```
You are an expert at infering information-seeking intents in ambiguous
    questions. Given a question and the conversation so far, enumerate the
    most plausible interpretations of what the user might mean. Each
    interpretation should be a single sentence, meaningfully different from
    the others, e.g. targeting different contexts/details/aspects. Each
    interpretation must imply a different factual answer. If two
    interpretations could be satisfied with the same answer, merge them into
    one. The user might have multiple, compound, or broad information-seeking
     intents. The user might know partial details about their implied scoping
     contexts. Salient utterances usually mean they have obvious scoping
    contexts assuming common ground, but they might also be asking about the
    less obvious contexts. They also often neglect capitalization of proper
    nouns.
```

Template 11: CondAmbigQA Hypothesis Proposer Message

```
Now use your knowledge to infer this question.
Ambiguous question: "{question}"
{conv_block}{belief_block}Based on your knowledge and this context, enumerate
     at most {k} distinct diverse and plausible interpretations of the
    question, ranked by their plausibility across implications of subtexts
    they might have. Each interpretation must follow this format:
  The user is asking: "<an explicit question that pins down an asking intent
      >" (i.e., about <brief description and reasoning of the specific asking
       intent subtext and corresponding assumed contexts)
Output a numbered list -one interpretation per line, no extra commentary.
Plausible interpretations:
```

Consistency scorer $S_{\text{LLM}}(\theta^n, h_t)$. The semantic log-likelihood that drives the belief update (Section 3.1) is computed by scoring the user response to a clarifying question for consistency against each hypothesis on a 0–10 scale, yielding the weights w^n.

Template 12: CondAmbigQA Intent-Dialogue Consistency Scoring System Prompt

```
You are a dialog consistency judge. Your task is to determine whether ALL of
    the user's responses across a conversation are consistent with the
    interpeted context and information-seeking intents of their original
    question.

Follow these evaluation steps:
1. Read ALL clarifying questions and ALL user responses, not just the most
    recent one.
2. Check whether each response aligns with what someone holding the stated
    interpretation would say.
3. Penalize if any response contradicts or is unrelated to the interpretation.

4. Reward if all responses consistently confirm, elaborate on, or are
    naturally explained by the interpretation.

Score interpretation:
  10 = all responses strongly consistent with this interpretation
   5 = responses are neutral / neither confirm nor deny the interpretation
   0 = one or more responses contradict this interpretation

Use the full continuous range 0-10 (e.g. 3, 7, 8.5); do not limit yourself to
     only 0, 5, or 10.
Respond with ONLY a JSON object: {"score": <number 0-10>, "reason": "<one
    sentence>"}
```

Template 13: CondAmbigQA Intent-Dialogoue Consistency Scoring Message

```
Original question: "{original_question}"
Hypothesis (possible meaning): {hypothesis}

{conv_block}The assistant asked: "{clarifying_question}"
The user answered: "{observed_answer}"
```

User clarifying response forecast. At planning time, in order to score the value of each *question*, the agent role-plays a user holding a hypothesis θ^n and forecasts a concise response to the question.

Template 14: CondAmbigQA User Response Forecast System Prompt

```
You are roleplaying as a brilliant laconic user who asked an question with a
    specific information-seeking intent in mind. Answer the clarifying
    question as that user would, based only on your intended context and
    intent. You always pack the maximum relevant information into the fewest
    words -one concise sentence, If the question is not applicable to your
    intended meaning, say: "That is irrelevant."
```

Template 15: CondAmbigQA User Response Forecast Message

```
I asked: "{original_question}"
What I actually meant: {hypothesis}
```

```
{conv_block}The assistant now asks: "{clarifying_question}"

My answer (as few words as possible):
```

User correction forecast (user-termination mode). To score the value of each *answer* at planning time, the agent role-plays a user holding a hypothesis θ^n and forecasts a corresponding post-hoc correction. For this, we directly reuse Template 5 and Template 6, conditioned the hypothesized intent θ^n with the evaluating answer m^{act} injected into the history.

Question generator π^{ask} (best-of-K). Generates the K candidate clarifying questions whose VoI is evaluated by the 1-step lookahead. The model first reasons about what information is still needed, then proposes K distinct questions conditioned on the belief summary.

Template 16: CondAmbigQA Question Generator System Prompt

```
You are a knowledge assistant trying to ask a user clarifying questions to
    resolve the ambiguities in their information-seeking query. Given the
    user's original question and a set of contemplated interpretations (with
    belief weights), generate questions that would best clarify user's asking
     intent and implied context. The user might have multiple, compound, or
    broad information-seeking targets. The user might know partial details
    about their implied scoping contexts. Salient utterances usually mean
    they have obvious scoping contexts assuming common ground, but they might
     also be asking about the less obvious contexts.
```

Template 17: CondAmbigQA Question Generator Message

```
Here is the case you are clarifying.
Original question: "{question}"
{conv_block}Possible interpretations (with probability weight):
{interp_list}
First, generate a line of reasoning in this format Thought: <one sentence of
    reasoning about what you still need to know> Then, generate exactly {k}
    distinct clarifying questions. Output a numbered list -one question per
    line.
Clarifying questions:
```

Answer generator π^{act}. When the agent commits to an answer, a single response is generated conditioned on the full belief summary (the best-of-K act policy with K=1); its expected reward $\bar{G} = \sum_n w^n \hat{G}_{\text{LLM}}(a^{\text{act}}, \theta^n)$ supplies the value of acting.

Template 18: CondAmbigQA Answer Generator System Prompt

```
You are a factual knowledge assistant. Answer the user's question given your
    knowledge and the subtext they likely meant. Provide a complete, factual,
     and substantial answer that include all relevant details.Use multiple
    sentences if needed but prefer succinct information-densed answers.Do not
     regurgitate the user's clarifying answers.The user might have multiple,
    compound, or broad information-seeking targets.
```

Template 19: CondAmbigQA Answer Generator Message

```
The user asked: "{original_question}"
{conv_block}Contemplating interpretations of what the user meant (most likely
     first):
{belief_block}
Output exactly two lines:
Thought: <one sentence reasoning about which interpretation best fits the
    conversation>
Action: Answer | <Your complete factual answer to what the user intended to
    ask; include all relevant details>
```

F.5 ReAct and SC-BoN-ReAct

The ReAct baseline interleaves a one-sentence `Thought` with an `Action` that is either `Ask | <question>` or `Answer | <answer>`, maintaining no explicit belief. The system prompt differs by termination variant; the user-termination variant adds that the user may correct wrong answers, encouraging earlier commitment.

Template 20: CondAmbigQA ReAct Agent System Prompt (Agent Termination Mode)

```
You are a factual knowledge assistant answering an information-seeking
    question of a user. You may ask the user up to {max_q} clarifying
    question(s) to resolve ambiguity in a user's query before giving a final
    answer. Only ask questions that would help clarifying user's asking
    intent. Do not bounce back to the user rephrases of the question.
The user might have multiple, compound, or broad information-seeking targets.
     The user might know partial details about their implied scoping contexts
    . Salient utterances usually mean they have obvious scoping contexts
    assuming common ground, but they might also be asking about the less
    obvious contexts.

At each step output exactly two lines:
  Thought: <one sentence of reasoning about what you still need to know>
  Action: Ask | <your clarifying question>
OR
  Thought: <one sentence reasoning you have enough information>
  Action: Answer | <your complete factual answer>

Rules:
- One action per response, nothing else.
- Ask only if a clarification would substantially help clarifying the
    question.
- Answer must be complete and factual; include all relevant details. Use
    multiple sentences if needed but prefer succinct information-densed
    answers. Do not regurgitate the user's clarifying answers.
```

Template 21: CondAmbigQA ReAct Agent System Prompt (User Termination Mode)

```
You are a factual knowledge assistant answering an information-seeking
    question of a user. You may ask the user up to {max_q} clarifying
    question(s) to resolve ambiguity in a user's query before giving a final
    answer. Only ask questions that would help clarifying user's asking
    intent. Do not bounce back to the user rephrases of the question.
```

```
The user might have multiple, compound, or broad information-seeking targets.
     The user might know partial details about their implied scoping contexts
    . Salient utterances usually mean they have obvious scoping contexts
    assuming common ground, but they might also be asking about the less
    obvious contexts.

If your answer is off-topic, the user will correct you and you may try again.
     Use this to your advantage: you can commit to an answer sooner and
    update based on feedback, rather than exhausting all clarifying questions
     upfront.

At each step output exactly two lines:
  Thought: <one sentence of reasoning about what you still need to know>
  Action: Ask | <your clarifying question>
OR
  Thought: <one sentence reasoning you have enough information>
  Action: Answer | <your complete factual answer>

Rules:
- One action per response, nothing else.
- Ask only if a clarification would substantially help clarifying the
    question.
- Answer must be complete and factual; include all relevant details. Use
    multiple sentences if needed but prefer succinct information-densed
    answers. Do not regurgitate the user's clarifying answers.
```

Dialogue scaffolding. The initial user turn presents the ambiguous query; user answers are injected as observation turns; an exhausted question budget triggers a forcing message; under user termination, corrections are injected as clarification observations.

Template 22: CondAmbigQA ReAct Agent Initial Message

```
Question: "{question}"

Choose your first action.
```

Template 23: CondAmbigQA ReAct Agent Message

```
User: {response}
```

Template 24: CondAmbigQA ReAct Agent Exhausted Budget Trigger Message

```
You have used all your clarifying questions. You must answer now.

Choose your final action (Answer only).
```

Template 25: CondAmbigQA ReAct Agent User Correction Message

```
Your answer was off-topic. The user clarified: {clarification}

Choose your next action.
```

SC-BoN-ReAct selection judges. SC-BoN-ReAct requests K samples from ReAct completions at generation-time, decides the macro-action (ask vs. answer) by self-consistency vote, and selects the best candidate among the $N \leq K$ winning-type actions with an LLM judge. The two selection judges receive all same-type candidates with the question and dialogue history and return the best index.

Template 26: SC-BoN-ReAct Question Selection System Prompt

```
You are an evaluation judge selecting the best clarifying question to ask a
    user whose original information-seeking question is ambiguous.

Evaluation criteria: Determine which candidate question would most help
    disambiguate the user's asking intent based on the dialogue so far.

Follow these evaluation steps:
1. Check whether each candidate targets a genuine ambiguity in the original
    question that is not already resolved by the conversation.
2. Penalize questions that merely rephrase the original question or repeat
    information already established in the dialogue.
3. Prefer questions that are specific, answerable, and would maximally narrow
     down the user's true intent given the partial history.

Respond with ONLY the number of the best candidate, e.g. "2".
```

Template 27: SC-BoN-ReAct Question Selection Message

```
Original question: {question}

Dialogue so far:
{dialogue}

Candidate clarifying questions:
{candidates}
```

Template 28: SC-BoN-ReAct Answer Selection System Prompt

```
You are an evaluation judge selecting the best final answer to an ambiguous
    information-seeking question, given the clarifying dialogue that has
    taken place.

Evaluation criteria: Determine which candidate answer is most factually
    correct and complete given the user's asking intent as revealed by the
    conversation.

Follow these evaluation steps:
1. Check whether each candidate directly addresses the information-seeking
    target implied by the original question and the dialogue.
2. Heavily penalize omission of the information-seeking target or the
    introduction of irrelevant information.
3. Prefer answers that are accurate, concise, and consistent with all user
    responses in the dialogue without contradicting them.

Respond with ONLY the number of the best candidate, e.g. "2".
```

Template 29: SC-BoN-ReAct Answer Selection Message

```
Original question: {question}

Dialogue so far:
{dialogue}

Candidate answers:
{candidates}
```

F.6 ReflectionDPO-ReAct

ReflectionDPO-ReAct uses the ReAct templates above with a finetuned model; the data-generation pipeline is described in Appendix H. The privileged reflection template below — which appends the ground-truth answer and asks for the `Answer` action that should have been produced — is used both to generate the Answer-type training signal and, at inference time, by the Oracle-ReAct teacher.

Template 30: CondAmbigQA Privileged Reflection Message

```
Reflect on this conversation so far with this privileged information. Given
    that the elaborate extractive answer from the correct document is: "{
    a_star}", generate the Answer action you should have given at this point.

Output exactly two lines:
  Thought: <one sentence reasoning you have enough information to answer>
  Action: Answer | <your concise answer aligned with the true extractive
      answer>
```

F.7 Non-Interactive Baselines

Direct Answer answers the ambiguous query in a single turn under full ambiguity with a minimal system prompt:

Template 31: CondAmbigQA Direct Answer System Prompt

```
You are a helpful assistant.
```

Oracle-ReAct (the ReflectionDPO teacher and non-interactive topline) is given the ground-truth answer y at every call via the reflection template above and rephrases it without asking.

Oracle reference submits y verbatim with no LLM call, serving as a metric ceiling.

G CondAmbigQA Parameters

We detail the hyperparameters of the CondAmbigQA experiments and the assignment of LLMs to the modular roles of REVOIR.

G.1 Environment Roles

As noted in the main text, REVOIR's reasoning components — the hypothesis proposer (Q_{LLM}), the consistency scorer S_{LLM}, the question generator (π^{ask}), the answer generator (π^{act}), and the judge/scorer ($\hat{G}_{\text{LLM}}$, S_{LLM}) — can each be instantiated with a different model, yet we opt for a unifed backbone across experiments to isolate the effect of the

method. Other LLM-based modules required in the experimental environment are the user simulator π^{user} and the post-episode evaluation judge G. Together, our implementation groups these into three configurable roles:

- **Agent model:** drives the reasoning process of the assistive agent.
- **User model:** drives the user simulator π^{user} (clarification answers, satisfaction checks, post-hoc corrections).
- **Judge model:** drives the post-episode AnswerCorrectness evaluation.

In our reported experiments:

- **Agent model:** Inference-time methods (REVOIR, REIGN, Entropy Thresholding, ReAct, SC-BoN-ReAct) are evaluated with three models: `gpt-5.4-mini-2026-03-17`, `meta-llama/Llama-3.1-70B-Instruct`, and `meta-llama/Llama-3.1-8B-Instruct`. ReflectionDPO-ReAct is finetuned from `meta-llama/Llama-3.1-8B-Instruct`.
- **Reasoning effort (`gpt-5.4-mini` only):** The VoI agents (REVOIR, REIGN, Entropy Thresholding) and SC-BoN-ReAct use `reasoning_effort: none`; for ReAct, Oracle-ReAct, and Direct Answer sweep effort over {none, low, medium, high, xhigh}.
- **User and judge models:** `gpt-4o-mini-2024-07-18` throughout, for both the user simulator and the evaluation judge.

G.2 Global Episode Parameters

Shared across all interactive methods:

- **Question budget:** 10 clarifying questions per episode.
- **Clarification budget:** under user termination, the user issues at most 5 corrections after unsatisfactory answers.
- **Word budgets** $(B_{\text{agent}}, B_{\text{user}})$**:** the piecewise-linearly word-budgeted cost function C defined in Section 4.1 penalizes agent and user messages beyond per-message word budgets. The main REVOIR and REIGN runs use $(B_{\text{agent}}, B_{\text{user}}) = (100, 50)$; for Llama-3.1-8B we report a budget sweep with $B_{\text{agent}} \in \{100, 150, \ldots, 400\}$ and $B_{\text{user}} = B_{\text{agent}}/2$. Budgets are disabled (0) for Entropy Thresholding and do not constrain generation directly.
- **Agent generation:** all expert-model generation — hypothesis and question proposal, mentalized forecasting, consistency scoring, and answer generation — uses `temperature: 0.0` (greedy), with `agent_max_tokens: 2048`; candidate hypotheses and questions are produced as a single list per call rather than by repeated sampling.
- **Macro-action selection:** $\arg\max$ over V^{ask} vs. V^{act} (greedy)
- **User generation:** greedy decoding with `user_temperature: 0.0`, `user_top_p: 1.0`, and `user_max_tokens: 128`.
- **Judge evaluation:** the post-episode AnswerCorrectness metric uses DeepEval's `GEval`. The underlying LLM is calle at `temperature: 0` and `top_logprobs: 20`. The metric forms a probability-weighted expectation of the integer scores over the (default) 0–10 range and normalizes it to $[0, 1]$.

G.3 VoI Planner Parameters (REVOIR, REIGN, Entropy Thresholding)

- **Belief particles:** N=5 interpretation hypotheses in the belief $\hat{b}_t$ (`n_hypotheses`).
- **Question generation:** $K = 5$ candidate questions generated and evaluated by the best-of-K ask policy per turn.
- **Act policy:** a single belief-conditioned answer, i.e. best-of-K with K=1, whose expected reward $\bar{G} = \sum_n w^n \hat{G}_{\text{LLM}}$ supplies V^{act} (`optimal_answer_approx: direct_belief_condition`).

Table G.1: The normalized concentration threshold τ of $1 - H(\hat{b}_t)/H_{\max}$, above which Entropy Thresholding answers immediately, is tuned on the Dev split, per agent model and termination variant.

Configuration	Agent term.	User term.
`gpt-5.4-mini`	0.85	0.65
`Llama3-70B`	0.25	0.85
`Llama3-8B`	0.65	0.45

G.4 Entropy Thresholding

G.5 SC-BoN-ReAct Parameters

- **Self-consistency ensemble size:** $K = 5$ in the main comparisons; for Llama-3.1-8B we additionally sweep $K \in \{5, 10, 15, 20\}$.
- **Sampling temperature:** 0.7 for the pool, required to be > 0 for diversity (`sample_temperature`); the best-of-N selection judge runs greedily.
- **Self-consistency threshold τ:** the fraction of `Answer` votes (over parseable samples) required to commit to answering rather than asking, tuned on the Dev split per agent model and termination variant. In the Llama-3.1-8B sweep, τ is additionally tuned per K (Table G.2).

Table G.2: Self-consistency threshold τ for SC-BoN-ReAct, tuned on the Dev split.

Configuration	Agent term.	User term.
`gpt-5.4-m`, K=5	0.9	0.6
`Llama3-70B`, K=5	0.6	0.7
`Llama3-8B`, K=5	0.6	0.6
`Llama3-8B`, K=10	0.5	0.6
`Llama3-8B`, K=15	0.6	0.5
`Llama3-8B`, K=20	0.7	0.7

H Adapting ReflectionDPO to Ambiguous Question Answering

We adapt the ReflectionDPO algorithm of (Patel et al., 2025) to CondAmbigQA. The training data is curated by dynamically comparing a standard ReAct policy against a privileged oracle. First, the student ReAct model rolls out its interaction with the user simulator. Unlike ADAPT, where the reference actions at each step are entirely generated by a teacher conditioned on the partial history, CondAmbigQA provides extractive ground-truth answers — the "true plan" degenerates into a known answer y, which we use directly as the oracle signal.

At each conversational turn, the student is presented with the partial history together with a ReAct-formatted oracle action: the `Thought` block contains the ground-truth clarification question and its triggering condition, while the `Answer` block contains the ground-truth extractive answer. The student is then prompted to *reflect* on what question it could have asked to elicit the missing information required to arrive at this oracle answer. This process yields three components per turn: the student's original rollout action, the oracle's answer, and the student's candidate reflection question. The conditional log-probability of generating the ground-truth answer is also leveraged as the utility score for preference-pair filtering.

These components are filtered to construct preference pairs. The student's original rollout is assigned as the rejected response, and either the reflection question (Ask) or the oracle answer (Answer) is selected as the chosen response. However, because the oracle answer a_{oracle} is typically verbose, its generative log-probability under the student is naturally low relative to the rollout; directly performing preference alignment on rollout–oracle pairs risks degrading the model's general fluency. To mitigate this, we prompt the Oracle-ReAct model — privileged to the oracle extractive answer — to *repackage* a_{oracle} in its own words, and use that rephrased version a_{teacher} as the preferred response whenever Answer is the chosen action.

Two scalar thresholds govern which (episode, step) records become training pairs. ε_1 gates Ask inclusion: a step produces an Ask pair only when the oracle answer log-probability improves by more than ε_1 nats per token after the model receives the reflection question,

Algorithm 1 Reflection Mechanism

function REFLECTION($P_{\pi_{\text{student}}}(\cdot)$, a_{student}, a_{oracle}, h_t)
  $a_{\text{teacher}} \leftarrow P_{\pi_{\text{teacher}}}(\cdot \mid h_t, a_{\text{oracle}})$ ▷ Oracle-ReAct rephrases extractive answer
  $a_q \leftarrow \text{GETQUESTION}(P_{\pi_{\text{student}}}, a_{\text{student}}, a_{\text{oracle}})$
  $r \leftarrow \text{ASKUSER}(h_t, a_q)$
  $\Delta_q \leftarrow \log P_{\pi_{\text{student}}}(a_{\text{oracle}} \mid h_t \oplus a_q \oplus r) - \log P_{\pi_{\text{student}}}(a_{\text{oracle}} \mid h_t)$
  $\Delta_t \leftarrow \log P_{\pi_{\text{student}}}(a_{\text{student}}) - \log P_{\pi_{\text{student}}}(a_{\text{teacher}})$
  if $\Delta_t < 0$ **then**
    $a_{\text{chosen}} \leftarrow a_{\text{teacher}}$ ▷ teacher already more likely than rollout
  else if $\Delta_q > \varepsilon_1$ **then**
    $a_{\text{chosen}} \leftarrow a_q$ ▷ reflection meaningfully helps
  else if $\Delta_t < \varepsilon_2$ **then**
    $a_{\text{chosen}} \leftarrow a_{\text{teacher}}$ ▷ rollout close enough to teacher
  else
    $a_{\text{chosen}} \leftarrow$ None ▷ skip this data point
  end if
  return a_{chosen}
end function

i.e. $\Delta_q > \varepsilon_1$. A lower ε_1 admits more ASK pairs but at the cost of including weak reflection signals. ε_2 gates ANSWER inclusion: a step produces an ANSWER pair when the rollout is worse than the teacher answer by less than ε_2 nats per token, i.e. $\Delta_t < \varepsilon_2$. A higher ε_2 admits more ANSWER pairs but risks including teacher answers that are noticeably less fluent than the rollout.

We set a quality ceiling of $\varepsilon_2 \leq 0.30$ based on the empirical distribution of Δ_t: at $\varepsilon_2 = 0.30$ the mean gap is ≈ 0.19 nats/token, indicating the teacher answer can still reasonably be preferred; beyond 0.50 the mean exceeds 0.45 nats/token, a regime where teacher answers are markedly less natural than rollouts. We impose $\varepsilon_1 \geq 0.05$ to exclude near-zero reflection gains that provide no meaningful learning signal, and cap the ASK:ANSWER ratio at 4:1 to prevent the policy from becoming reflexively question-happy.

We sweep $\varepsilon_1 \in [0.01, 0.80]$ and $\varepsilon_2 \in [0.02, 0.30]$ on the full reflection corpus (17,090 records) and select the point that maximises total kept pairs subject to the constraints above.

Operating points.

Setting	ε_1	ε_2	Filtered	Ask:Ans (filtered)	Cap C	Strat. train	Ask:Ans (strat.)
Ask-heavy	0.09	0.30	5649	3.6 : 1	376	2669	3.8 : 1
Ask-moderate	0.13	0.30	3366	1.5 : 1	218	1547	1.7 : 1

The *ask-heavy* setting maximises dataset size within the ratio constraint; the *ask-moderate* setting tightens ε_1 to equalise the ASK and ANSWER signal, trading coverage for a more symmetric preference distribution. Both settings use $\varepsilon_2 = 0.30$ (the quality ceiling), which is also the global optimum for ANSWER inclusion under our constraints.

Step-stratified sampling. Reflection pairs are not uniformly distributed across conversation steps. Early steps ($t = 0$–2) are over-represented because few episodes reach late steps under the default ReAct policy. Training on the raw distribution therefore biases the policy toward early-step behaviour and under-trains it on mid-to-late recovery moves.

To correct for this, we apply step-stratified sampling. Let $\mathcal{D}_t$ denote the set of retained pairs at step t and let $C = \underset{t}{\text{median}}\, |\mathcal{D}_t|$. The stratified training set is

$$\mathcal{D}^* = \bigcup_{t=0}^{T} \text{Sample}\big(\mathcal{D}_t,\ \min(|\mathcal{D}_t|, C)\big), \tag{13}$$

where each step is independently sampled without replacement up to C pairs. Steps with fewer than C pairs are kept in full; over-represented steps are downsampled to exactly C.

Within each step, the Ask:Answer ratio is preserved proportionally so that the relative preference signal at each turn is not distorted.

Using the median rather than the minimum prevents the long tail of rare late steps from collapsing the dataset to a trivially small size, while still substantially flattening the step distribution, which shifts the global ratio only slightly (ask-heavy $3.6:1 \rightarrow 3.8:1$; ask-moderate $1.5:1 \rightarrow 1.7:1$).

Training details. We finetune using LoRA (Hu et al., 2022) with rank $r = 4$ and scaling factor $\alpha = 16$, matching the adapter configuration of the original ADAPT codebase (Patel et al., 2025).

We deviate from ADAPT in the choice of preference optimization loss. ADAPT uses DPO (Rafailov et al., 2023), which is appropriate when chosen and rejected responses are short, atomic actions: the log-probability differences are well-calibrated and the relative contrastive objective is stable. In our setting, both chosen responses (rephrased answers or reflection questions) and rejected responses (student rollout actions) are elaborate, multi-sentence texts. DPO applied to long-form outputs is susceptible to reward hacking via likelihood displacement: the optimizer can satisfy the contrastive objective by reducing the log-likelihood of *both* responses while keeping their relative ordering intact, which degrades generation quality without meaningfully improving preference alignment.

We therefore use ORPO (Hong et al., 2024), which folds a negative log-likelihood term on the chosen response together with an odds-ratio penalty on the rejected response into a single objective, without requiring a reference model or a separate SFT warm-up phase:

$$\mathcal{L}_{\text{ORPO}} = -\log P_\theta(y_w \mid x) \; - \; \lambda \cdot \log \sigma\left(\log \frac{P_\theta(y_w \mid x)}{1 - P_\theta(y_w \mid x)} - \log \frac{P_\theta(y_l \mid x)}{1 - P_\theta(y_l \mid x)}\right), \tag{14}$$

where y_w and y_l are the chosen and rejected responses respectively and λ is a weighting coefficient. The NLL term anchors the absolute likelihood of chosen responses upward, closing off the reward-hacking mode available to DPO, while the odds-ratio term suppresses rejected responses relative to this moving anchor rather than relative to a frozen reference policy. A natural alternative would be to anchor on the gold extractive answer y^* directly, grounding the model in factual content independent of how well the teacher rephrased it. However, the extractive answers in CondAmbigQA are terse, unnaturally phrased fragments that make poor generation targets; anchoring on y^* would penalize fluent paraphrases and conflict with the fluency goal that motivated using a_{teacher} in the first place. We therefore anchor on $y_w = a_{\text{teacher}}$ throughout.

Why it underperforms. We speculate that the gap stems primarily from a mismatch between the training signal and test-time requirements: reflection questions are selected based on how much they improve the log-probability of the extractive oracle answer y, which may not correlate with whether a question improves open-ended generative answer quality. The small post-curation dataset likely compounds this, under-determining a reliable clarification policy rather than overcoming the base model's prior. More broadly, the result illustrates that finetuning a clarification policy does not transfer cheaply to arbitrary tasks and contexts as it demands task-specific data curation and can still optimize a proxy that diverges from the deployment objective. REVOIR, on the other hand, obtains its clarification behavior at inference time, with no training, by reasoning directly about the value of information for the task at hand.

I Extended Details on ADAPT

I.1 Adapting ADAPT to REVOIR

Two-stage Variant. The original ADAPT runtime freely interleaves questions and physical actions, making one-step VoI reasoning difficult to apply. We therefore decouple each episode into (i) an **elicitation stage**, in which REVOIR iteratively compares $V^{\text{ask}}(s, b)$ against $V^{\text{act}}(s, b)$ and either poses a clarifying question $m^{\text{ask}} \in \mathcal{M}^{\text{ask}}$ or commits to execution, and

(ii) an **execution stage**, in which π^{act} carries out a full multi-step plan conditioned on the updated belief $b_t(\theta)$, after which the episode terminates. We note that this two-phase decoupling *limits* the information gathering capacity of REVOIR, while preserving the full complexity of grounded planning in ADAPT.

I.2 REVOIR Variants in ADAPT

The ADAPT domain introduces two additional design choices in how REVOIR maintains and updates its belief $\hat{b}_t$, yielding four variants reported in our experiments.

Belief representation. In the *joint belief* variant, each particle θ^n is a candidate set of atomic preference statements drawn from the space of plausible user preferences for the task. The belief $\hat{b}_t$ is a distribution over these complete preference sets, and the consistency scorer $S_{\text{LLM}}(\theta^n, h_t)$ evaluates how well the full set θ^n explains the conversation so far ($K = 6$ particles). In the *factored belief* variant, rather than tracking complete sets, the agent maintains an independent Bernoulli probability $p_m \in [0, 1]$ for each atomic preference γ_m, representing the agent's belief that γ_m belongs to the user's true preference set θ. This representation supports a larger effective particle count since each particle corresponds to a binary inclusion decision per preference, and enables more targeted clarifying questions directed at individual uncertain dimensions of θ.

Weight update scheme. In the *batch rescoring* variant, particle weights are recomputed from scratch at each turn by scoring the full conversation history h_t against each particle in a single pass: $w_t^n = S_{\text{LLM}}(\theta^n, h_t)$. In the *incremental update* variant, retained particles cache their accumulated log-weight from the previous turn and augment it with only the new message's contribution, $w_t^n = w_{t-1}^n + S_{\text{LLM}}(\theta^n, h_t \mid h_{t-1})$, avoiding redundant recomputation. Particles that are pruned and replaced by fresh samples cannot benefit from this cache; they instead replay the full sequential accumulation from the start of the conversation to bring their weights up to date.

Goal reward approximation. Because plan execution in ADAPT is multi-step, evaluating $\bar{G}(s, b, a^{\text{act}})$ requires simulating the consequences of a full plan rather than a single action. REVOIR approximates this through a pipeline of three specialized modules — a *planner*, a *simulator*, and a *proxy judge* — each of which can in principle be instantiated with different models. First, the planner generates a single plan conditioned on the current belief summary and history. Second, the simulator takes the plan and the initial scene graph and produces an estimated final state s' by stepping through the plan's state changes. Third, the proxy judge evaluates s' against the current belief to produce a reward estimate. In our experiments, all three modules are instantiated with the same LLM, but the modular decomposition allows each to be replaced independently — for instance, with a specialized simulator or a finetuned reward model.

The reward aggregation in the judge differs between belief variants. Under *joint belief*, the judge scores the PSR of s' against each preference set particle θ^n and the expected reward is the weighted average $\bar{G}(s, b, a^{\text{act}}) = \sum_n w^n \cdot \text{PSR}(s', \theta^n)$. Under *factored belief*, a mean-field weighted reward is used instead:

$$\bar{G}(s, b, a^{\text{act}}) = \frac{\sum_m p_m \cdot \mathbf{1}[\gamma_m \text{ satisfied in } s']}{\sum_m p_m \cdot \mathbf{1}[\gamma_m \text{ satisfied or violated in } s']}, \tag{15}$$

where p_m is the Bernoulli inclusion probability of preference γ_m. This weights each preference's contribution by its posterior probability of belonging in θ, so preferences the agent is confident the user holds influence the reward estimate more than uncertain ones.

Combining the two belief representations with the two update schemes yields the four REVOIR variants — **Joint + Batch Rescoring**, **Joint + Incremental Update**, **Factored + Batch Rescoring**, and **Factored + Incremental Update**.

I.3 Full Results on ADAPT

Full results for ADAPT are presented in Table I.1.

Test Personas	Method	PSR (%)	#Q
Privileged / Forced			
Seen Personas	Always-Ask LLM	52.1 ± 0.9	22.2 ± 0.1
	Teacher LLM	65.5 ± 1.1	0.0
Unseen Personas	Always-Ask LLM	50.8 ± 1.6	22.4 ± 0.2
	Teacher LLM	65.5 ± 1.9	0.0
Fine-tuned			
Seen Personas	STaR-GATE	34.1 ± 0.8	2.0 ± 0.0
	Reflection-DPO	44.1 ± 1.0	9.8 ± 0.1
Unseen Personas	STaR-GATE	33.5 ± 1.4	2.0 ± 0.0
	Reflection-DPO	42.9 ± 1.6	9.5 ± 0.2
Few-shot ICL			
Seen Personas	**REVOIR (Joint + Batch Rescoring)**	49.8 ± 2.0	2.5 ± 0.3
	REVOIR (Joint + Incremental Update)	49.5 ± 1.9	2.4 ± 0.2
	REVOIR (Factored + Batch Rescoring)	**56.3 ± 1.3**	**1.7 ± 0.2**
	REVOIR (Factored + Incremental Update)	**58.8 ± 0.4**	**2.1 ± 0.2**
Unseen Personas	**REVOIR (Joint + Batch Rescoring)**	48.0 ± 7.4	2.4 ± 0.5
	REVOIR (Joint + Incremental Update)	50.2 ± 7.8	2.4 ± 0.3
	REVOIR (Factored + Batch Rescoring)	**55.8 ± 8.1**	**1.9 ± 0.2**
	REVOIR (Factored + Incremental Update)	**56.0 ± 3.5**	**2.0 ± 0.5**
Zero-shot / No fine-tuning			
Seen Personas	Never-Ask LLM	27.5 ± 0.7	0.0 ± 0.0
	Baseline LLM	34.7 ± 0.8	2.3 ± 0.0
	ReAct	36.1 ± 0.8	2.3 ± 0.0
	REVOIR (Joint + Batch Rescoring)	35.7 ± 3.8	2.2 ± 0.3
	REVOIR (Joint + Incremental Update)	35.0 ± 2.1	2.3 ± 0.3
	REVOIR (Factored + Batch Rescoring)	**49.2 ± 3.1**	**1.9 ± 0.4**
	REVOIR (Factored + Incremental Update)	**51.5 ± 2.5**	**2.0 ± 0.2**
Unseen Personas	Never-Ask LLM	28.6 ± 1.2	0.0 ± 0.0
	Baseline LLM	34.3 ± 1.3	2.3 ± 0.1
	ReAct	36.8 ± 1.4	2.4 ± 0.1
	REVOIR (Joint + Batch Rescoring)	36.4 ± 8.5	2.4 ± 0.2
	REVOIR (Joint + Incremental Update)	33.0 ± 8.0	2.0 ± 0.1
	REVOIR (Factored + Batch Rescoring)	**51.0 ± 7.4**	**2.0 ± 0.4**
	REVOIR (Factored + Incremental Update)	**49.5 ± 2.6**	**2.2 ± 0.3**

Table I.1: Main results on the ADAPT benchmark (4-fold cross-validation; std across split means). Patel et al. (2025) numbers are reproduced from their Table 1. In-context examples from seen personas injected at hypothesis particle regeneration.

Factored belief improves zero-shot performance significantly. In zero-shot, Joint-Belief REVOIR variants achieve 33.0–36.4% average PSR, comparable to ReAct and Baseline LLM, while Factored-Belief variants jump to 49.2–51.0%, nearly matching Always-Ask LLM (50.9–52.1%) with 11× fewer questions (2 vs. 22). The factored representation enables more targeted elicitation by tracking each preference independently, rather than scoring complete preference sets that may be poorly calibrated. However, Joint-Belief REVOIR still does well in the few-shot case even on unseen personas ($48.0 - -50.2$), indicating that joint belief updates can still be calibrated if there are good examples of what preference sets look like.

REVOIR with ICL outperforms fully trained methods. With few-shot ICL, Factored REVOIR reaches 56.3–58.8% average PSR on seen personas and 55.8–56.0% on unseen

personas, exceeding Always-Ask by 6 points and outperforming Reflection-DPO (44%) by 13–15 points — with $\sim 5\times$ fewer questions. This is significant given that REVOIR requires no task-specific training, showing in-context examples suffice to ground the beliefs and guide the planner. STaR-GATE, by contrast, fails to exceed the non-interactive Baseline LLM (34%), suggesting its training signal is too sparse to learn effective clarification.

Generalization to unseen personas. Factored REVOIR variants show minimal PSR degradation from seen to unseen personas (e.g., 58.8 $\rightarrow$ 56.0% for Incremental Update with ICL), with notably lower variance than Joint variants, which exhibit standard deviations of 7–8% on unseen personas. This suggests that factoring the belief over individual preferences produces representations that transfer more reliably across persona distributions.

J ADAPT Prompt Library and Episode Lifecycle

This section collects the prompt templates used in the preference-aligned household-assistance domain, together with a concise description of the episode lifecycle and user simulator. Listings show the templates with run-time values substituted by `{placeholders}`.

J.1 Episode Lifecycle

Each episode pairs a breakfast task with a persona whose hidden preference set $\theta = \{\gamma_1, \ldots, \gamma_M\}$ the agent never observes. Following the two-stage formulation, REVOIR runs an **elicitation stage** — iteratively comparing $V^{\text{ask}}(s, b)$ against $V^{\text{act}}(s, b)$ to either pose a clarifying question m^{ask} or commit — followed by an **execution stage**, in which π^{act} carries out a multi-step plan conditioned on the elicited belief $\hat{b}_t(\theta)$ without further questions. Episodes are scored by the preference satisfaction rate $G(s, \theta) = \text{PSR}$.

Evaluation splits. For comparability with the original ADAPT baselines — which are trained and then tested for generalization across personas — we reuse the benchmark's four-way split that crosses persona novelty (seen vs. held-out personas) with in-context guidance. Our methods, however, are training-free and operate in only two regimes: *few-shot*, with the ground-truth preference profiles of the seen personas injected as reference examples into the proposer and scorer prompts (shown as `{example_profiles}` below), or *zero-shot*, with no examples provided. Thus, for REVOIR, the seen-versus-held-out persona settings only reflect whether a tested personat appear among the few-shot examples.

J.2 User Simulator

Following Patel et al. (2025), the user simulator $\pi^{\text{user}}(\cdot \mid h_t, \theta)$ is an LLM role-playing the true persona, prompted with the task, the live scene state, the persona's ground-truth preferences, and the dialogue history. It answers preference questions concisely in first person, deflects questions about object location or availability (encouraging the agent to search), and stays consistent with its prior answers. The simulator only answers questions; ADAPT episodes have no satisfaction check or unprompted correction (cf. user termination in CondAmbigQA).

Template 32: ADAPT User Simulator System Prompt

```
You are teaching a household assistive robot in performing various assistive
    tasks in a manner {persona_name} would like. The robot may not know {
    persona_name}'s preferences, so your job is to guide the robot to perform
     the given task for {persona_name}. Be sure to guide the robot to make
    only those dishes that the task calls for, e.g. if the task is to make a
    waffle do not ask the robot to make other things, such as coffee. Answer
    direct questions regarding your preferences, and not the availability or
    location of objects. In the latter case, encourage the robot to search
    and explore different locations. Even if the robot makes an irreversible
```

```
    error, be sure to provide a correction so that the robot does not repeat
    it's mistakes the next time.

Given the current state of the house and what you know about {persona_name}
    and the task at hand, you will respond to the robot's last question
    concisely, and in first person, as if you are {persona_name}.

Environment State:
{environment_state}

Task: {task}
 {persona_name} has the following preferences.
{preference_1}
{preference_2}
{...}

Look at the following interaction and provide a short answer to the robot's
    last question based on {persona_name}'s preferences. If {persona_name} is
     flexible in their preference, make a choice arbitrarily, but make sure
    to tell the robot that usually {persona_name} is flexible, and options
    which they would be okay with. Make sure to be consistent with your
    previous feedback.
```

Template 33: ADAPT User Simulator Message

```
[user] Action: Ask "{question_t}"

[assistant] {answer_t}

[user] Action: Ask "{question}"
```

J.3 Hypothesis Proposers and Consistency Scorers

The belief proposer Q_{LLM} and consistency scorer S_{LLM} (Section 3.1) are instantiated differently for the two belief representations of Appendix I.2. In both, the scorer realizes either *batch rescoring* (the full conversation is rescored each turn) or *incremental update* (per-turn scores are accumulated, with retained particles scoring only new turns and fresh particles replaying history).

Joint belief. The proposer generates the candidate preference *profiles* (the particle set), regenerated each turn with the current belief as context so high-probability profiles are refined and unlikely ones dropped:

Template 34: ADAPT Joint-belief Hypothesis Proposer System Prompt

```
You are an expert at inferring user food preferences from conversation. Given
     a breakfast task and the user's Q&A responses so far, generate 6
    distinct candidate preference profiles that could explain this user's
    responses. Each profile is a short bulleted list of -38 specific
    preference statements about food preparation, ingredients, or serving
    style. Make the profiles meaningfully diverse -they should represent
    different plausible preference patterns, not minor variations of the same
     profile.
Respond ONLY with a valid JSON array of exactly 6 strings. Each string is one
     complete preference profile with bullet points using '-'.
```

```
Reference preference profiles from known users in this domain (use these to
    understand the preference vocabulary and relevant dimensions):
{example_profiles}
```

Template 35: ADAPT Joint-belief Hypothesis Proposer Message

```
Task: {task}

Conversation so far:
Q: {question_t}
A: {answer_t}

Current candidate profiles (most likely first).
Drop profiles with very low probability; refine or extend the rest based on
    new evidence:
  1. Weight: 45% | {profile_1}
  2. Weight: 30% | {profile_2}
  3. Weight: 25% | {profile_3}

Generate 6 diverse preference profiles as a JSON array of strings.
```

The scorer rates the holistic consistency of the whole conversation with each candidate profile on a 0–10 scale, forming the particle weights w^n:

Template 36: ADAPT Joint-Belief-Dialogue Consistency Scoring System Prompt

```
You are a dialog consistency judge for personalised breakfast task planning.
    Given a candidate preference profile and a conversation, rate on 0-10 how
     consistent ALL of the user's responses are with the profile.

Evaluation steps:
1. Read ALL questions and responses in the conversation.
2. Check whether each response aligns with what someone holding this profile
    would say.
3. Penalise any response that contradicts the profile.
4. Reward responses that confirm or elaborate on the profile.

Score:
  10 = all responses strongly consistent with this profile
   5 = responses are neutral (neither confirm nor deny)
   0 = one or more responses contradict this profile
Use the full range 0-10 (e.g. 3, 7, 8.5).
Respond ONLY with JSON: {"score": <number 0-10>, "reason": "<one sentence>"}

Reference preference profiles from known users in this domain (use these to
    understand the preference vocabulary and relevant dimensions):
{example_profiles}
```

Template 37: ADAPT Joint-Belief-Dialogue Consistency Scoring Message

```
Task context: {task}

Candidate preference profile:
{candidate_profile}
```

```
Conversation so far:
Q: {question_t}
A: {answer_t}

Question asked: {question}
User's response: {user_response}

How consistent are ALL of the user's responses with the above preference
    profile?
```

Factored belief. The proposer generates the atomic preferences γ_m over which independent Bernoulli beliefs p_m are maintained, covering the full breakfast experience and regenerated (belief-guided) each turn:

Template 38: ADAPT Factored-belief Atom Proposer System Prompt

```
You are an expert at identifying the preference dimensions that matter for a
    personalised breakfast experience. Given a task and a kitchen scene,
    generate atomic preference items that cover the FULL end-to-end breakfast
     meal -not only the specific dish named in the task, but every aspect of
    the meal the user might care about. Think across ALL relevant categories,
     including but not limited to:
  - Cooking method / style (e.g. scrambled vs fried, toasted vs soft)
  - Cooking medium (e.g. butter vs oil vs cooking spray)
  - Seasoning & flavour (e.g. salt, pepper, hot sauce, herbs)
  - Condiments & toppings (e.g. ketchup, jam, syrup, cheese, cream)
  - Side dishes (e.g. toast, fruit, hash browns, bacon)
  - Beverage (e.g. coffee, tea, juice, milk, water)
  - Dietary restrictions (e.g. avoids dairy, gluten-free, vegan)
  - Serving style (e.g. plated together, warm plate, garnished)
  - Portion / quantity (e.g. one egg vs two, small vs large portion)
Each atom is a short binary statement -the user either has it or doesn't (e.g.
     'prefers butter over oil', 'drinks coffee with milk', 'avoids dairy').
    Generate exactly 16 atoms. Respond ONLY with a valid JSON array of
    strings, one atom per string.

Reference preference profiles from known users in this domain (use these to
    understand the preference vocabulary and relevant dimensions):
{example_profiles}
```

Template 39: ADAPT Factored-belief Atom Proposer Message

```
Task: {task}

Scene layout:
{scene_layout}

Conversation so far:
Q: {question_t}
A: {answer_t}

Current atom inclusion probabilities (refine atoms with very low or very high
     probability; add new ones if the conversation reveals uncovered
    dimensions):
  - {atom_1} (p=0.72)
  - {atom_2} (p=0.31)
```

```
Generate a JSON array of atomic preference items covering the full breakfast
    experience for this task and scene.
```

Under *batch rescoring*, all atoms are scored jointly against the conversation in one call, each 0–10 score mapped to a log-likelihood-ratio update of p_m:

Template 40: ADAPT Belief Atom-Dialogue Consistency Batch Scoring System Prompt

```
You are a Bayesian preference judge for personalised breakfast task planning.
     Given a list of preference atoms and a user's full conversation, score
    each atom on 0-10 reflecting how likely the user HAS that preference
    given all their responses:

  10 = user clearly has this preference (strong evidence)
   5 = conversation is uninformative about this preference
   0 = user clearly does not have this preference (strong counter-evidence)

Use the full range (e.g. 2, 6, 8.5). Consider ALL responses, not just the
    latest.
Respond ONLY with a JSON object -one entry per atom, keys must match exactly:
{"atom text": <0-10>, ...}

Reference preference profiles from known users in this domain (use these to
    understand the preference vocabulary and relevant dimensions):
{example_profiles}
```

Template 41: ADAPT Belief Atom-Dialogue Consistency Batch Scoring Message

```
Task context: {task}

Conversation so far:
Q: {question_t}
A: {answer_t}

Question asked: {question}
User's response: {user_response}

Preference atoms to score:
- {atom_1}
- {atom_2}
- {...}

Return JSON {"atom": <score 0-10>} for every atom.
```

Under *incremental update*, each atom is scored independently (one call per atom) so the assessment of one preference is not influenced by the others:

Template 42: ADAPT Belief Atom-Dialogue Consistency Individual Scoring System Prompt

```
You are scoring the consistency of a user's conversation with a specific
    preference.
```

```
Given the conversation between a robot assistant and the user, rate on 0-10
    how consistent everything the user has said is with the user HAVING this
    preference:

  10 = the conversation is highly consistent with the user having this
      preference
   5 = the conversation is neutral / uninformative about this preference
   0 = the conversation is highly inconsistent with the user having this
       preference

Use the full range (e.g. 2, 6, 8.5). Consider ALL of the user's responses.
Respond ONLY with JSON: {"score": <0-10>}

Reference preference profiles from known users in this domain (use these to
    understand the preference vocabulary and relevant dimensions):
{example_profiles}
```

Template 43: ADAPT Belief Atom-Dialogue Consistency Individual Scoring Message

```
Task context: {task}

Conversation so far:
Q: {question_t}
A: {answer_t}

Question asked: {question}
User's response: {user_response}

Preference: "{atom}"

Return JSON {"score": <0-10>}.
```

J.4 Question Generator π^{ask}

Candidate clarifying questions are sampled from a prompt containing the task, scene layout, dialogue history, and the current belief summary — the candidate profiles (joint) or the atoms with their inclusion probabilities (factored). The system prompt is shared; the two belief modes differ only in how the belief summary is rendered in the user turn.

Template 44: ADAPT Question Proposal System Prompt

```
You are an expert at eliciting user preferences for personalised breakfast
    preparation. Your goal is to ask ONE targeted question that will best
    reveal the user's preferences relevant to the current task. The question
    should be natural and conversational -something a helpful robot assistant
     would ask. Avoid yes/no questions; prefer open-ended questions that
    reveal specific ingredient or preparation preferences. Do NOT ask about
    things you already know from prior answers. Respond with ONLY the
    question text, no preamble.
```

Template 45: ADAPT Joint-belief-based Question Proposal Message

```
Task: {task}
```

```
Scene layout:
{scene_layout}

Current belief about user type:
Inferred {K} candidate preference profiles.


Reference preference profiles from known users in this domain (use these to
    understand the preference vocabulary and relevant dimensions):
{example_profiles}

Candidate user preference profiles:
  profile_1 (p={p_1}):
    - {preference_1}
    - {preference_2}
  profile_2 (p={p_2}):
    - {preference_1}
    - {preference_2}

Previous Q&A:
  Q: {question_t}
  A: {answer_t}

What ONE clarifying question should you ask to best understand the user's
    preferences for this task?
```

Template 46: ADAPT Factored-belief-based Question Proposal Message

```
Task: {task}

Scene layout:
{scene_layout}

Current belief about user type:
{K} preference atoms inferred with independent Bernoulli beliefs.


Reference preference profiles from known users in this domain (use these to
    understand the preference vocabulary and relevant dimensions):
{example_profiles}

Inferred preference atoms (the user's true profile is an unknown subset of
    these):
  - {atom_1} (p={p_1})
  - {atom_2} (p={p_2})
  - {...}
Ask a question that resolves uncertainty about which atoms the user has.

Previous Q&A:
  Q: {question_t}
  A: {answer_t}

What ONE clarifying question should you ask to best understand the user's
    preferences for this task?
```

J.5 VoI Lookahead: Response Forecasting

To estimate V^{ask}, the agent samples hypotheses from the belief (all profiles in the joint variant; sampled atom compositions in the factored variant), forecasts each one's likely

answer to a candidate question by role-playing a user holding those preferences, then rescores and re-evaluates the goal reward on the updated belief:

Template 47: ADAPT Planning-time User Response Forecast System Prompt

```
You are teaching a household assistive robot in performing various assistive
    tasks in a manner the user would like. The robot may not know the user's
    preferences, so your job is to guide the robot to perform the given task
    for the user. Be sure to guide the robot to make only those dishes that
    the task calls for, e.g. if the task is to make a waffle do not ask the
    robot to make other things, such as coffee. Answer direct questions
    regarding your preferences, and not the availability or location of
    objects. In the latter case, encourage the robot to search and explore
    different locations. Even if the robot makes an irreversible error, be
    sure to provide a correction so that the robot does not repeat it's
    mistakes the next time.

Given the current state of the house and what you know about the user and the
     task at hand, you will respond to the robot's last question concisely,
    and in first person, as if you are the user.

Task: {task}
 The user has the following preferences.
{preference_1}
{preference_2}
{...}

Look at the following interaction and provide a short answer to the robot's
    last question based on the user's preferences. If the user is flexible in
     their preference, make a choice arbitrarily, but make sure to tell the
    robot that usually the user is flexible, and options which they would be
    okay with. Make sure to be consistent with your previous feedback.
```

Template 48: ADAPT Planning-time User Response Forecast Message

```
[user] Action: Ask "{question_t}"

[assistant] {answer_t}

[user] Action: Ask "{question}"
```

J.6 Goal-Reward Pipeline (Planner–Simulator–Proxy Judge)

The expected goal reward $\bar{G}(s, b, a^{\text{act}})$ — which serves as the elicitation stop value V^{act} and the quantity re-evaluated inside the VoI lookahead — is computed by the modular planner–simulator–proxy judge pipeline of Appendix I.2, conditioned on the full belief rather than a single committed profile. In the reported variants:

Planner. Generates a concrete numbered action plan conditioned on the belief summary:

Template 49: ADAPT Stage-1 Planner System Prompt

```
You are an expert planner for a personalised household-task robot. Given a
    task, a scene description, and a user's known preferences, produce a
    concrete numbered action plan the robot should execute to best satisfy
    those preferences. List only physical actions (Move, Pick-up, Pour, Turn
```

```
    on, etc.). Do not include commentary or explanations -numbered actions
    only.
```

Template 50: ADAPT Stage-1 Planner Message

```
Task: {task}

Scene layout:
{scene_layout}

Preferences for the inferred user profile:
- {belief_user_info}

Write the numbered action plan the robot should follow.
```

Simulator. Deduces the plan's final scene state as structured facts (entities created, items mixed and cooked, serving order, final locations) without stepping the true environment:

Template 51: ADAPT Planning-time Environment Simulator System Prompt

```
You are an expert at simulating the physical effects of household-robot
    action plans.

Given a task, a scene description, and a numbered robot action plan, deduce
    the final state of the scene after the plan completes.

Output ONLY a single JSON object with these keys:
  entities_created -list of objects *created* (assembled or cooked) during
      the plan
  mixed_together -list of groups; each group is a list of objects combined
      together
  cooked -list of objects that were cooked (placed on heat source + heated)
  used -list of all objects the plan physically interacted with
  serving_order -list of finished dishes/drinks in the order they were served
  final_locations -object →location at plan end (include all moved objects)
  transformations -source_object →resulting_entity for any cooking/assembly
      steps

Use the exact object and location names from the scene description. If a
    field has no entries, use an empty list or object. No explanations -JSON
    only.
```

Template 52: ADAPT Planning-time Environment Simulator Message

```
Task: {task}

Scene layout:
{scene_layout}

Action plan:
{plan_text}

Deduce the final scene state as JSON.
```

Proxy Judge. Evaluates each preference as satisfied, violated, or inapplicable given the deduced state. The per-preference outcomes are aggregated into $\bar{G}$; under the factored belief this is the mean-field weighted PSR of Appendix I.2, with each preference weighted by its inclusion probability p_m.

Template 53: ADAPT Proxy Judge System Prompt

```
You are an expert evaluator for personalised household-task robots.

Given a task, a deduced final scene state (JSON), and a numbered list of user
     preferences, decide whether each preference is satisfied, violated, or
    inapplicable given the final state.

Output ONLY a JSON array with one object per preference:
[
  {"preference": "<exact preference text>", "outcome": "satisfied|violated|
      inapplicable", "reason": "<1 sentence>"}
]

Definitions:
  satisfied -the final state fulfils this preference
  violated -the final state contradicts this preference
  inapplicable -the preference is irrelevant to this specific task/scene

Return one entry per preference in the same order. No extra commentary.
```

Template 54: ADAPT Proxy Judge Message

```
Task: {task}

Final scene state:
{
  "entities_created": [
    "{...}"
  ],
  "mixed_together": [
    "{...}"
  ],
  "cooked": [
    "{...}"
  ],
  "used": [
    "{...}"
  ],
  "serving_order": [
    "{...}"
  ],
  "final_locations": {
    "{object}": "{location}"
  },
  "transformations": {
    "{object}": "{entity}"
  }
}

User preferences:
1. {preference_1}
2. {preference_2}
3. {...}
```

```
Evaluate each preference.
```

J.7 Execution Policy π^{act}

In the execution stage the planner emits one action per step, conditioned on the full elicited belief (phrased as information gathered from the user). Generation is constrained by a scene-specific grammar rebuilt from the live environment each step, forcing syntactically valid actions over currently valid entities; the loop runs greedily until `Declare Done`. The system prompt carries the task, scene, belief, and action vocabulary and ends by asking for the next action with no `user`-role template. At the first step it is the only message; at later steps it is followed by the running rollout, with each prior action appended as an `Action: <action>` assistant turn and each environment result as an `Observation: <result> user` turn (omitted when a step yields no observation). The `user` turns thus carry only observations.

Template 55: ADAPT Stage-2 Action Generation System Prompt

```
You are an expert at task planning, and know how to provide assistance in a
    manner that {persona_name} wants for preparing and serving breakfasts
    such as making cereal, pancakes, toast, waffles, french toast, coffee,
    tea, etc. You have the ability to take the following actions:
- Open <X>: open an instance of an articulated furniture or object. e.g. Open
     cabinet
- Close <X>: close an instance of an articulated furniture or object. e.g.
    Close cabinet
- Heat <X>: heat a container, which is located on a heating appliance. e.g.
    Heat pan_0
- Turn on <X>: turn on an appliance. e.g. Turn on stove_0
- Turn off <X>: turn off an appliance. e.g. Turn off stove_0
- Search <X>: search a container in the house. e.g. Search counter_0
- Look for <X>: Look for an object in the whole house. Use this with a single
     word referring to the most generic category name of an object. Be sure
    to look for objects one-at-a-time. e.g. Look for milk
- Move <X> to <Y>: move an object X from wherever it currently is to the
    furniture or location Y. e.g. Move plate_0 to table_2
- Mix all items in <X> to get <Y>: mix items that exist in a container,
    typically to create a new entity. e.g. Mix all items in bowl_0 to get
    cake_batter
- Cook items in <X> to get <Y>: cook something in a stove, oven or other such
     appliance to create a cooked version of that entity. e.g. Cook items in
    pan_0 to get scrambled_eggs
- Chop <X> to get <Y>: chop items on a cutting board to create a chopped
    version of that entity. e.g. Chop apple_0 to get chopped_apple
- Pour <X> from <Y> to <Z>: pour an entity X from one container Y to another
    container Z. e.g. Pour milk from milk_carton_0 to mug_0
- Move <X> from <Y> to <Z>: move content X of container Y to another
    container Z. e.g. Move apple from apple_bag_0 to bowl_1
- <X> items in <Y> to get <Z>: freeform action to change object state, such
    as whisk, heat, blend, etc. e.g. whisk items in pan_0 to get custard,
    brew coffee_grounds to get brewed_coffee etc.
- <X> the object <Y> to get <Z>: freeform action to change object state, such
     as chop, peel, crack, wash, wipe, etc. e.g. chop the object tomato_1 to
    get finely_chopped_tomato, chop the object onion_0 to get sliced_onion,
    crack the object egg_4 to get cracked_egg, etc.
- Declare Done: indicate that the task is complete. Make sure to use this
    exactly once at the end of the task.

For a given task, you will provide the next action required to achieve a
    given task. e.g.
```

```
Action: Move apple_0 from counter_0 to table_0.Do NOT repeat your last action.

Note that you must do the task in a way that the user prefers. Think of
    different variations, modifications, sides, etc. applicable to the given
    task, and do the task in a way that you think the user would prefer.

From your conversation with the user, you have gathered the following:
{belief_user_info}

You will be performing tasks in a house with the following layout:
{scene_layout}

What is the next action required to achieve the task: {task}?
```

K ADAPT Parameters

K.1 Environment Roles

For comparability with baselines in (Patel et al., 2025), both the user simulator and REVOIR's modules — the hypothesis proposer Q_{LLM}, the consistency scorer S_{LLM}, the question generator π^{ask}, the planner–simulator–proxy judge that estimates $\bar{G}$, and the execution policy π^{act} — are all driven by a single model in our ADAPT experiments: `meta-llama/Llama-3.1-70B-Instruct`, served locally via a vLLM OpenAI-compatible endpoint. Unlike CondAmbigQA, there is no separate judge model required since the goal reward G = PSR is computed by the benchmark's programmatic per-persona preference predicates against the final state scene.

K.2 Benchmark Protocol

- **Personas and tasks:** 16 personas (12 seen / 4 held-out per split) and 8 breakfast tasks, evaluated under 4-fold cross-validation; we report means and standard deviations across split means.
- **Evaluation splits:** we reuse the benchmark's four-way split for comparability with the original ADAPT baselines.

K.3 Belief and Reward Settings

The four variants combine two belief representations with two update schemes (Appendix I.2); all share the goal-reward pipeline below.

- **Belief representation:** *joint* maintains K=6 profile particles; *factored* maintains independent Bernoulli beliefs over K=16 atomic preferences.
- **Goal reward $\bar{G}$:** estimated by the planner–simulator–proxy judge pipeline with an explicit generated plan and independent state-deduction/per-preference scoring.
- **Consistency scoring:** verbalized 0–10 scores, mapped to softmax weights (joint) or per-atom log-likelihood-ratio updates (factored).

K.4 Elicitation-Stage Parameters

- **Candidate questions:** 3 per turn, sampled and deduplicated by the best-of-K proposer.
- **VoI lookahead samples:** the joint variant enumerates all 6 profiles; the factored variant draws 4 atom compositions per candidate question.
- **Question cost:** since the original baselines are not cost-aware or question-throttled, we only impose a minimal flat per-question cost $C(m^{\text{ask}}) = 5 \times 10^{-4}$ subtracted from V^{ask} to avoid infinite episodes.

- **Stopping:** no hard question cap; elicitation only stops by the VoI rule $V^{\text{act}} \geq V^{\text{ask}} - C(m^{\text{ask}})$.

K.5 Execution-Stage Parameters

- **Decoding:** greedy decoding (`temperature_stage2: 0.0`) with grammar-constrained generation, the grammar rebuilt from the live scene each step.
- **Step budget:** unlimited; episodes end at `Declare Done`.

K.6 Sampling Temperatures

- **Question generation:** 0.8.
- **Mentalized response forecasting (VoI lookahead):** 0.7.
- **Planner plan generation:** 0.3.
- **Scoring and judging calls** (consistency scoring, state deduction, per-preference evaluation): 0.0.
- **Execution-stage action generation:** 0.0.