BALAR : A BAYESIAN AGENTIC LOOP FOR ACTIVE REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models increasingly operate in interactive settings where solving a task requires multiple rounds of information exchange with a user. However, most current systems treat dialogue reactively and lack a principled mechanism to reason about what information is missing. We propose **BALAR** (**B**ayesian **A**gentic **L**oop for **A**ctive **R**easoning), a task-agnostic outer-loop algorithm that requires no fine-tuning and enables multi-turn interaction between an LLM agent and a user. BALAR maintains a structured belief over latent states, selects clarifying questions by maximizing expected mutual information, and dynamically expands its state representation when the current one proves insufficient. We evaluate BALAR on three diverse benchmarks: AR-Bench-DC (detective cases), AR-Bench-SP (thinking puzzles), and iCraft-MD (clinical diagnosis). BALAR outperforms all baselines across the three benchmarks, with $\mathbf{14.6}\%$ higher accuracy on AR-Bench-DC, $\mathbf{38.5}\%$ on AR-Bench-SP, and $\mathbf{30.5}\%$ on iCraft-MD. We further study whether BALAR can serve as a teacher for a questioning policy through supervised fine-tuning (SFT), direct preference optimization (DPO), and dense-reward reinforcement learning (RL). Across 18 iCraft-MD replications, distilling BALAR into a Llama-8B yields relative gains in frozen-Qwen final-answer accuracy of $\mathbf{8.1}\%$ with SFT, $\mathbf{10.9}\%$ with DPO, and $\mathbf{12.1}\%$ with RL over the untuned policy.

1 INTRODUCTION

Modern deployments of large language models (LLMs) span domains where user intent is rarely unambiguous: a patient asking a medical AI for advice might omit critical symptoms, a customer service agent must resolve which product a user is asking about from a vague description. In all these cases, the bottleneck is not the reasoning capacity of the LLM, but the absence of a mechanism that (1) detects when a prompt is ambiguous, (2) formulates targeted clarifying questions, (3) integrates responses, and (4) decides when enough information has been gathered to commit to an answer.

Existing approaches address this challenge in different ways. Tree-of-Thoughts (ToT) (Yao et al., 2023) focuses on internal search over intermediate reasoning states using BFS/DFS to maintain multiple partial solutions, whereas Uncertainty-of-Thoughts (UoT) (Hu et al., 2024) relies on forward simulation of interaction trees to select follow-up questions. These procedures can be computationally costly, and neither maintains an explicit belief over latent user intent. Other methods such as `CollabLLM` (Wu et al., 2025) learn to select questions by fine-tuning on simulated trajectories, introducing costly training dependencies. The `MediQ Expert` system (Li et al., 2024) operates through a fixed pipeline of LLM calls and is task specific. Benchmarks such as AR-Bench (Zhou et al., 2025) and MediQ (Li et al., 2024) expose the gap: state-of-the-art LLMs at the time of their publication fall far below an oracle that has access to all private information. A broader discussion of active reasoning, interactive dialogue, dialogue state tracking, and policy distillation is in Appendix A.

We address this gap with **BALAR** (**B**ayesian **A**gentic **L**oop for **A**ctive **R**easoning), a *task-agnostic*, *training-free* Bayesian outer loop. The key insight is to model user intent as a latent discrete variable over a structured space of disambiguating dimensions where each dimension captures a facet of potential ambiguity (e.g. *severity level*). BALAR initializes this belief at "sleep time" (before interaction begins), then iteratively selects the unasked question with highest mutual information with the belief, updates the posterior via Bayes' rule upon receiving each response, and dynamically

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.

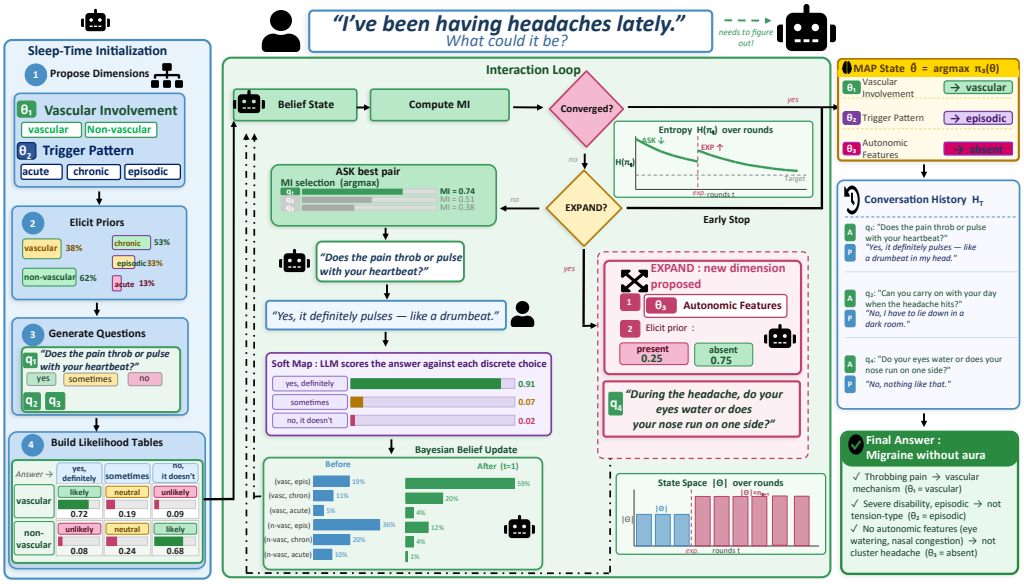


Figure 1: **BALAR overview.** Given an ambiguous query, BALAR performs multi-turn reasoning. *Sleep-time initialization* (left): the agent constructs a latent state representation by proposing disambiguating dimensions $\{\theta_j\}$, eliciting priors $\pi^{(j)}$, generating candidate questions $\mathcal{Q}$, and estimating likelihood tables $L^j(y \mid \theta_j; q)$. *Interaction loop* (center): the agent maintains a belief $\pi_t(\theta)$ and iteratively selects the unasked question q maximizing mutual information $I_t(\theta; Y \mid \mathcal{H}_t; q)$. User responses are incorporated via a Bayesian update. When the entropy gap cannot be closed within the remaining rounds, BALAR triggers EXPAND, introducing new dimensions and targeted questions. *Final answer* (right): once the belief concentrates, an LLM produces the final answer using the MAP state $\hat{\theta} = \arg\max_\theta \pi_T(\theta)$ and history $\mathcal{H}_T$.

expands the state space by proposing new dimensions when the current representation is insufficient. We illustrate the full pipeline in Figure 1 using a medical example.

Beyond using BALAR as an inference-time controller, we use it as a teacher to amortize reasoning into a questioning policy. BALAR provides belief states, selected questions, and information-theoretic supervision: SFT imitates BALAR's state updates and questions, DPO learns to prefer questions with higher mutual information, and RL rewards questions that reduce uncertainty over the final answer. The student generates questions directly at test time without executing the Bayesian loop.

Contributions. We make the following contributions:

- We propose BALAR, a task-agnostic, training-free Bayesian outer loop that enables LLM agents to engage in structured multi-turn interaction, actively selecting informative questions and updating a belief over latent task states during dialogue (Section 3).
- We study BALAR as a teacher for a questioning policy through supervised distillation of compact belief states, preference optimization over question information gain, and dense-reward reinforcement learning from answer-entropy reduction. SFT, DPO, and RL yield relative gains in final-answer accuracy of 8.1%, 10.9%, and 12.1%, respectively, over the untuned policy (Section 4.2).
- We evaluate BALAR on three benchmarks across open-weight LLMs. BALAR outperforms baselines by 14.6% on AR-Bench-DC, 38.5% on AR-Bench-SP, and 30.5% on iCraft-MD (Section 4.1).

2 Problem Setup

We consider an LLM agent interacting with a single user to resolve an ambiguous prompt. Generalization to the multi-user setting is in Appendix B, with results on AR-Bench-DC in Section 4.

Definition 2.1. An **interaction instance** is a tuple $(\mathbf{p}, \mathbf{c})$ with $\mathbf{p}$ an *ambiguous prompt* and $\mathbf{c}$ optional *meta-context* (publicly known background). A user holds *private information* not visible to the agent.

Let $y^\star$ denote the correct final answer to $\mathbf{p}$. The agent may ask questions $q \in \mathcal{Q}$ to the user, receiving a *natural language response* r. We treat this as a *Bayesian active information gathering* problem: the agent maintains a belief over a dynamic latent state θ representing user intent, selects actions to reduce uncertainty, and commits to an answer when sufficiently confident.

3 BALAR: Bayesian Agentic Loop for Active Reasoning

BALAR operates in two phases: a *sleep-time initialization* (Lin et al., 2025) that constructs a structured belief and a question bank from LLM calls, and an *interaction loop* that adaptively selects questions, updates the belief, and expands the state space as needed.

Running example. We trace a concrete instance: a patient submits the prompt *"I've been having headaches lately. What could it be ?"* This complaint could indicate migraine, tension headache, or hypertensive crisis, and the patient cannot be expected to use clinical terminology such as *vascular involvement* or *trigger pattern*. BALAR's goal is to infer these latent medical concepts from patient-friendly conversational questions, without the patient ever encountering medical jargon.

3.1 Sleep-Time Initialization

Given $(\mathbf{p}, \mathbf{c})$, the agent performs four initialization steps before interacting with the user.

Disambiguating dimensions. A single LLM call proposes p *disambiguating dimensions* $(\theta_1, \ldots, \theta_p)$, with each θ_j having a finite value set $\Theta_j = \{v_{j,1}, \ldots, v_{j,n_j}\}$. In the example of Figure 5, θ_2 is "trigger pattern", with values "acute", "chronic", and "episodic". The prompt queries for such disambiguating dimensions and their values by defining the dimensions as distinct unresolved factors whose value would narrow the answer. No task-specific knowledge base is pre-selected. We retain the same prompt structure across tasks, adapting only the domain-specific wording.

Prior elicitation. Our Bayesian approach relies on priors $\pi^{(j)}(v_{j,k})$. Given a finite label set $\mathcal{L}$, we enable an LLM to define these priors by prompting it to assign a prior label $\ell^{\mathrm{p}}_{j,k} \in \mathcal{L}$ to each value $v_{j,k}$ (see Figure 6). A fixed map ϕ maps each label to a positive, unnormalized weight for its corresponding value, so that $\pi^{(j)}(v_{j,k}) = \phi(\ell^{\mathrm{p}}_{j,k}) / \sum_{k'=1}^{n_j} \phi(\ell^{\mathrm{p}}_{j,k'})$. We choose $\mathcal{L} = \{\texttt{likely}, \texttt{neutral}, \texttt{unlikely}\}$ as a general-purpose three-level scale. The role-specific prompts are in Appendix V, and the label definitions are in Appendix J.1.

Question generation. A single LLM call generates $|\mathcal{Q}|$ candidate clarifying questions, each with a discrete answer set $\mathcal{Y}_q$ (see Figure 7). Questions are designed to be informative about the dimensions from Step 1. Note that the user provides free-form responses to these questions and is not shown the discrete answer set $\mathcal{Y}_q$. Instead, $\mathcal{Y}_q$ is used internally for tractable belief updates (see Section 3.4).

Likelihood table construction. A separate LLM call defines the *dimension-level likelihood matrix*:

$$L^j(y \mid \theta_j = v_{j,k}; q) \;=\; \frac{\phi(\ell^{\mathrm{l}}_{q,j,k,y})}{\sum_{y' \in \mathcal{Y}_q} \phi(\ell^{\mathrm{l}}_{q,j,k,y'})}, \quad \forall (v_{j,k}, y) \in \Theta_j \times \mathcal{Y}_q. \tag{1}$$

Each call handles exactly one question-dimension pair $(q, j) \in \mathcal{Q} \times [p]$ (see Figure 8) and jointly returns a likelihood label $\ell^{\mathrm{l}}_{q,j,k,y} \in \mathcal{L}$ for every $(v_{j,k}, y) \in \Theta_j \times \mathcal{Y}_q$. Thus, labels are elicited jointly within each dimension-level matrix and separately across pairs.

3.2 Structured Belief State

Under an a priori *independence assumption* across dimensions, the prior over the joint state $\theta = (\theta_1, \ldots, \theta_p) \in \Theta := \prod_{j=1}^p \Theta_j$ factorizes as $\pi_0(\theta) = \prod_{j=1}^p \pi^{(j)}(\theta_j)$. The *state-level likelihood* for question q is obtained by combining dimension-level likelihoods. For $\theta \in \Theta$ and $y \in \mathcal{Y}_q$:

$$K(y \mid \theta; q) = \frac{\prod_{j=1}^p L^j(y \mid \theta_j; q)}{\sum_{y' \in \mathcal{Y}_q} \prod_{j=1}^p L^j(y' \mid \theta_j; q)}. \tag{2}$$

This is a modeling choice that enables efficient computation. BALAR's deterministic inference layer maintains the global joint belief, and performs the Bayesian updates and information-theoretic calculations. The LLM supplies the qualitative labels used to construct priors, likelihoods, and soft observations, but it does not perform the probabilistic inference.

3.3 Information-Theoretic Question Selection

At round t, let $\pi_t(\theta) = \mathbb{P}(\theta \mid \mathcal{H}_t)$ denote the posterior over states given the interaction history $\mathcal{H}_t$. Its *belief entropy* is $\mathbb{H}(\pi_t) = -\sum_{\theta\in\Theta} \pi_t(\theta) \log \pi_t(\theta)$. The agent selects the unasked question with the highest *mutual information* (MI):

$$q_t^* \in \arg\max_{q\notin\mathcal{H}_t} I_t(\theta;\, Y \mid \mathcal{H}_t;\, q) = \arg\max_{q\notin\mathcal{H}_t} \left[\mathbb{H}(\pi_t) - \mathbb{H}(\theta \mid Y, \mathcal{H}_t; q)\right], \tag{3}$$

where $\mathbb{H}(\theta \mid Y, \mathcal{H}_t; q)$ denotes the *conditional entropy*:

$$\mathbb{H}(\theta \mid Y, \mathcal{H}_t; q) = \sum_{y\in\mathcal{Y}_q} \underbrace{\mathbb{P}(Y = y \mid \mathcal{H}_t; q)}_{p_t^q(y)} \mathbb{H}\big(\underbrace{\mathbb{P}(\theta \mid Y = y, \mathcal{H}_t; q)}_{\pi_t^{q,y}(\theta)}\big). \tag{4}$$

The conditional entropy calculation relies on the *answer predictive distribution* :

$$p_t^q(y) \;=\; \mathbb{P}(Y = y \mid \mathcal{H}_t;\, q) \;=\; \sum_{\theta\in\Theta} \pi_t(\theta) K(y \mid \theta; q), \tag{5}$$

and the posterior probability of θ assuming answer y is given to question q :

$$\pi_t^{q,y}(\theta) \;=\; \mathbb{P}(\theta \mid Y = y, \mathcal{H}_t;\, q) \;=\; \frac{K(y \mid \theta; q)\, \pi_t(\theta)}{\sum_{\theta'\in\Theta} K(y \mid \theta'; q)\, \pi_t(\theta')}. \tag{6}$$

Theoretical Guarantee. Assuming answers to distinct questions are conditionally independent given θ, BALAR's MI-maximizing selection within the Ask sub-loop satisfies $G_k \geqslant (1 - 1/e)G_k^*$, where G_k is the expected information gain of the greedy policy after k rounds and G_k^* is the information gain of the best fixed set of at most k candidate questions. The proof is in Appendix E.

3.4 Soft Bayesian Belief Update

The user provides a free-form natural language answer r_t, which need not coincide with any element of $\mathcal{Y}_{q_t^*}$ (see Figure 9). An LLM maps r_t to an answer label $\ell_y^{\mathrm{a}} \in \mathcal{L}$ for each choice $y \in \mathcal{Y}_{q_t^*}$, inducing a probability vector :

$$\hat{\omega}_y(r_t, q_t^*) \;=\; \frac{\phi(\ell_y^{\mathrm{a}})}{\sum_{y'\in\mathcal{Y}_{q_t^*}} \phi(\ell_{y'}^{\mathrm{a}})}, \qquad \forall y \in \mathcal{Y}_{q_t^*}. \tag{7}$$

The *effective per-state likelihood* under this soft observation is

$$\widehat{K}(r_t \mid \theta; q_t^*) \;=\; \sum_{y\in\mathcal{Y}_{q_t^*}} \hat{\omega}_y(r_t, q_t^*)\, K(y \mid \theta; q_t^*). \tag{8}$$

The posterior then updates as:

$$\pi_{t+1}(\theta) \;=\; \frac{\widehat{K}(r_t \mid \theta; q_t^*)\, \pi_t(\theta)}{\sum_{\theta'\in\Theta} \widehat{K}(r_t \mid \theta'; q_t^*)\, \pi_t(\theta')}. \tag{9}$$

3.5 Dynamic State Expansion

When the existing question bank is exhausted or its best available MI indicates insufficient progress within the remaining rounds, BALAR triggers an **EXPAND** step.

Entropy gap criterion. Let $\alpha \in (0, 1 - 1/|\Theta|]$. Define the target entropy as the entropy of a distribution that assigns probability $1 - \alpha$ to a single state and distributes the remaining mass α uniformly over the other $|\Theta| - 1$ states :

$$\mathbb{H}_\alpha = -(1-\alpha)\log(1-\alpha) - \alpha \log\Big(\tfrac{\alpha}{|\Theta|-1}\Big). \tag{10}$$

Equivalently, among all distributions satisfying $\max_{\theta\in\Theta} \pi(\theta) \geqslant 1 - \alpha$, this choice achieves the largest possible entropy. The gap between BALAR's current belief entropy and this target entropy is $\Delta_t = \max(0, \mathbb{H}(\pi_t) - \mathbb{H}_\alpha)$. This gap drives when we choose to expand the state space. In particular, if the gap exceeds the estimated information gain available from the current question bank over the remaining $T - t$ rounds, scaled by λ, BALAR triggers an EXPAND call, where T is the maximum number of rounds. Formally, the condition is

$$\Delta_t > \lambda I_t^*(T - t), \tag{11}$$

where $I_t^* = \max_{q \text{ unasked}} I_t(\theta; Y \mid \mathcal{H}_t; q)$. This condition has an interpretation in terms of the number of rounds required to close the entropy gap. To better understand this, suppose that, while the representation and question bank remain fixed, the conditional expected information gain of subsequent queries is no greater than the current best gain I_t^*. Then the expected cumulative reduction in entropy over k rounds satisfies

$$\mathbb{E}[\mathbb{H}(\pi_t) - \mathbb{H}(\pi_{t+k}) \mid \mathcal{H}_t] = \sum_{i=0}^{k-1} \mathbb{E}[I_{t+i}(\theta;\, Y \mid \mathcal{H}_{t+i}; q_i) \mid \mathcal{H}_t] \;\leqslant\; k\, I_t^*, \tag{12}$$

for any continuation using the current question bank. To reach the target entropy $\mathbb{H}_\alpha$ from the current entropy $\mathbb{H}(\pi_t)$ in expectation, the agent must therefore close a gap of $\Delta_t = \max(0,\, \mathbb{H}(\pi_t) - \mathbb{H}_\alpha)$, which requires at least $k^* = \lceil \Delta_t / I_t^* \rceil$ rounds under this assumption, with $k^* = \infty$ when $I_t^* = 0 < \Delta_t$. The EXPAND condition $\Delta_t > \lambda\, I_t^*\, (T - t)$ implies that $k^* > \lambda\,(T - t)$, i.e the projected number of rounds exceeds a λ-multiple of the remaining budget. With $\lambda = 1$, the current bank cannot close the expected gap within the horizon. Smaller λ makes expansion more aggressive while larger λ favors continuing to ask from the current bank.

EXPAND procedure. When triggered, BALAR:

1. Generates a new dimension θ_{p+1} via an LLM call conditioned on the current conversation history $\mathcal{H}_t$, and all existing dimensions $(\theta_j)_{j\in[p]}$.
2. Elicits a prior $\pi^{(p+1)}$ for the new dimension, conditioned on $\mathcal{H}_t$.
3. Expands the belief state under independence: $\pi_t^{\text{new}}(\theta, \theta_{p+1}) = \pi_t(\theta) \cdot \pi^{(p+1)}(\theta_{p+1})$.
4. Recomputes likelihood tables for all existing questions over the new dimension.
5. Generates new clarifying questions $\mathcal{Q}_{\text{new}}$ that target the new dimension θ_{p+1} and the m_{top} existing dimensions with the highest marginal entropy, where m_{top} is a hyperparameter.
6. Computes likelihood tables for the new questions $q \in \mathcal{Q}_{\text{new}}$ over all dimensions $(\theta_j)_{j\in[p+1]}$.

Theoretical Guarantee. The EXPAND action increases state entropy by representing uncertainty along a new coordinate before subsequent ASK rounds reduce it. Let $Y^\star$ denote the random variable for the correct final answer, and recall that $\theta_{1:p} = (\theta_1, \ldots, \theta_p)$ is the current representation, and θ_{p+1} is the proposed new dimension. The relevant value of θ_{p+1} is not its added entropy, but its residual task information $I(Y^\star; \theta_{p+1} \mid \theta_{1:p}, \mathcal{H}_t)$. Appendix F proves that this representation value is monotone and, under conditional independence of candidate dimensions given $Y^\star$, $\theta_{1:p}$, and $\mathcal{H}_t$, submodular. If the LLM acts as an η-approximate marginal-gain oracle, where $\eta \in (0, 1]$ measures proposal quality, k expansions recover at least a $(1 - e^{-\eta})$ fraction of the best k-dimension refinement.

3.6 CONVERGENCE AND FINAL ANSWER

Convergence criterion. BALAR uses two convergence conditions depending on whether the task exposes a discrete answer set $\mathcal{A}$.

Answer-probability convergence. For each dimension j, BALAR elicits a dimension-level answer distribution $P(a \mid \theta_j)$ over possible answers $a \in \mathcal{A}$. We take the product of these factors as an

unnormalized answer compatibility score. BALAR averages this score over the joint belief and then normalizes over answer options, obtaining the predictive answer belief at round t:

$$\hat{p}_t(a) = \frac{\sum_{\theta \in \Theta} \pi_t(\theta) \prod_{j=1}^{p} P(a \mid \theta_j)}{\sum_{a' \in \mathcal{A}} \sum_{\theta \in \Theta} \pi_t(\theta) \prod_{j=1}^{p} P(a' \mid \theta_j)}. \tag{13}$$

The loop terminates when $\max_{a \in \mathcal{A}} \hat{p}_t(a) \geqslant 1 - \alpha$.

Marginal-fraction convergence. When the answer is a free-form explanation with no fixed answer set, BALAR instead declares convergence when a β-fraction of dimensions are individually concentrated:

$$\frac{1}{p} \sum_{j=1}^{p} \mathbf{1}\left[\max_{\theta_j \in \Theta_j} \pi_t^{(j)}(\theta_j) \geqslant 1 - \alpha \right] \;\geqslant\; \beta, \tag{14}$$

where $\pi_t^{(j)}$ is the marginal of π_t for θ_j, and $\beta \in (0, 1]$ controls how many dimensions must converge before termination. $\beta = 1$ requires all dimensions to concentrate marginally. In both cases, the loop terminates when the total round budget T or the ASK-round budget T_{ask} is exhausted.

Final answer. The MAP state $\hat{\theta} = \arg\max_{\theta \in \Theta} \pi_T(\theta)$ is extracted. A final LLM call takes $\hat{\theta}$ as a structured disambiguation summary appended to the original prompt and conversation history $\mathcal{H}_T$ to produce the answer (see Figure 10).

Appendix B provides the full algorithm in Algorithm 1. Implementation details including atomic reasoning, parallelism and the verifier are discussed in Appendix C. Appendix I analyzes both LLM-calls complexity and the cost of the operations used for exact Bayesian inference.

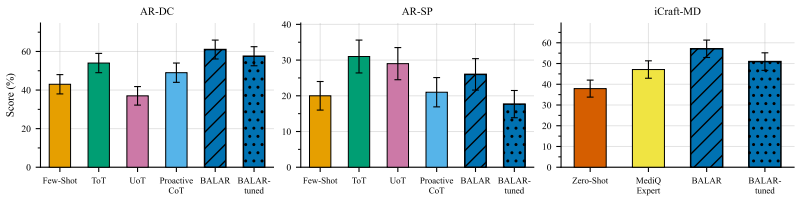
4 Results

We evaluate BALAR on AR-Bench-DC (detective reasoning), AR-Bench-SP (situation puzzles), and iCraft-MD (clinical diagnosis). Each task hides information the agent must elicit. In AR-Bench-DC, suspects hold private case information, and the agent must identify the murderer. In AR-Bench-SP, the agent must uncover the explanation known to the puzzle narrator. In iCraft-MD, patient-specific clinical facts are private to the patient, and the agent must choose the correct clinical answer.

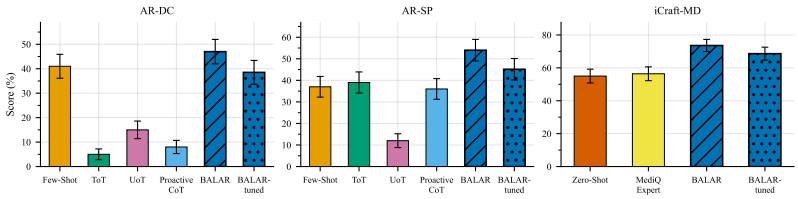
We use the same baselines as in the published benchmark papers: On AR-Bench (Zhou et al., 2025), we compare with Few-Shot, UoT, ToT, and Proactive CoT. On iCraft-MD (Li et al., 2024), we compare with Zero-Shot and MediQ Expert. Baseline selection follows the original benchmark studies: we use AR-Bench's reported comparison set for AR-Bench, and Zero-Shot plus MediQ Expert for iCraft-MD (Zhou et al., 2025; Li et al., 2024). We use the baselines' original implementations and prompts, with configurations reported as best in their benchmark papers, and allow up to 25 user-directed questions. Our agent models are Qwen2.5-7B/14B/32B-Instruct, QwQ-32B, and Llama-3.1-8B-Instruct. Qwen2.5-32B-Instruct serves as the user simulator. We also test SFT, DPO, and RL distillation on iCraft-MD, using Qwen2.5-32B-Instruct traces to train Llama-3.1-8B-Instruct. Full experimental details are in Appendix H.

4.1 BALAR Benchmark Results

Figure 2 reports outcome scores for the two main agent models : exact answer accuracy for AR-Bench-DC and iCraft-MD, and non-strict semantic equivalence for AR-Bench-SP (Kuhn et al., 2023). BALAR outperforms all baselines across all three benchmarks and both agent models, except in the case of AR-Bench-SP with `Llama-3.1-8B-Instruct`, where the smaller model may impact BALAR's structured prompting. Under `Qwen2.5-32B-Instruct`, BALAR achieves relative gains of $\mathbf{14.6}\%$ on AR-Bench-DC, $\mathbf{38.5}\%$ on AR-Bench-SP, and $\mathbf{30.5}\%$ on iCraft-MD over the strongest respective baselines. To contextualize this gain on iCraft-MD, we compare against an oracle that receives full patient information : BALAR with `Qwen2.5-32B-Instruct` without access to any private information achieves 73.6%, reaching 88% of the oracle upper bound of 83.6% (Appendix S). This suggests that BALAR's Bayesian formulation provides more robust performance than methods relying on prompt-driven or search-based heuristics, or task-tailored designs such as MediQ Expert.



(a) `Llama-3.1-8B-Instruct`



(b) `Qwen2.5-32B-Instruct`

Figure 2: *BALAR-tuned* denotes configurations selected based on average validation error over 500 randomly seeded 50/50 validation-test splits and evaluated by their average accuracy on the corresponding held-out test sets. All other methods use their full-dataset best configurations. Outcome is *exact accuracy* for AR-DC and iCraft-MD, and *non-strict semantic equivalence* for AR-SP. Error bars show binomial standard errors.

Held-out configuration selection. In Figure 2, results for BALAR are for its best configuration of $(\alpha, \beta, p, |\mathcal{Q}|)$, matching the procedure done in the compared-to published benchmark results. For each LLM model and dataset, we additionally compare to *BALAR-tuned*, where configurations are selected based on average validation error over 500 randomly seeded 50/50 validation-test splits and evaluated by their average accuracy on the corresponding held-out test sets. Comparing to the baseline results that retain the configuration-selection bias of the original benchmark papers (Zhou et al., 2025; Li et al., 2024), leading to an unfair comparison, *BALAR-tuned* remains ahead of the strongest baseline in four of six model–dataset comparisons.

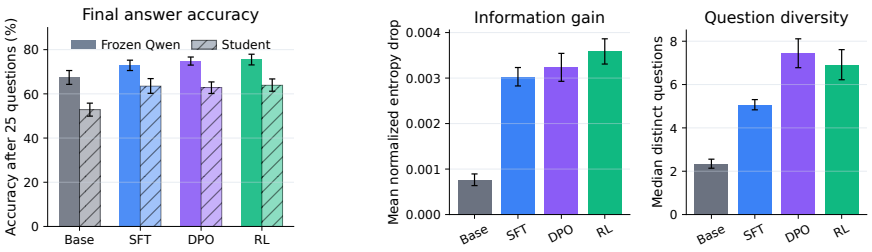
4.2 Distilling BALAR into a Questioning Policy

We next evaluate whether BALAR can teach `Llama-3.1-8B-Instruct` to ask useful questions without executing BALAR at test time. Figure 3a reports final-answer accuracy after 25 questions across 18 replications. With the frozen Qwen readout, mean accuracy increases from 67.5% for the base policy to 72.9% after SFT, 74.8% after DPO, and 75.6% after RL, corresponding to relative gains of 8.1%, 10.9%, and 12.1%, respectively. The student's own readout improves from 52.9% for the base policy to 63.5%, 62.9%, and 63.9%, respectively.

The dialogue statistics in Figure 3b show the same broad improvement. Mean normalized answer-entropy reduction per question is 0.00076 for the base policy, 0.00302 for SFT, 0.00323 for DPO, and 0.00360 for RL. The mean replication-level median number of distinct questions increases from 2.3 for the base policy to 5.1 for SFT, 7.4 for DPO, and 6.9 for RL. Thus all three trained policies ask a broader set of questions that, on average, causes a larger update in the frozen Qwen answer distribution. DPO produces the greatest question diversity, whereas RL produces the largest entropy reduction and highest mean frozen-Qwen accuracy.

4.3 Effect of Reasoning Mode

Table 1a reports the effect of replacing `Qwen2.5-32B-Instruct` with the reasoning-tuned `QwQ-32B` at T_{ask}=10. Switching models yields relative gains of 10.9% on AR-Bench-DC and 36.5% on AR-Bench-SP, while performance remains unchanged on iCraft-MD suggesting reasoning capacity matters more for deductive tasks than for clinical decision-making: structured hypothesis elimination benefits most from deeper reasoning, while clinical diagnosis appears less bottlenecked by reasoning depth than by information gathering.



(a) Final-answer accuracy.

(b) Information gain and question diversity.

Figure 3: BALAR distillation results on iCraft-MD. **(a)** Accuracy after 25 questions using either the frozen Qwen readout or the student's own readout. **(b)** Mean normalized answer-entropy drop per question and the median number of distinct questions over 25 turns. The 18 replications comprise three patient splits, both fold directions, and three training seeds. Error bars are 95% bootstrap intervals over the replication-level values.

4.4 Scaling with Model Size

To isolate the effect of model scale from hyperparameter choice, we fix the best hyperparameter configuration found for `Qwen2.5-32B-Instruct` and evaluate the *same configuration* on `Qwen2.5-7B-Instruct` and `Qwen2.5-14B-Instruct`. Table 1b shows that BALAR scales consistently with model size on iCraft-MD: accuracy improves from 50.7% (7B) to 62.1% (14B) to 73.6% (32B), suggesting BALAR scales with model capacity rather than saturating at smaller scales.

4.5 Ablation Studies

Table 1c isolates two key components of BALAR on iCraft-MD with `Qwen2.5-32B-Instruct`, using $T_{\text{ask}}=10$, $\alpha=0.1$, $p=5$, and $|\mathcal{Q}|=10$. For question selection, we compare : (i) random selection from the question bank, (ii) LLM-prompted selection ("which question should I ask next given the history?"), and (iii) MI maximization (ours). This isolates the value of the information-theoretic criterion from the rest of the framework. Replacing MI maximization with random or LLM-prompted selection drops accuracy by 23.3% and 9.6% respectively, confirming the value of the information-theoretic criterion. We also consider disabling EXPAND and fixing the state space and question bank to their initial values. Disabling EXPAND reduces accuracy by 13.7%, highlighting that the initial state space can be insufficient and BALAR benefits from dynamic expansion. Together, these results

Table 1: Effect of reasoning, model scale, question selection, and expansion on BALAR performance.

(a) Reasoning mode.

Dataset	**Qwen-32B**	**QwQ-32B**
AR-DC	46.0 ± 5.0	51.0 ± 5.0
AR-SP	52.0 ± 5.0	71.0 ± 4.5
iCraft-MD	71.4 ± 3.8	70.0 ± 3.9

(b) Model scaling.

Model	**iCraft-MD**
Qwen-7B	50.7 ± 4.2
Qwen-14B	62.1 ± 4.1
Qwen-32B	73.6 ± 3.7

(c) Question selection and expansion.

Variant	**Accuracy (%)**
MI (ours)	67.9 ± 3.9
Random sel.	52.1 ± 4.2
LLM sel.	61.4 ± 4.1
No expansion	58.6 ± 4.2

confirm that the gains reported in Section 4.1 are driven by the combination of principled question selection and adaptive state refinement, rather than by the Bayesian scaffolding alone. We additionally study sensitivity of the choice of prior mapping in Appendix M. Further analyses of prompt sensitivity, cross-model label consistency, question quality, dialogue state tracking, predictive-belief calibration, and answer-belief dynamics are reported in Appendices Q, J.2, N, T, O, and P, respectively.

4.6 Information Gain from Questions

We analyze how much uncertainty is removed by successive ASK actions. Figure 4 (bottom) plots the cumulative entropy reduction as a function of the number of questions asked under `Qwen2.5-32B-Instruct`. Let $\pi_{t_k}^{\text{old}}$ denote the belief state immediately before the k-th ASK update, and $\pi_{t_k}^{\text{new}}$ the posterior after incorporating the observed response. The entropy change at round k is $\Delta_k = \mathbb{H}(\pi_{t_k}^{\text{new}}) - \mathbb{H}(\pi_{t_k}^{\text{old}})$. We report the cumulative information gain $-\sum_{i=1}^{k} \Delta_i$, indexed by

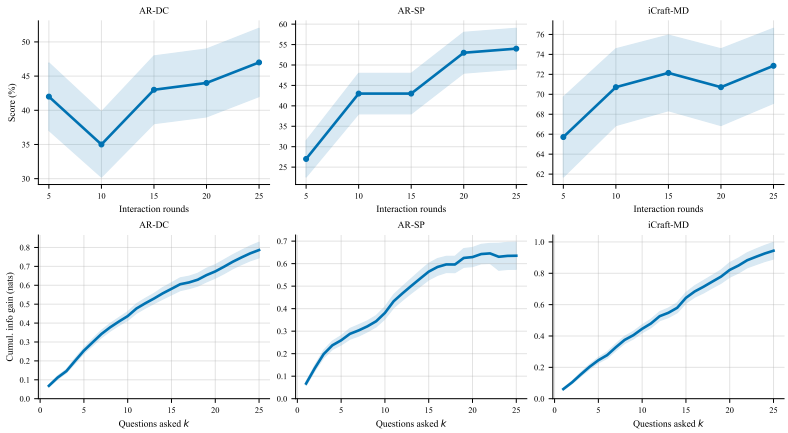


Figure 4: **Top:** BALAR score (%) as a function of interaction rounds $K \in \{5, 10, 15, 20, 25\}$. **Bottom:** Cumulative entropy reduction induced by successive ASK actions. Solid lines show the mean across runs, shaded regions denote standard errors. All results use `Qwen2.5-32B-Instruct`.

the number of questions asked k, which measures the total uncertainty removed by questions alone. Figure 4 (bottom) shows that it grows consistently with questions asked across all datasets, confirming that MI-based selection extracts meaningful uncertainty reduction at every round. AR-Bench-SP plateaus around k=20, suggesting diminishing returns once key dimensions are resolved, while AR-Bench-DC and iCraft-MD remain approximately linear throughout.

Figure 4 (top) shows BALAR accuracy as a function of the number of interaction rounds K under `Qwen2.5-32B-Instruct`, obtained by truncating the dialogue at round $K \in \{5, 10, 15, 20, 25\}$ and recomputing the final answer from the resulting partial history. BALAR improves monotonically with more rounds on AR-Bench-SP and iCraft-MD, demonstrating that additional questions consistently refine the belief toward the correct answer. On AR-Bench-DC, performance is less monotone, likely due to the multi-user setting. Further analysis is in Appendix R.

5 Conclusion

We presented BALAR, a training-free Bayesian outer loop that equips LLM agents with a principled mechanism to detect, track, and resolve prompt ambiguity through targeted multi-turn dialogue. The framework maintains a structured factored belief over a space of disambiguating dimensions, selects questions by mutual information maximization, and dynamically expands the state representation when the existing belief is insufficient. BALAR is task-agnostic, compatible with any instruction-following LLM, and its sleep-time initialization keeps per-turn interaction latency low. Its belief state and information-theoretic signals can train a smaller model to ask better questions without running the Bayesian loop at test time. More broadly, BALAR can serve as both an inference-time belief-state controller and a source of supervision for learned questioning policies. Our evaluations demonstrate BALAR's impact across diverse benchmarks.

Limitations. BALAR depends on the agent model's reliability in constructing and using structured representations, which may limit its benefits for less capable models. The independence assumption across dimensions introduces approximation error, and the functional form $K(\theta, y; q) \propto \prod_j L^j(y \mid \theta_j; q)$ is a modeling choice. Likelihood tables estimated by label-to-weight maps may be miscalibrated.[1] The method is most beneficial when the initial context is genuinely ambiguous.

Future work. Promising extensions include: (1) online calibration of likelihood tables using observed answers, directly targeting the mutual-information estimates rather than only the predictive answer belief evaluated in Appendix O, and (2) new forms of Bayesian distillation that transfer BALAR's structured states, posterior probabilities, and information-gain signals into learned policies.

[1] BALAR mitigates elicitation error in two ways: LLM calls are kept close to atomic, and an optional verifier checks each response for semantic validity (Appendix C). The components most exposed to this error like likelihood construction can also be delegated to a more capable LLM than the one driving the interaction.

6 AI USE STATEMENT

We used Large Language Models to correct grammar and flag stylistic issues in text initially written by the authors. We also used LLMs as coding assistants for infrastructure tasks, including bash scripts, job scheduling, and file transfer, and to polish plots for conference presentation. We used LLMs to check the proofs in Appendices E and F for errors. The mathematical claims and proofs were formulated and written by the authors, and any issues flagged by the LLMs were verified and corrected by the authors. We did not use LLMs to implement methods: the core BALAR implementation was written without LLM assistance. We did not use LLMs to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, support qualitative and thematic data analysis, or interpret results. Generating synthetic data sets, translation, and cleaning or reformatting datasets are not applicable to this work. LLMs used as components of BALAR, as user simulators, and as evaluation judges are part of the method and experimental protocol. The authors reviewed the AI-assisted material and take responsibility for the contents, claims, and artifacts in this paper.

7 ETHICS STATEMENT

This work evaluates existing AR-Bench and iCraft-MD benchmark cases using LLM-based user simulators. The experiments did not recruit human participants or involve live clinical interactions. The iCraft-MD experiments concern clinical cases, and their simulated results should not be interpreted as evidence of clinical safety or effectiveness. We use these cases only for the benchmark evaluation described in this paper.

8 REPRODUCIBILITY STATEMENT

The main text describes BALAR and the experimental protocol. The appendix provides the theoretical proofs, prompts, hyperparameters, and additional evaluation details, including the model configurations and the procedures for distillation. The source code is provided in the supplementary material. These materials are intended to support reproduction of the method.

REFERENCES

Marwa Abdulhai, Isadora White, Charlie Snell, Charles Sun, Joey Hong, Yuexiang Zhai, Kelvin Xu, and Sergey Levine. LMRL gym: Benchmarks for multi-turn reinforcement learning with language models. *CoRR*, 2023.

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.

Maximillian Chen, Xiao Yu, Weiyan Shi, Urvi Awasthi, and Zhou Yu. Controllable mixed-initiative dialogue generation through prompting. In *ACL*, 2023.

Maximillian Chen, Ruoxi Sun, Sercan Ö. Arik, and Tomas Pfister. Learning to clarify: Multi-turn conversations with action-based contrastive self-training. *CoRR*, abs/2406.00222, 2024.

Yizhou Chi, Jessy Lin, Kevin Lin, and Dan Klein. CLARINET: augmenting language models to ask clarification questions for retrieval. *CoRR*, abs/2405.15784, 2024.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Yang Deng, Wenqiang Lei, Wai Lam, and Tat-Seng Chua. A survey on proactive dialogue systems: Problems, methods, and prospects. In *IJCAI*, 2023a.

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. *arXiv preprint arXiv:2305.13626*, 2023b.

Pierre Erbacher and Laure Soulier. CIRCLE: multi-turn query clarifications with reinforcement learning. *CoRR*, abs/2311.02737, 2023.

Meiqi Guo, Mingda Zhang, Siva Reddy, and Malihe Alikhani. Abg-coqa: Clarifying ambiguity in conversational question answering. In *AKBC*, 2021.

Joey Hong, Sergey Levine, and Anca D. Dragan. Zero-shot goal-directed dialogue via RL on imagined conversations. *CoRR*, 2023.

E.J. Horvitz, D.E. Heckerman, B.N. Nathwani, and L.M. Fagan. Diagnostic strategies in the hypothesis-directed pathfinder system. In *Proceedings of the First Conference on Artificial Intelligence Applications*, pp. 630–636, Denver, Colorado, December 1984. IEEE Computer Society Press.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander J. Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *ArXiv*, abs/2305.02301, 2023. URL https://api.semanticscholar.org/CorpusID:258461606.

Zhiyuan Hu, Yue Feng, Anh Tuan Luu, Bryan Hooi, and Aldo Lipani. Unlocking the potential of user feedback: Leveraging large language model as user simulators to enhance dialogue system. In *CIKM*. ACM, 2023.

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 large language models. *arXiv preprint arXiv:2402.03271*, 2024.

Sedrick Keh, Justin Chiu, and Daniel Fried. Asking more informative questions for grounded retrieval. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 4429–4442, 2024.

Gangwoo Kim, Sungdong Kim, Byeongguk Jeon, Joonsuk Park, and Jaewoo Kang. Tree of clarifications: Answering ambiguous questions with retrieval-augmented large language models. In *EMNLP*, 2023.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*, 2023.

Wai-Chung Kwan, Xingshan Zeng, Yuxin Jiang, Yufei Wang, Liangyou Li, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. Mt-eval: A multi-turn capabilities evaluation benchmark for large language models. In *EMNLP*, 2024.

Stella Li, Vidhisha Balachandran, Shangbin Feng, Jonathan Ilgen, Emma Pierson, Pang Wei W Koh, and Yulia Tsvetkov. Mediq: Question-asking llms and a benchmark for reliable interactive clinical reasoning. *Advances in Neural Information Processing Systems*, 37:28858–28888, 2024.

Lizi Liao, Grace Hui Yang, and Chirag Shah. Proactive conversational agents in the post-chatgpt world. In *SIGIR*, 2023.

Kevin Lin, Charlie Snell, Yu Wang, Charles Packer, Sarah Wooders, Ion Stoica, and Joseph E Gonzalez. Sleep-time compute: Beyond inference scaling at test-time. *arXiv preprint arXiv:2504.13171*, 2025.

Fangwen Mu, Lin Shi, Song Wang, Zhuohao Yu, Binquan Zhang, Chenxue Wang, Shichao Liu, and Qing Wang. Clarifygpt: Empowering llm-based code generation with intention clarification. *CoRR*, abs/2310.10996, 2023.

Kevin Murphy. Agentic forecasting using sequential bayesian updating of linguistic beliefs. *arXiv preprint arXiv:2604.18576*, 2026.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

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. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Hossein A. Rahmani, Xi Wang, Yue Feng, Qiang Zhang, Emine Yilmaz, and Aldo Lipani. A survey on asking clarification questions datasets in conversational systems. In *ACL*, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Lior Shani, Aviv Rosenberg, Asaf B. Cassel, Oran Lang, Daniele Calandriello, Avital Zipori, Hila Noga, Orgad Keller, Bilal Piot, Idan Szpektor, Avinatan Hassidim, Yossi Matias, and Rémi Munos. Multi-turn reinforcement learning from preference human feedback. *CoRR*, abs/2405.14655, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Blaise Thomson. *Statistical Methods for Spoken Dialogue Management*. Springer Theses. Springer London, 2013. ISBN 978-1-4471-4922-4. doi: 10.1007/978-1-4471-4923-1. URL https://doi.org/10.1007/978-1-4471-4923-1.

Blaise Thomson and Steve Young. Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588, 2010. ISSN 0885-2308. doi: https://doi.org/10.1016/j.csl.2009.07.003. URL https://www.sciencedirect.com/science/article/pii/S0885230809000497.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

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 (ICML)*, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Michael J. Q. Zhang and Eunsol Choi. Clarify when necessary: Resolving ambiguity through interaction with lms. *CoRR*, abs/2311.09469, 2023.

Tong Zhang, Peixin Qin, Yang Deng, Chen Huang, Wenqiang Lei, Junhong Liu, Dingnan Jin, Hongru Liang, and Tat-Seng Chua. CLAMBER: A benchmark of identifying and clarifying ambiguous information needs in large language models. In *ACL*, 2024a.

Xuan Zhang, Yang Deng, Zifeng Ren, See-Kiong Ng, and Tat-Seng Chua. Ask-before-plan: Proactive language agents for real-world planning. *CoRR*, abs/2406.12639, 2024b.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn RL. In *ICML*, 2024.

Zhanke Zhou, Xiao Feng, Zhaocheng Zhu, Jiangchao Yao, Sanmi Koyejo, and Bo Han. From passive to active reasoning: Can large language models ask the right questions under incomplete information? In *ICML*, 2025.

APPENDIX CONTENTS

A Related Work **15**

B Multi-user extension **16**

C Practical Considerations **19**

D BALAR Illustrations **20**

E Near-Optimality of Greedy MI Maximization **21**

F Representation Value of Dynamic Expansion **22**

G Graphical Model **25**

H Experimental Setup **25**

I Computational Complexity **27**

J Likelihood labels **29**

K Semantic Equivalence Metric **30**

L Hyperparameters **30**

M Prior Sensitivity **31**

N Question Quality **31**

O Predictive-Belief Calibration **31**

P Answer-Belief Dynamics **32**

Q Sensitivity of Initial Dimensions **33**

R Round Distribution **33**

S Oracle **34**

T Dialogue State Tracking Evaluation **35**

U Distillation **36**

V Prompts **38**

A Related Work

Active reasoning benchmarks. In *active reasoning*, a model must iteratively acquire missing information through interaction rather than solving a problem from a fully specified prompt. The AR-Bench benchmark (Zhou et al., 2025) evaluates this capability by placing language models in multi-turn environments where they must ask informative questions to uncover hidden facts before producing a final answer. Empirical results reveal a substantial gap between passive and active reasoning performance: even state-of-the-art models achieve relatively low accuracy, and models frequently ask vague or redundant questions while struggling to accumulate useful information across turns. More broadly, related benchmarks evaluate multi-turn interaction and the handling of underspecified requests (Abdulhai et al., 2023; Kwan et al., 2024; Guo et al., 2021; Zhang et al., 2024a; Rahmani et al., 2023). These findings highlight the need for architectures that explicitly reason about uncertainty and guide question selection in a principled manner. BALAR addresses this by maintaining a structured posterior over latent user intent and selecting clarifying questions by maximizing expected information gain, providing a principled mechanism for strategic information gathering across interaction rounds.

Interactive medical dialogue. The MEDIQ framework (Li et al., 2024) introduces a benchmark for evaluating the ability of LLMs to proactively seek missing information in clinical decision-making tasks. MEDIQ converts existing datasets into interactive tasks by revealing only limited initial patient information and requiring the model to iteratively gather missing evidence before making a decision. Experiments show that prompting state-of-the-art LLMs to ask questions often *degrades* performance relative to answering directly with partial information, highlighting the difficulty of proactive information-seeking for current models. In contrast to task-specific pipelines such as MediQ Expert, BALAR provides a general-purpose Bayesian outer-loop that operates across domains without relying on specialized heuristics.

Search-based and uncertainty-aware reasoning. Tree-of-Thoughts (ToT) (Yao et al., 2023) extends chain-of-thought reasoning by organizing intermediate reasoning steps as a search tree, combining LM generation with BFS/DFS to maintain multiple partial solutions and backtrack when necessary. Uncertainty of Thoughts (UoT) (Hu et al., 2024) similarly relies on forward simulation of interaction trees, propagating information-theoretic rewards across hypothetical dialogue trajectories to select the question with highest expected uncertainty reduction. While both approaches improve reasoning through exploration, they incur substantial computational cost and do not maintain an explicit probabilistic model of the problem state. The idea of selecting questions by maximizing expected entropy reduction over a belief state has classical precedent in expert systems. Horvitz et al. (1984) introduced this strategy in the PATHFINDER system for lymph-node pathology diagnosis, where a probability distribution over diseases is updated after each observation. Related approaches also use model uncertainty or question informativeness to decide when to clarify (Keh et al., 2024; Zhang & Choi, 2023; Kim et al., 2023). BALAR maintains and updates a posterior belief over a latent space, selecting questions through mutual information. Although it also requires multiple LLM calls, these are decomposed into independent computations that can be executed in parallel, avoiding the sequential overhead of trajectory-based methods.

Proactive dialogue through learning and prompting. Proactive agents, which take initiative by asking clarifying questions or steering the dialogue, have been studied extensively (Deng et al., 2023a; Liao et al., 2023; Chen et al., 2023). CollabLLM (Wu et al., 2025) uses reinforcement learning to fine-tune LLMs to optimize collaboration outcomes, estimating a multiturn-aware reward by simulating future conversations with LLM-based user simulators (Hong et al., 2023; Hu et al., 2023). Proactive Chain-of-Thought (ProCoT) (Deng et al., 2023b) instead induces proactive behaviors through prompting, augmenting standard prompting with intermediate reasoning steps that describe the dialogue state and plan the next action. Similar prompting approaches elicit clarification questions for code generation, retrieval, and planning (Mu et al., 2023; Chi et al., 2024; Zhang et al., 2024b). STaR-GATE (Andukuri et al., 2024) takes a self-improvement approach: starting from a pretrained model, it iteratively fine-tunes on questions that increase the likelihood of high-quality task responses, bootstrapping better clarification-seeking behavior without requiring human-labeled trajectories. Both approaches demonstrate that models can be made to ask clarifying questions, but rely on either additional training or heuristic prompt engineering. BALAR requires

neither: its Bayesian formulation directly identifies informative queries from a maintained posterior, enabling principled proactive dialogue without fine-tuning or prompt heuristics. Other learning-based approaches train clarification policies through preference optimization or reinforcement learning (Chen et al., 2024; Erbacher & Soulier, 2023).

Dialogue state tracking and POMDP dialogue management. Dialogue state tracking (DST) maintains a belief over latent user goals and other dialogue variables as evidence accumulates over a conversation. In the classical POMDP formulation, this belief is the information state used by a dialogue policy to select system actions under uncertainty. The Bayesian Update of Dialogue State framework (BUDS) of Thomson & Young (2010) represents the dialogue state with a factored dynamic Bayesian network and performs tractable approximate updates using loopy belief propagation. It couples this tracker with a component-based policy learned using episodic Natural Actor Critic. Thomson (2013) gives a broader treatment of the resulting architecture, inference algorithms, parameter estimation, and policy optimization. BALAR inherits the central DST idea of maintaining a structured Bayesian belief over latent user state, but addresses a different representation and control regime. Classical BUDS assumes a domain-specific state structure and ontology fixed at design time, and learns a policy from a reward function and training dialogues. BALAR instead uses LLM calls to construct the dimensions, their value sets, candidate questions, and likelihood tables. It then selects questions through closed-form mutual information rather than a learned policy, and can add new dimensions and questions during interaction. Thus, classical BUDS tracks uncertainty within a predefined dialogue state, whereas BALAR also attempts to refine the state representation when its current question bank offers insufficient projected information gain.

Post-training and policy distillation. Supervised fine-tuning (SFT) on demonstrations is a standard way to teach language models to follow instructions and adopt task-specific behavior (Wei et al., 2021; Ouyang et al., 2022). Reasoning distillation extends this approach by training a smaller model on explanations or intermediate states produced by a larger teacher (Hsieh et al., 2023). Reinforcement learning from human feedback instead optimizes a reward, commonly with Proximal Policy Optimization (PPO) (Christiano et al., 2017; Schulman et al., 2017; Ouyang et al., 2022). Direct Preference Optimization (DPO) avoids an explicit reward-modeling and online RL stage by directly increasing the likelihood of preferred outputs relative to rejected ones (Rafailov et al., 2023). Group-relative policy optimization further replaces a learned critic with advantages computed by comparing rewards among outputs sampled for the same prompt (Shao et al., 2024). Multi-turn extensions of these methods optimize trajectory-level rewards over an entire interaction (Shani et al., 2024; Zhou et al., 2024). These methods usually optimize instruction following, response preference, or reasoning correctness. BALAR distillation instead targets active information gathering: SFT copies a compact Bayesian MAP state and BALAR-selected question, DPO prefers high over low mutual-information questions, and RL rewards the entropy reduction caused by each answer.

B MULTI-USER EXTENSION

The main text presents BALAR for a single user. This appendix gives the corresponding multi-user formulation, where the agent may ask a set $\mathcal{U}$ of users. Candidate actions are question-user pairs (q, u). Setting $|\mathcal{U}| = 1$ recovers the main-text formulation.

Disambiguating-dimension generation, prior elicitation, and question generation are unchanged from the main text. The multi-user extension first changes likelihood-table construction, which is performed for each question-user pair.

Likelihood table construction. A separate LLM call defines the *dimension-level likelihood matrix*:

$$L^j(y \mid \theta_j = v_{j,k}; q, u) = \frac{\phi(\ell^{\mathrm{l}}_{q,u,j,k,y})}{\sum_{y' \in \mathcal{Y}_q} \phi(\ell^{\mathrm{l}}_{q,u,j,k,y'})}, \quad \forall (v_{j,k}, y) \in \Theta_j \times \mathcal{Y}_q. \tag{15}$$

Each call handles one triple $(q, u, j) \in \mathcal{Q} \times \mathcal{U} \times [p]$ and returns a label $\ell^{\mathrm{l}}_{q,u,j,k,y} \in \mathcal{L}$ for every cell $(v_{j,k}, y)$.

Structured belief state. The factorized prior over dimensions is unchanged from the main text. The state-level likelihood for pair (q, u) is obtained by combining dimension-level likelihoods. For

$\theta \in \Theta$ and $y \in \mathcal{Y}_q$:

$$K(y \mid \theta; q, u) = \frac{\prod_{j=1}^{p} L^j(y \mid \theta_j; q, u)}{\sum_{y' \in \mathcal{Y}_q} \prod_{j=1}^{p} L^j(y' \mid \theta_j; q, u)}. \tag{16}$$

Information-theoretic question selection. The posterior π_t and belief entropy are as defined in the main text. For a candidate pair $(q, u) \in \mathcal{Q} \times \mathcal{U}$, the answer predictive distribution is:

$$p_t^{q,u}(y) = \mathbb{P}(Y = y \mid \mathcal{H}_t; q, u) = \sum_{\theta \in \Theta} \pi_t(\theta) K(y \mid \theta; q, u). \tag{17}$$

Upon observing answer y from pair (q, u), the posterior becomes:

$$\pi_t^{q,u,y}(\theta) = \frac{K(y \mid \theta; q, u)\, \pi_t(\theta)}{\sum_{\theta' \in \Theta} K(y \mid \theta'; q, u)\, \pi_t(\theta')}. \tag{18}$$

The conditional entropy after querying pair (q, u) is:

$$\mathbb{H}(\theta \mid Y, \mathcal{H}_t; q, u) = \sum_{y \in \mathcal{Y}_q} p_t^{q,u}(y)\, \mathbb{H}(\pi_t^{q,u,y}). \tag{19}$$

The agent selects the unasked pair (q_t^*, u_t^*) with the highest mutual information (MI):

$$I_t(\theta; Y \mid \mathcal{H}_t; q, u) = \mathbb{H}(\pi_t) - \mathbb{H}(\theta \mid Y, \mathcal{H}_t; q, u). \tag{20}$$

Soft Bayesian belief update. The selected user u_t^* provides a free-form response r_t to question q_t^*. The soft mapping $\hat{\omega}_y(r_t, q_t^*)$ remains as defined in the main text. The effective likelihood for the selected pair is:

$$\widehat{K}(r_t \mid \theta; q_t^*, u_t^*) = \sum_{y \in \mathcal{Y}_{q_t^*}} \hat{\omega}_y(r_t, q_t^*) K(y \mid \theta; q_t^*, u_t^*). \tag{21}$$

The posterior update retains the main-text form:

$$\pi_{t+1}(\theta) = \frac{\widehat{K}(r_t \mid \theta; q_t^*, u_t^*)\, \pi_t(\theta)}{\sum_{\theta' \in \Theta} \widehat{K}(r_t \mid \theta'; q_t^*, u_t^*)\, \pi_t(\theta')}. \tag{22}$$

Dynamic state expansion. The entropy-gap trigger is unchanged from the main text. In the multi-user setting, the best available MI is taken over unasked question-user pairs:

$$I_t^* = \max_{(q,u) \text{ unasked}} I_t(\theta; Y \mid \mathcal{H}_t; q, u), \qquad \Delta_t > \lambda I_t^* (T - t). \tag{23}$$

Suppose that, while the representation and question bank remain fixed, the conditional expected information gain of subsequent queries is no greater than the current best gain I_t^*. Then the expected cumulative reduction in entropy over k rounds satisfies

$$\mathbb{E}[\mathbb{H}(\pi_t) - \mathbb{H}(\pi_{t+k}) \mid \mathcal{H}_t] = \sum_{i=0}^{k-1} \mathbb{E}[I_{t+i}(\theta;\, Y \mid \mathcal{H}_{t+i}; q_i, u_i) \mid \mathcal{H}_t] \;\leqslant\; k\, I_t^*, \tag{24}$$

for any continuation using the current question bank.

The interpretation of this bound is unchanged from the main text. To reach the target entropy $\mathbb{H}_\alpha$ from the current entropy $\mathbb{H}(\pi_t)$ in expectation, the agent must close a gap of $\Delta_t = \max(0,\, \mathbb{H}(\pi_t) - \mathbb{H}_\alpha)$. This requires at least $k^* = \lceil \Delta_t / I_t^* \rceil$ rounds under the assumption above, with $k^* = \infty$ when $I_t^* = 0 < \Delta_t$. The EXPAND condition $\Delta_t > \lambda I_t^* (T - t)$ implies $k^* > \lambda (T - t)$.

The steps for proposing a new dimension, eliciting its prior, expanding the state under independence, and generating questions for it are unchanged from the main text. The likelihood-table steps are performed for every user. When θ_{p+1} is added, BALAR recomputes $L^{p+1}(y \mid \theta_{p+1} = v_{p+1,k}; q, u)$ for every existing question q and user u. For each new question $q \in \mathcal{Q}_{\text{new}}$, it computes $L^j(y \mid \theta_j = v_{j,k}; q, u)$ for all users u and dimensions $j \in [p+1]$.

Convergence criteria, round budgets, and final-answer generation remain as in the main text (Section 3.6). BALAR uses the same stopping rules and produces the final answer from the MAP state $\hat{\theta} = \arg\max_\theta \pi_T(\theta)$ and interaction history $\mathcal{H}_T$.

The full multi-user algorithm is given in Algorithm 1. The proofs for greedy MI selection and state expansion are in Appendices E and F, respectively.

Algorithm 1: Bayesian Agentic Loop for Active Reasoning (BALAR)

Input: $\mathbf{p}$ ambiguous prompt, $\mathbf{c}$ context, users $\mathcal{U}$, config $(p, |\mathcal{Q}|, \alpha, \beta, T, T_{\text{ask}}, \lambda, m_{\text{top}}, \phi, \mathcal{L})$
Output: Final answer $\hat{y}$

$\{\theta_j, \Theta_j\}_{j=1}^{p} \leftarrow$ PROPOSEDIMENSIONS$(\mathbf{p}, \mathbf{c})$
$\pi^{(j)} \leftarrow$ ELICITPRIOR$(\mathbf{p}, \mathbf{c}, \theta_j)$ for all j (parallel)
$\mathcal{Q} \leftarrow$ GENERATEQUESTIONS$(\mathbf{p}, \mathbf{c}, \{\theta_j\})$
$L_{q,u,j} \leftarrow$ ESTIMATELIKELIHOOD$(\mathbf{p}, \mathbf{c}, q, u, \theta_j)$ for all (q, u, j) (parallel)
$\pi_0 \leftarrow \bigotimes_{j=1}^{p} \pi^{(j)}$, $K_{q,u} \leftarrow$ BUILDSTATELIKELIHOOD$(L_{q,u,\cdot})$
$t \leftarrow 1$, $n_{\text{asked}} \leftarrow 0$, asked $\leftarrow \varnothing$
while $t \leqslant T$ ***and*** $n_{asked} < T_{ask}$ **do**
  if $\mathcal{A}$ *exists* ***and*** $\max_a \hat{p}_t(a) \geqslant 1 - \alpha$ **then**
    break
  end
  if $\frac{1}{p}\sum_{j=1}^{p} \mathbf{1}[\max_v \pi_{t-1}^{(j)}(v) \geqslant 1 - \alpha] \geqslant \beta$ **then**
    break
  end
  $I_{q,u} \leftarrow I_{t-1}(\theta; Y | \mathcal{H}_{t-1}; q, u)$ for all $(q, u) \notin$ asked
  if *no unasked* (q, u) **then**
    if *state cap reached* **then**
      break
    end
    $I_t^* \leftarrow 0$
  else
    $(q^*, u^*) \leftarrow \arg\max I_{q,u}$, $I_t^* \leftarrow I_{q^*,u^*}$
  end
  $\Delta_t \leftarrow \max(0, \mathbb{H}(\pi_{t-1}) - \mathbb{H}_\alpha)$
  if *no unasked pair* ***or*** $\Delta_t > \lambda I_t^* (T - t)$ **then**
    if *state cap reached* **then**
      break
    end
    $\theta_{p+1}, \Theta_{p+1} \leftarrow$ NEWDIMENSION$(\mathbf{p}, \mathbf{c}, \mathcal{H}_{t-1})$
    $\pi^{(p+1)} \leftarrow$ ELICITPRIOR$(\mathbf{p}, \mathbf{c}, \theta_{p+1}, \mathcal{H}_{t-1})$
    $\pi_{t-1} \leftarrow \pi_{t-1} \otimes \pi^{(p+1)}$, $p \leftarrow p + 1$
    $L_{q,u,p+1} \leftarrow$ ESTIMATELIKELIHOOD$(\cdot)$ for $q \in \mathcal{Q}, u \in \mathcal{U}$
    $\mathcal{Q}_{\text{new}} \leftarrow$ GENERATEEXPANDEDQUESTIONS$(\cdot)$
    $L_{q,u,j} \leftarrow$ ESTIMATELIKELIHOOD$(\cdot)$ for $q \in \mathcal{Q}_{\text{new}}, u \in \mathcal{U}, j \in [p]$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \mathcal{Q}_{\text{new}}$
    $K_{q,u} \leftarrow$ BUILDSTATELIKELIHOOD$(L_{q,u,\cdot})$
  else
    $r_t \leftarrow$ GETUSERANSWER(u^*, q^*)
    $\hat{\omega} \leftarrow$ SOFTMAP(r_t, q^*)
    $\widehat{K}_{q^*,u^*}(r_t \mid \theta) \leftarrow \sum_{y \in \mathcal{Y}_{q^*}} \hat{\omega}_y K_{q^*,u^*}(y \mid \theta)$
    $Z_t \leftarrow \sum_{\theta' \in \Theta} \widehat{K}_{q^*,u^*}(r_t \mid \theta') \pi_{t-1}(\theta')$
    $\pi_t(\theta) \leftarrow \widehat{K}_{q^*,u^*}(r_t \mid \theta) \pi_{t-1}(\theta) / Z_t$
    asked $\leftarrow$ asked $\cup \{(q^*, u^*)\}$, $n_{\text{asked}} \leftarrow n_{\text{asked}} + 1$
  end
  $t \leftarrow t + 1$
end
$\hat{\theta} \leftarrow \arg\max_\theta \pi_{t-1}(\theta)$
$\hat{y} \leftarrow$ FINALANSWER$(\mathbf{p}, \mathbf{c}, \mathcal{H}_{t-1}, \hat{\theta})$
return $\hat{y}$

C PRACTICAL CONSIDERATIONS

Atomic reasoning. Most LLM calls in BALAR are kept as close to atomic as possible: each call typically handles a single dimension, question, user, or value. This keeps prompts small and helps avoid the degradation that occurs when too many likelihood judgments are packed into one context window. In particular, if too many likelihood entries are evaluated jointly, the model tends to return nearly uniform likelihoods, which in turn drives mutual information toward zero throughout the loop.

There is, however, one deliberate exception: likelihood estimation is performed at the level of a *single* $(\text{user}, \text{question}, \text{dimension})$ triple, producing the full table over that dimension's values and the question's answer choices in one call. In principle, one could push this further and score each individual cell atomically, but this would be substantially more expensive in the number of parallel LLM calls. We therefore adopt this as a practical trade-off between granularity and computational cost.

Verifier. BALAR uses two layers of output checking. First, every LLM response is required to conform to a structured output schema (a Pydantic model). If the response does not parse into the required schema, the call is retried, up to a fixed maximum number of attempts. This handles formatting and structural failures such as malformed JSON, missing required fields, or invalid types.

An optional verifier LLM provides a second layer of checking. Unlike schema validation, which is purely syntactic and structural, the verifier is used to assess whether the response is logically coherent and semantically valid for the task. For example, it can flag issues such as inconsistent reasoning, or other task-level errors that still satisfy the formal schema. When the verifier rejects an output, it returns corrective feedback, and the model is queried again with that feedback appended to the prompt.

A lightweight version of this verification principle which is active in all current experiments is that every prompt explicitly asks the model to generate a short `reason` before committing to its output label or value. This chain-of-thought prior to the structured output encourages the model to surface its reasoning explicitly, making logical inconsistencies easier to detect and reducing labeling errors even without a full verification pass.

Parallelism. BALAR is implemented using asynchronous execution. Most LLM calls are written as `async` functions and are dispatched concurrently using `asyncio.gather`, with a semaphore controlling the maximum number of simultaneous API requests. This design is used not only during initialization (Steps 1–4), but also throughout the interaction loop for tasks such as likelihood-table construction, prior estimation, and expansion-related computations. As a result, large batches of independent LLM calls can run in parallel, so the dominant cost becomes the latency of the slowest request rather than the sum of all calls. This allows the system to maintain interactive latency even when many likelihood evaluations are required.

State space size. The joint state space $|\Theta| = \prod_j n_j$ grows exponentially with the number of dimensions. To keep inference tractable, BALAR enforces a configurable upper bound on the total number of states. Before triggering an `EXPAND` step, the algorithm checks whether adding another dimension would keep the total state space within this cap. If the expansion would exceed the limit, no further dimensions are introduced.

D BALAR ILLUSTRATIONS

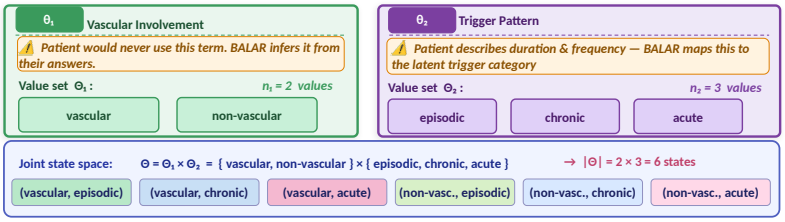


Figure 5: A single LLM call proposes two dimensions: θ_1 = *Vascular Involvement* (Θ_1 = {vascular, non-vascular}) and θ_2 = *Trigger Pattern* (Θ_2 = {episodic, chronic, acute}), yielding a joint state space of $|\Theta| = 6$ states.

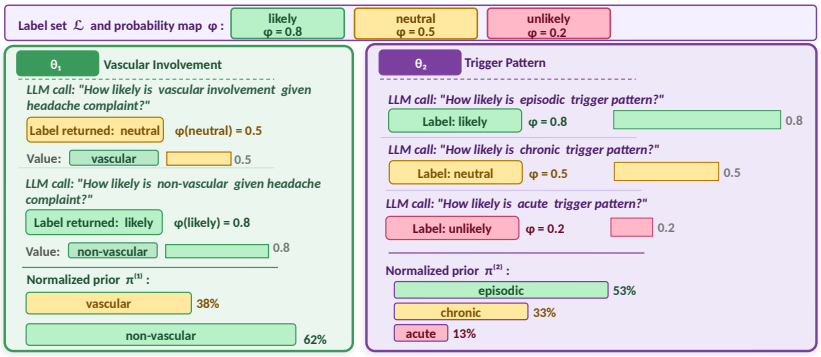


Figure 6: Parallel LLM calls assign a prior label $\ell^{\mathrm{p}} \in \mathcal{L}$ to each dimension value. Here, the LLM judges *vascular* as `neutral` and *non-vascular* as `likely`, yielding $\pi^{(1)} = [0.38, 0.62]$, while *episodic* is `likely`, *chronic* `neutral`, *acute* `unlikely`, giving $\pi^{(2)} = [0.53, 0.33, 0.13]$.

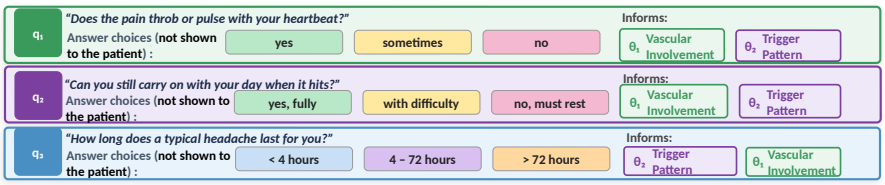


Figure 7: An LLM call generates $|\mathcal{Q}| = 3$ questions conditioned on the proposed dimensions. Each question informs many dimensions simultaneously. No question uses medical jargon.

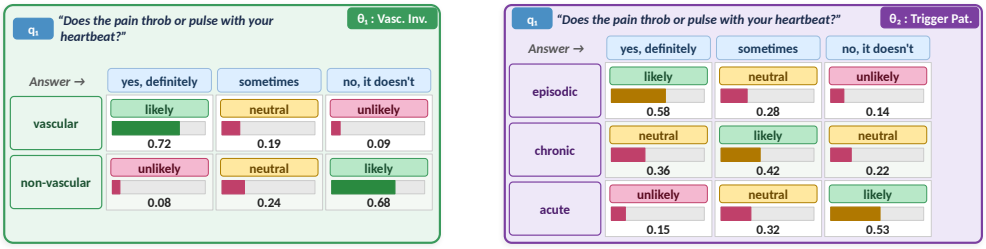


Figure 8: Six parallel LLM calls fill the $|\mathcal{Q}| \times p = 3 \times 2$ likelihood matrices. Shown here are the two tables for q_1: vascular states are much more likely to answer *"yes, definitely"* ($L = 0.72$) than non-vascular states ($L = 0.08$).

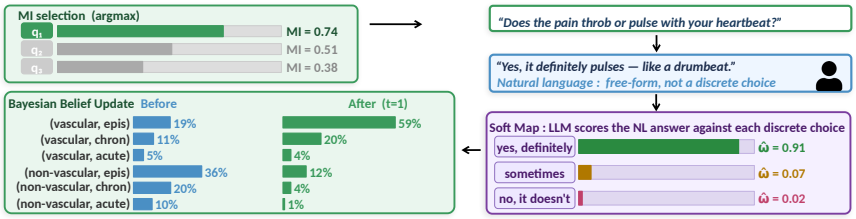


Figure 9: q_1 has the highest MI and is asked first. The patient replies in free-form natural language. The soft-map LLM assigns $\hat{\omega} = [0.91, 0.07, 0.02]$ over the discrete choices. A Bayesian update shifts mass toward vascular states: (vasc, epis) rises from 19% to 59%.

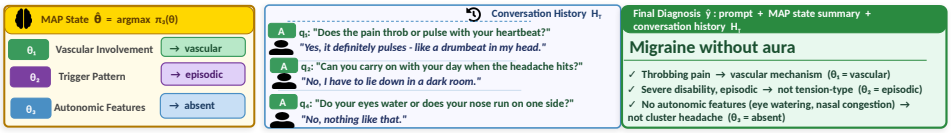


Figure 10: The MAP state $\hat{\theta} = (\text{vascular}, \text{episodic}, \text{absent})$, combined with the conversation history $\mathcal{H}_T$, conditions a final LLM call that produces the diagnosis *Migraine without aura*.

E Near-Optimality of Greedy MI Maximization

We analyze the multi-user version of BALAR, where each candidate action is a question-user pair $(q, u) \in \mathcal{Q} \times \mathcal{U}$. The single-user setting in the main text is recovered when $|\mathcal{U}| = 1$. Under conditional independence, BALAR's greedy question-selection recovers at least a $(1 - 1/e)$ fraction of the information gain of the best fixed set of at most k pairs in $\mathcal{Q} \times \mathcal{U}$.

Let $\theta \in \Theta$ be the latent state with prior π_0. Each *(question, user) pair* $(q, u) \in \mathcal{Q} \times \mathcal{U}$ yields an answer $Y_{q,u} \in \mathcal{Y}_q$ when queried. A history $\mathcal{H}_t = \{(q_i, u_i, Y_{q_i,u_i})\}_{i=1}^{t}$ records the pairs asked and answers received up to round t. We write $(q, u) \notin \mathcal{H}_t$ to mean the pair has not yet been asked. We make the following assumption throughout.

Assumption E.1. The answers $\{Y_{q,u}\}_{(q,u)\in\mathcal{Q}\times\mathcal{U}}$ are conditionally independent given θ.

The *greedy policy* π^g selects at each round

$$(q_t^*, u_t^*) \in \arg\max_{(q,u)\notin\mathcal{H}_t} I_t(\theta; Y_{q,u} \mid \mathcal{H}_t; q, u)\ ^{2},$$

which is exactly BALAR's ASK selection rule of Section 3.3. Let $\mathcal{H}_t^g$ denote the history generated by π^g after t rounds. Define

$$G_t := \mathbb{H}(\theta) - \mathbb{H}(\theta \mid \mathcal{H}_t^g), \qquad G_k^* := \max_{\substack{B\subseteq\mathcal{Q}\times\mathcal{U}\\ |B|\leqslant k}} I\big(\theta; \{Y_{q,u}\}_{(q,u)\in B}\big),$$

where G_k^* is the information gain of the best fixed set of at most k pairs.

Theorem E.2. *Under Assumption E.1, BALAR's greedy MI-maximising (q, u)-selection satisfies*

$$G_k \geqslant \big(1 - \tfrac{1}{e}\big)\, G_k^*,$$

recovering at least a $(1 - 1/e) \approx 63\%$ fraction of the cumulative information gain of the best fixed set of at most k pairs.

[2]We use the semicolon to separate conditioning variables from fixed parameters: $f_D(A \mid B; C)$ denotes quantity f indexed by D (e.g., the round index t), where $A \mid B$ is read as "A given B" in the probabilistic sense, and C denotes a fixed parameter or context not treated as a random variable (e.g., the query–user pair (q, u)).

Proof of Theorem E.2. We first establish an averaged inequality for the MI of a fixed pair.

Lemma E.3. *Under Assumption E.1, fix a realized history $\mathcal{H}_t = h_t$ and a set C of pairs not yet asked at h_t. For any pair (q, u) not yet asked and not in C,*

$$\mathbb{E}_{y_C \sim \mathbb{P}(Y_C \mid \mathcal{H}_t = h_t)} \left[I(\theta;\, Y_{q,u} \mid \mathcal{H}_t = h_t, Y_C = y_C;\, q, u) \right] \;\leqslant\; I(\theta;\, Y_{q,u} \mid \mathcal{H}_t = h_t;\, q, u),$$

where Y_C denotes the answers to pairs in C. The conditional mutual information on the left is averaged over Y_C given $\mathcal{H}_t = h_t$.

Proof. By Assumption E.1, $Y_{q,u}$ and Y_C are independent given $(\theta, \mathcal{H}_t = h_t)$. For each y_C,

$$\begin{aligned} I(\theta;\, Y_{q,u} \mid \mathcal{H}_t = h_t, Y_C = y_C;\, q, u) = \mathbb{H}(Y_{q,u} \mid \mathcal{H}_t = h_t, Y_C = y_C;\, q, u) \\ - \mathbb{H}(Y_{q,u} \mid \theta, \mathcal{H}_t = h_t;\, q, u). \end{aligned}$$

Taking expectation over Y_C given $\mathcal{H}_t = h_t$ and using that conditioning reduces entropy on average gives the stated inequality. □

Let B^* attain the maximum in G_k^*. Fix a realized history $\mathcal{H}_t^g = h_t^g$, and let $(q_i, u_i)_{i=1}^r$ be the pairs in B^* that have not yet been asked, where $r \leqslant k$. The answers to pairs in B^* that have already been asked are specified by $\mathcal{H}_t^g$. By the chain rule and Lemma E.3,

$$I\Big(\theta;\, \{Y_{q,u}\}_{(q,u)\in B^*} \,\Big|\, \mathcal{H}_t^g = h_t^g\Big) = \sum_{i=1}^{r} I(\theta;\, Y_{q_i,u_i} \mid \mathcal{H}_t^g = h_t^g, Y_{q_1,u_1}, \ldots, Y_{q_{i-1},u_{i-1}};\, q_i, u_i) \tag{25}$$

$$\leqslant \sum_{i=1}^{r} I(\theta;\, Y_{q_i,u_i} \mid \mathcal{H}_t^g = h_t^g;\, q_i, u_i) \tag{26}$$

$$\leqslant k \max_{(q,u)\notin h_t^g} I(\theta;\, Y_{q,u} \mid \mathcal{H}_t^g = h_t^g;\, q, u). \tag{27}$$

The inequality on the first line uses Lemma E.3. Taking expectations over the history, we get

$$G_k^* - G_t = \mathbb{E}[\mathbb{H}(\theta \mid \mathcal{H}_t^g)] - \mathbb{H}\big(\theta \mid \{Y_{q,u}\}_{(q,u)\in B^*}\big) \tag{28}$$

$$\leqslant \mathbb{E}_{h_t^g}\Big[I\Big(\theta;\, \{Y_{q,u}\}_{(q,u)\in B^*} \,\Big|\, \mathcal{H}_t^g = h_t^g\Big)\Big] \tag{29}$$

$$\leqslant k\, \mathbb{E}_{h_t^g}\Big[\max_{(q,u)\notin h_t^g} I(\theta;\, Y_{q,u} \mid \mathcal{H}_t^g = h_t^g;\, q, u)\Big] \tag{30}$$

$$= k\,(G_{t+1} - G_t), \tag{31}$$

Rearranging equation 31:

$$G_k^* \;\leqslant\; k\, G_{t+1} - (k-1)\, G_t,$$

which gives

$$G_k^* - G_{t+1} \;\leqslant\; \big(1 - \tfrac{1}{k}\big)\,(G_k^* - G_t).$$

So with $G_0 = 0$:

$$G_k^* - G_k \;\leqslant\; \big(1 - \tfrac{1}{k}\big)^k\, G_k^* \;\leqslant\; e^{-1}\, G_k^*,$$

The result of Theorem E.2 then follows.

This guarantee applies to the ASK sub-loop with a fixed pair space $\mathcal{Q} \times \mathcal{U}$. The EXPAND action changes both the latent representation and the pair space, so it is not covered by Theorem E.2. Appendix F gives a separate conditional result for the representation value of successful dimension expansion.

F REPRESENTATION VALUE OF DYNAMIC EXPANSION

The greedy guarantee in Appendix E is stated for the multi-user extension, where candidate actions are pairs (q, u). Here we give a separate, conditional characterization of EXPAND. The result formalizes the value of adding a missing dimension.

Fix an interaction history $\mathcal{H}_t$ and condition throughout on this history. Recall that $Y^\star$ denotes the correct final answer and $\theta_{1:p} = (\theta_1, \ldots, \theta_p)$ denotes BALAR's current latent representation. Consider

an idealized pool of candidate dimensions $\theta_{p+1}, \ldots, \theta_{p+m}$ not present in $\theta_{1:p}$. For $S \subseteq [m]$, write $\theta_{p+S} := (\theta_{p+j})_{j \in S}$ and define the *representation value*

$$F_t(S) := I(Y^\star; \theta_{p+S} \mid \theta_{1:p}, \mathcal{H}_t) = \mathbb{H}(Y^\star \mid \theta_{1:p}, \mathcal{H}_t) - \mathbb{H}(Y^\star \mid \theta_{1:p}, \theta_{p+S}, \mathcal{H}_t). \tag{32}$$

Thus $F_t(S)$ measures how much residual task uncertainty becomes representable after adding θ_{p+S}. This definition distinguishes the value of a dimension from the entropy it adds to the latent state: a high-entropy dimension can still be irrelevant to $Y^\star$.

Theorem F.1. *For every $S \subseteq [m]$ and $j \notin S$,*

$$F_t(S \cup \{j\}) - F_t(S) = I(Y^\star; \theta_{p+j} \mid \theta_{1:p}, \theta_{p+S}, \mathcal{H}_t) \geqslant 0. \tag{33}$$

Hence F_t is monotone. Suppose additionally that the candidate dimensions are conditionally independent given the target $Y^\star$, current representation $\theta_{1:p}$, and history $\mathcal{H}_t$. Then F_t is submodular. In particular, for $S \subseteq T \subseteq [m]$ and $j \notin T$,

$$I(Y^\star; \theta_{p+j} \mid \theta_{1:p}, \theta_{p+S}, \mathcal{H}_t) \geqslant I(Y^\star; \theta_{p+j} \mid \theta_{1:p}, \theta_{p+T}, \mathcal{H}_t). \tag{34}$$

Proof. The chain rule for mutual information gives

$$F_t(S \cup \{j\}) - F_t(S) = I(Y^\star; \theta_{p+j} \mid \theta_{1:p}, \theta_{p+S}, \mathcal{H}_t) \geqslant 0, \tag{35}$$

which proves monotonicity. For submodularity, let the right-hand side be $\Delta_j(S)$. Its entropy form is

$$\begin{aligned} \Delta_j(S) &= \mathbb{H}(\theta_{p+j} \mid \theta_{1:p}, \theta_{p+S}, \mathcal{H}_t) - \mathbb{H}(\theta_{p+j} \mid Y^\star, \theta_{1:p}, \theta_{p+S}, \mathcal{H}_t) && (36) \\ &= \mathbb{H}(\theta_{p+j} \mid \theta_{1:p}, \theta_{p+S}, \mathcal{H}_t) - \mathbb{H}(\theta_{p+j} \mid Y^\star, \theta_{1:p}, \mathcal{H}_t), && (37) \end{aligned}$$

where the second equality follows from conditional independence. If $S \subseteq T$, conditioning reduces the first entropy term, while the second term is unchanged. Therefore $\Delta_j(S) \geqslant \Delta_j(T)$. □

The marginal gain in equation 33 gives an exact criterion: θ_{p+j} is useful relative to the dimensions already added precisely when it carries positive residual target information. The next result shows that if a finite collection of missing dimensions resolves the target, at least one of them must carry a nontrivial share of that information.

Corollary F.2. *Assume the conditions of Theorem F.1. Let $D \subseteq [m]$ satisfy $\mathbb{H}(Y^\star \mid \theta_{1:p}, \theta_{p+D}, \mathcal{H}_t) = 0$. For any $S \subsetneq D$, let $r = |D \backslash S|$. Then*

$$\max_{j \in D \backslash S} I(Y^\star; \theta_{p+j} \mid \theta_{1:p}, \theta_{p+S}, \mathcal{H}_t) \geqslant \frac{\mathbb{H}(Y^\star \mid \theta_{1:p}, \theta_{p+S}, \mathcal{H}_t)}{r}. \tag{38}$$

Proof. By the assumption on D, $F_t(D) - F_t(S) = \mathbb{H}(Y^\star \mid \theta_{1:p}, \theta_{p+S}, \mathcal{H}_t)$. Submodularity gives

$$\begin{aligned} F_t(D) - F_t(S) &\leqslant \sum_{j \in D \backslash S} \big[F_t(S \cup \{j\}) - F_t(S)\big] && (39) \\ &= \sum_{j \in D \backslash S} I(Y^\star; \theta_{p+j} \mid \theta_{1:p}, \theta_{p+S}, \mathcal{H}_t). && (40) \end{aligned}$$

At least one of the r terms in the right-hand side is greater than or equal to their average. □

BALAR does not enumerate the idealized candidate pool or evaluate equation 33 exactly. It asks an LLM to propose a novel dimension. To state what would follow from a proposal mechanism of known quality, we abstract it as an approximate oracle.

Assumption F.3. At expansion step i, let $S_i \subseteq [m]$ index the dimensions already added. The proposal mechanism returns a random index $\widehat{J}_i \in [m] \backslash S_i$ such that

$$\mathbb{E}\Big[I(Y^\star; \theta_{p+\widehat{J}_i} \mid \theta_{1:p}, \theta_{p+S_i}, \mathcal{H}_t) \mid S_i\Big] \geqslant \eta \max_{j \notin S_i} I(Y^\star; \theta_{p+j} \mid \theta_{1:p}, \theta_{p+S_i}, \mathcal{H}_t) \tag{41}$$

for some $\eta \in (0, 1]$, and $S_{i+1} = S_i \cup \{\widehat{J}_i\}$.

The following theorem concerns k expansions at the fixed history $\mathcal{H}_t$.

Theorem F.4. *Assume the conditions of Theorem F.1 and Assumption F.3, and let* $1 \leqslant k \leqslant m$. *Let* $S_k^\star \in \arg\max_{|S|\leqslant k} F_t(S)$, *and let* S_k *be the set obtained after* k *calls to the approximate oracle. Then*

$$\mathbb{E}[F_t(S_k)] \geqslant \left[1 - \left(1 - \frac{\eta}{k}\right)^k\right] F_t(S_k^\star) \geqslant (1 - e^{-\eta}) F_t(S_k^\star). \tag{42}$$

For an exact oracle ($\eta = 1$), *this becomes the* $(1 - 1/e)$ *guarantee.*

Proof. Monotonicity and submodularity imply

$$F_t(S_k^\star) - F_t(S_i) \leqslant F_t(S_i \cup S_k^\star) - F_t(S_i) \tag{43}$$

$$\leqslant k \max_{j \notin S_i} \big[F_t(S_i \cup \{j\}) - F_t(S_i)\big]. \tag{44}$$

Combining this inequality with equation 41 gives

$$\mathbb{E}[F_t(S_{i+1}) - F_t(S_i) \mid S_i] \geqslant \frac{\eta}{k}\big[F_t(S_k^\star) - F_t(S_i)\big]. \tag{45}$$

Therefore, conditioned on S_i,

$$\mathbb{E}[F_t(S_k^\star) - F_t(S_{i+1}) \mid S_i] = F_t(S_k^\star) - F_t(S_i) - \mathbb{E}[F_t(S_{i+1}) - F_t(S_i) \mid S_i] \tag{46}$$

$$\leqslant \left(1 - \frac{\eta}{k}\right)\big[F_t(S_k^\star) - F_t(S_i)\big]. \tag{47}$$

Taking expectations on both sides gives

$$F_t(S_k^\star) - \mathbb{E}[F_t(S_{i+1})] \leqslant \left(1 - \frac{\eta}{k}\right)\left(F_t(S_k^\star) - \mathbb{E}[F_t(S_i)]\right). \tag{48}$$

Since $S_0 = \varnothing$ and $F_t(\varnothing) = 0$, iterating this inequality for $i = 0, \ldots, k-1$ yields

$$F_t(S_k^\star) - \mathbb{E}[F_t(S_k)] \leqslant \left(1 - \frac{\eta}{k}\right)^k F_t(S_k^\star). \tag{49}$$

Rearranging,

$$\mathbb{E}[F_t(S_k)] \geqslant \left[1 - \left(1 - \frac{\eta}{k}\right)^k\right] F_t(S_k^\star) \geqslant (1 - e^{-\eta}) F_t(S_k^\star), \tag{50}$$

where the last inequality follows from $(1 - \eta/k)^k \leqslant e^{-\eta}$. □

Adding θ_{p+1} is useful only if subsequent questions reveal it. Let W_{p+1} be the collection of responses to the questions targeting θ_{p+1}.

Assumption F.5. The responses W_{p+1} contain no task-relevant information beyond that mediated by the expanded latent state:

$$Y^\star \perp\!\!\!\perp W_{p+1} \mid \theta_{1:p}, \theta_{p+1}, \mathcal{H}_t. \tag{51}$$

Lemma F.6. *Under Assumption F.5,*

$$I(Y^\star; W_{p+1} \mid \theta_{1:p}, \mathcal{H}_t) = I(Y^\star; \theta_{p+1} \mid \theta_{1:p}, \mathcal{H}_t) - I(Y^\star; \theta_{p+1} \mid \theta_{1:p}, \mathcal{H}_t, W_{p+1}) \tag{52}$$

$$\geqslant I(Y^\star; \theta_{p+1} \mid \theta_{1:p}, \mathcal{H}_t) - \mathbb{H}(\theta_{p+1} \mid \theta_{1:p}, \mathcal{H}_t, W_{p+1}). \tag{53}$$

Consequently, if the targeted transcript leaves at most ε *entropy about* θ_{p+1}, *it realizes all but at most* ε *of the dimension's representation value.*

Proof. Applying the chain rule with θ_{p+1} gives

$$I(Y^\star; \theta_{p+1}, W_{p+1} \mid \theta_{1:p}, \mathcal{H}_t) = I(Y^\star; \theta_{p+1} \mid \theta_{1:p}, \mathcal{H}_t) + I(Y^\star; W_{p+1} \mid \theta_{1:p}, \theta_{p+1}, \mathcal{H}_t). \tag{54}$$

The second term is zero by Assumption F.5, so

$$I(Y^\star; \theta_{p+1}, W_{p+1} \mid \theta_{1:p}, \mathcal{H}_t) = I(Y^\star; \theta_{p+1} \mid \theta_{1:p}, \mathcal{H}_t). \tag{55}$$

Applying the chain rule in the opposite order gives

$$I(Y^\star; \theta_{p+1}, W_{p+1} \mid \theta_{1:p}, \mathcal{H}_t) = I(Y^\star; W_{p+1} \mid \theta_{1:p}, \mathcal{H}_t) + I(Y^\star; \theta_{p+1} \mid \theta_{1:p}, \mathcal{H}_t, W_{p+1}). \tag{56}$$

Therefore,

$$I(Y^\star; W_{p+1} \mid \theta_{1:p}, \mathcal{H}_t) = I(Y^\star; \theta_{p+1} \mid \theta_{1:p}, \mathcal{H}_t) - I(Y^\star; \theta_{p+1} \mid \theta_{1:p}, \mathcal{H}_t, W_{p+1}). \tag{57}$$

Finally, substituting $I(Y^\star; \theta_{p+1} \mid \theta_{1:p}, \mathcal{H}_t, W_{p+1}) \leqslant \mathbb{H}(\theta_{p+1} \mid \theta_{1:p}, \mathcal{H}_t, W_{p+1})$ into the preceding equality proves equation 53. □

Expansion can also break a plateau in the current question bank. Suppose every unasked pair in the current bank satisfies

$$\max_{(q,u)\ \text{unasked}} I(\theta_{1:p}; Y_{q,u} \mid \mathcal{H}_t;\, q, u) \leqslant \varepsilon. \tag{58}$$

Let $(q_{\text{new}}, u_{\text{new}})$ be a newly generated pair targeting θ_{p+1}, and denote $\nu = I(\theta_{p+1}; Y_{q_{\text{new}},u_{\text{new}}} \mid \theta_{1:p}, \mathcal{H}_t;\, q_{\text{new}}, u_{\text{new}})$. The chain rule then gives

$$\begin{aligned} I((\theta_{1:p}, \theta_{p+1}); Y_{q_{\text{new}},u_{\text{new}}} \mid \mathcal{H}_t;\, q_{\text{new}}, u_{\text{new}}) &= I(\theta_{1:p}; Y_{q_{\text{new}},u_{\text{new}}} \mid \mathcal{H}_t;\, q_{\text{new}}, u_{\text{new}}) \\ &\quad + I(\theta_{p+1}; Y_{q_{\text{new}},u_{\text{new}}} \mid \theta_{1:p}, \mathcal{H}_t;\, q_{\text{new}}, u_{\text{new}}) \geqslant \nu. \end{aligned} \tag{59}$$

Thus $\nu > \varepsilon$ exposes a one-step information opportunity larger than any pair available before expansion.

G Graphical Model

The complete BALAR process is not represented by one fixed-structure probabilistic graphical model: Expand can change the latent variables and question bank during interaction, and subsequent actions are selected adaptively from the observed history. Conditional on a realized sequence of Ask and Expand actions, however, each stage admits a graphical-model interpretation. Figure 11 illustrates one such trajectory. The initial joint prior factorizes across dimensions. An observed response to a selected question then generally induces posterior dependence between those dimensions. Expansion adds a new latent dimension using the conversation-conditioned prior described in Section 3.5, augments the question bank, and permits further belief updates. The resulting joint posterior and full conversation history finally condition the final answer generation call.

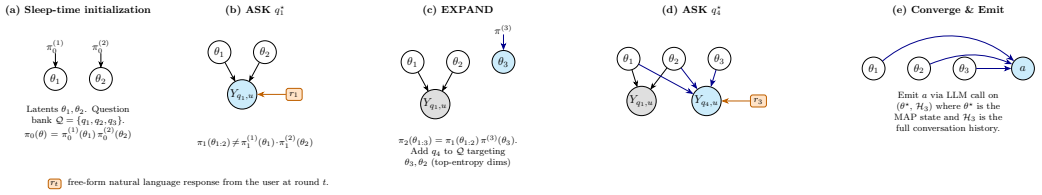


Figure 11: Graphical-model view of one realized BALAR trajectory. The sequence proceeds from factorized sleep-time initialization through an Ask action, an Expand action that introduces θ_3 and a targeted question q_4, a further Ask, and final answer generation. The diagram represents the probabilistic dependencies along this realized adaptive execution.

H Experimental Setup

H.1 Datasets

We evaluate on three structurally distinct benchmarks : **AR-Bench-DC** (detective reasoning), **AR-Bench-SP** (situation puzzles), and **iCraft-MD** (clinical diagnosis).

AR-Bench-DC (Zhou et al., 2025). 100 detective cases, each with scene metadata, victim description, and up to 5 suspect profiles. Each suspect holds private information (story, task, alibi). The agent must identify the true murderer by interrogating suspects. Scored by exact match.

AR-Bench-SP (Zhou et al., 2025). 100 situation puzzles, where the agent must reconstruct the hidden explanation of a puzzling scenario by asking yes/no questions to a user who knows the solution. The original AR-Bench evaluation measures the final answer using a character-level F1 similarity with the ground-truth explanation (Zhou et al., 2025). However, this metric is insensitive to semantic structure and word order. For instance, a random permutation of the same characters can still achieve a perfect score. We therefore do not use the character-level F1 metric and instead evaluate solutions using non-strict semantic equivalence (Kuhn et al., 2023) which we explain in Appendix K.

iCraft-MD (Li et al., 2024). 140 patient cases from CRAFT-MD. Each case contains a clinical multiple-choice question, patient demographic information, a chief complaint, and a list of atomic clinical facts private to the user. Scored by exact match on the correct answer option.

For AR-Bench-DC, each user is a suspect with its own private info (story + task). For AR-Bench-SP and iCraft-MD, there is a single user (the puzzle narrator or patient, respectively).

H.2 Baselines

We compare against dataset-specific interactive baselines implemented in our experimental pipeline. Our baseline comparison focuses on training-free, inference-time methods.

AR-Bench baselines. For the detective reasoning and situation puzzle benchmarks (AR-Bench-DC and AR-Bench-SP), we evaluate four baselines: **Few-Shot (AR-Bench)** which is the best-performing method reported in Zhou et al. (2025), **UoT** (Hu et al., 2024), **ToT** (Yao et al., 2023), and **Proactive CoT** (Deng et al., 2023b). All baselines are run with a maximum of 25 interaction turns, using the best hyperparameter settings reported in Zhou et al. (2025).

Medical dialogue baselines. For the medical benchmark iCraft-MD, we evaluate two baselines: **Zero-Shot**, where an LLM iteratively generates clarifying questions from the conversation history and produces a final answer after the interaction budget is exhausted, and **MediQ Expert** (Li et al., 2024), a five-module pipeline comprising initial assessment, abstention, question generation, information integration, and decision making. Both baselines are run with a maximum of 25 interaction rounds. For **MediQ Expert**, we adopt the best-performing configuration reported in Li et al. (2024) (Scale + Rational Generation + Self-Consistency (3)).

We follow the baseline sets used in the original benchmark studies (Zhou et al., 2025; Li et al., 2024) rather than evaluating every method on every dataset. We use the original implementations and retain the authors' prompt templates. Transferring these baselines across benchmarks would require adapting and validating their task-specific prompts and procedures.

Interaction-budget and prompt controls. All methods receive the same maximum budget of 25 user-directed questions. Few-Shot and Zero-Shot always use the full budget, whereas BALAR, UoT, ToT, Proactive CoT, and MediQ Expert may terminate earlier according to their own stopping behavior. BALAR therefore has no greater opportunity to query the user than its baselines. Prompt templates are fixed across all instances within each dataset. Across datasets, BALAR retains the same prompt structure and functional instructions while adapting domain-specific wording. The complete templates are provided in Appendix V.

H.3 Models and User Simulator

We evaluate five open-weight LLMs spanning a range of parameter scales and training paradigms. Specifically, we use `Qwen2.5-7B/14B/32B-Instruct` to evaluate scaling with model size, `QwQ-32B` to assess the effect of reasoning-tuned models, and `Llama-3.1-8B-Instruct` as an instruction-tuned model from a separate family to evaluate generality beyond the Qwen family. All models are served locally via vLLM using bfloat16 precision and bitsandbytes 4-bit quantization. For the Qwen-family models (`Qwen2.5-7B/14B/32B-Instruct`, `QwQ-32B`), we additionally enable YaRN rope scaling (factor $4\times$, original context length 32768) to support a maximum context length of 131072 tokens for long interaction histories. This is not used for `Llama-3.1-8B-Instruct`. In every reported BALAR run, the configured agent model is held fixed across all BALAR LLM calls, including dimension proposal, prior and likelihood elicitation, question generation, soft mapping, expansion, and final-answer generation. The agent model is used without access to external tools. The user simulator is fixed to `Qwen2.5-32B-Instruct` across all experiments, is prompted with the user's private information and must respond to the agent's question in natural language.

H.4 Distilling BALAR into a Questioning Policy

We additionally test whether a smaller model can learn BALAR's question-selection behavior on iCraft-MD and generate questions without running the Bayesian loop at evaluation time.

We use BALAR traces generated with `Qwen2.5-32B-Instruct` as supervision and train `Llama-3.1-8B-Instruct` as the student. For each of three patient-split seeds, the 140 cases are partitioned into two disjoint halves of 70 cases. We run both fold directions, training on one half and evaluating on the other, and repeat each fold with three training seeds. This yields 18 matched evaluation replications per method. The same 140 patients recur across splits and seeds. At turn t, every policy receives the same information available to the BALAR agent: the ambiguous prompt $\mathbf{p}$, the public meta-context $\mathbf{c}$, the observed interaction history $\mathcal{H}_t$, and the previous compact state. The policy never receives the user's private clinical facts or the correct answer $Y^\star$. In the offline SFT and DPO data, the previous compact state comes from the BALAR trace. During evaluation, each policy receives its own previously generated state. A compact state lists BALAR's active dimensions and, for each dimension, the value in the joint MAP state together with a label derived from that value's marginal posterior probability. Appendix U gives the exact inputs, targets, rewards, and losses.

Supervised fine-tuning. For every ASK turn in each training-patient BALAR trace, the target contains the compact state computed from the current belief π_t and the question q_t^* selected by BALAR[3]. The student is trained to map $(\mathbf{p}, \mathbf{c}, \mathcal{H}_t)$ and the previous compact state to this current state and question. This tests whether direct supervision can transfer BALAR's state tracking and question selection without executing BALAR at evaluation time.

Direct preference optimization. At a fixed history $\mathcal{H}_t$, the preferred and rejected outputs contain the same BALAR compact state but different questions. The preferred question has the largest recorded mutual information with the latent state, while the rejected question has the smallest. Starting from the fold's SFT model, DPO increases the relative probability of the higher-information question. Because the input and state output are identical within each pair, the preference signal acts only on question selection.

Dense-reward reinforcement learning. Starting from the fold's SFT model, the student samples groups of candidate questions at training histories $\mathcal{H}_t$. For each valid, non-repeated question, frozen `Qwen2.5-32B-Instruct` first estimates a distribution over the answer options from $(\mathbf{p}, \mathbf{c}, \mathcal{H}_t)$, then uses the user's private clinical facts to simulate a response, and finally re-estimates the answer distribution after adding that response to the history. The normalized decrease in answer-distribution entropy is the question's reward. Repeated and malformed questions receive fixed penalties. We optimize a one-turn group-relative policy objective with a KL penalty to the SFT model. The reward therefore measures the immediate information obtained from one question rather than success over a complete dialogue trajectory.

Evaluation protocol. For each held-out patient, the policy asks 25 questions. A frozen `Qwen2.5-32B-Instruct`, conditioned on the user's private clinical facts, supplies the responses. We measure final-answer accuracy in two ways. The primary evaluator is a separate frozen `Qwen2.5-32B-Instruct` call that receives only $(\mathbf{p}, \mathbf{c}, \mathcal{H}_t)$ and selects an answer option. This measures the information elicited by the policy independently of the student's ability to produce the final answer. The secondary evaluator is the student itself, given the same information. We also record normalized answer-entropy reduction and the number of distinct questions.

I COMPUTATIONAL COMPLEXITY

We characterize both the number of LLM calls made by BALAR and the cost of exact Bayesian inference. Parallelism is used to keep wall-clock latency low. Let p denote the number of dimensions, n the maximum number of values per dimension, $|\mathcal{Q}|$ the number of questions, $|\mathcal{U}|$ the number of users, $|\mathcal{A}|$ the number of possible answers (when defined), and $|\mathcal{Q}_{\text{new}}|$ the number of new questions generated per EXPAND step. For the inference analysis, let $S = |\Theta| = \prod_{j=1}^{p} n_j$ be the number of joint states and let $c_Y = \max_{q \in \mathcal{Q}} |\mathcal{Y}_q|$ be the largest discrete answer set.

Sleep-time initialization (Steps 1–4). Step 1 issues a single LLM call to propose all p dimensions jointly. Step 2 elicits one call per dimension value to assign a prior label, for a total of $\sum_j n_j \leqslant p \cdot n$

[3]The user u_t^* is always the single patient in iCraft-MD.

calls. These are dispatched concurrently. Step 3 issues a single call to generate all $|\mathcal{Q}|$ initial questions. Step 4 issues one call per (q, u, θ_j) triple to fill the likelihood table for that triple, for a total of $|\mathcal{Q}| \cdot |\mathcal{U}| \cdot p$ calls, all dispatched concurrently. When possible answers $\mathcal{A}$ are available, an additional p parallel calls estimate the answer-level likelihood tables (one per dimension). The total number of initialization calls is therefore:

$$C_{\text{init}} = 1 + p \cdot n + 1 + |\mathcal{Q}| \cdot |\mathcal{U}| \cdot p + p \;=\; \mathcal{O}(|\mathcal{Q}| \cdot |\mathcal{U}| \cdot p),$$

all parallelizable except for the sequential dependency between Steps 1–2–3–4 (each step uses outputs of the previous one).

ASK step. Each ASK round issues exactly 2 sequential LLM calls: one to the user simulator to obtain the natural language answer r_t, and one to soft-map r_t to a probability vector $\hat{\omega}$ over the discrete choices. The MI computation over all candidate (q, u) pairs is a single batched tensor operation (no LLM call). Over T_{ask} rounds, the total is $2T_{\text{ask}}$ sequential calls.

EXPAND step. Each EXPAND round issues the following calls:

1. One call to propose the new dimension θ_{p+1}.
2. n_{p+1} parallel calls to elicit its prior (one per value).
3. One call (optional) for the answer-level likelihood of the new dimension (when $\mathcal{A}$ is defined).
4. $|\mathcal{Q}| \cdot |\mathcal{U}|$ parallel calls to compute likelihood tables for all old questions over the new dimension.
5. One call to generate up to $|\mathcal{Q}_{\text{new}}|$ new questions.
6. $|\mathcal{Q}_{\text{new}}| \cdot |\mathcal{U}| \cdot (p + 1)$ parallel calls to compute likelihood tables for the new questions over all dimensions.

The dominant cost per EXPAND round is $\mathcal{O}((|\mathcal{Q}| + |\mathcal{Q}_{\text{new}}|(p + 1)) \cdot |\mathcal{U}|)$ LLM calls, all within each group executed concurrently.

Exact Bayesian inference. Constructing every state-level likelihood tensor $K_{q,u}(\theta, y)$ from its dimension-level factors costs $\mathcal{O}(|\mathcal{Q}||\mathcal{U}|pSc_Y)$. These tensors can then be reused until an EXPAND step changes the state or question space. At each ASK round, computing predictive answer distributions and conditional entropies for every unasked question–user pair costs

$$\mathcal{O}(|\mathcal{Q}||\mathcal{U}|Sc_Y). \tag{60}$$

The posterior update for the selected pair costs $\mathcal{O}(Sc_Y)$, computing all p dimension marginals costs $\mathcal{O}(pS)$, and computing the predictive answer distribution when $\mathcal{A}$ is available costs $\mathcal{O}(S|\mathcal{A}|)$. Thus the MI evaluation in Equation (60) is normally the dominant tensor operation per round.

Storing the joint belief requires $\mathcal{O}(S)$ memory while all state-level likelihoods require

$$\mathcal{O}(|\mathcal{Q}||\mathcal{U}|Sc_Y). \tag{61}$$

After expansion, the same bounds apply with the enlarged state count $S_{\text{new}} = S\, n_{p+1}$ and the augmented question bank. These operations are implemented as batched tensor computations and require no additional LLM calls. Because S grows exponentially with the number of dimensions, BALAR enforces the state-space cap described in Appendix C.

Final answer. A single LLM call produces the final answer conditioned on the MAP state and conversation history.

Empirical call counts. Table 2 reports mean LLM calls per dialogue for the six main-results settings. Initialization dominates AR-Bench-DC with Qwen-32B, whereas expansion dominates iCraft-MD with Qwen-32B and accounts for a substantial share of calls on AR-Bench-SP. The mean totals range from 83.6 ± 2.3 to 362.0 ± 7.6. Llama-8B and Qwen-32B denote `Llama-3.1-8B-Instruct` and `Qwen2.5-32B-Instruct`, respectively.

Table 2: LLM calls per dialogue for the main-results settings, reported as means ± standard errors. Stages are initialization, ASK, expansion, and final-answer generation.

Dataset	Model	Initialization	ASK	Expansion	Final	Total
AR-Bench-DC	Llama-8B	17.0 ± 0.0	20.0 ± 0.0	45.6 ± 2.3	1.0 ± 0.0	83.6 ± 2.3
AR-Bench-DC	Qwen-32B	275.9 ± 0.2	46.2 ± 1.2	38.9 ± 7.4	1.0 ± 0.0	362.0 ± 7.6
AR-Bench-SP	Llama-8B	70.8 ± 0.2	38.2 ± 0.4	107.0 ± 3.4	1.0 ± 0.0	217.0 ± 3.6
AR-Bench-SP	Qwen-32B	67.7 ± 0.3	42.6 ± 1.1	158.4 ± 6.7	1.0 ± 0.0	269.8 ± 7.6
iCraft-MD	Llama-8B	47.1 ± 0.3	10.0 ± 0.8	40.1 ± 3.2	1.0 ± 0.0	98.2 ± 4.0
iCraft-MD	Qwen-32B	8.2 ± 0.1	42.3 ± 1.0	254.5 ± 7.6	1.0 ± 0.0	306.0 ± 8.5

J LIKELIHOOD LABELS

J.1 DEFINITION

We use a discrete label set $\mathcal{L} = \{\texttt{likely}, \texttt{neutral}, \texttt{unlikely}\}$ to elicit priors and likelihoods from the LLM. Each label corresponds to a qualitative judgment grounded in the available context:

- **likely**: the value or answer is explicitly stated, strongly implied, or represents the most natural assumption given the prompt and context.
- **neutral**: the value or answer is plausible but not supported or contradicted by specific evidence. There is insufficient information to prefer it over alternatives.
- **unlikely**: the value or answer is contradicted by the prompt or context, or would require assumptions inconsistent with the provided information.

These qualitative labels are mapped to positive unnormalized weights via a fixed function ϕ, which are normalized within each prior or likelihood table to produce numerical probabilities.

J.2 CROSS-MODEL CONSISTENCY OF LABEL ELICITATION

We sample 850 evaluation units from BALAR traces on iCraft-MD with `Qwen2.5-32B-Instruct`, stratified across four roles: 70 prior units, 80 question-likelihood units, 100 answer-likelihood units, and 600 soft-map units. A prior unit contains one dimension's value set, a likelihood unit contains a complete table, and a soft-map unit contains one user response and its discrete answer choices. Priors and question likelihoods are sampled from initialization, answer likelihoods from initialization and expansion, and soft maps from ASK rounds. `GPT-5-nano` replays the original inputs, using the same role-specific prompts and fixed weight map as BALAR. These prompts are provided in Appendix V.

BALAR elicits qualitative labels from the LLM, maps them to weights through ϕ, and normalizes those weights into the probability vectors used by its Bayesian calculations. We therefore compare the models at two levels: whether they produce the same labels and whether their labels induce similar probability distributions. Exact agreement is the fraction of label cells on which the two models agree. Total variation between normalized probability vectors p and q is

$$\mathrm{TV}(p, q) = \frac{1}{2} \sum_{a} |p_a - q_a|. \tag{62}$$

The resulting exact-agreement and mean-TV scores are reported in Table 3. Soft mapping, which provides evidence for each posterior update, shows the strongest cross-model consistency.

Table 3: Cross-model label consistency between Qwen2.5-32B-Instruct and GPT-5-nano on iCraft-MD.

Role	Sampled units	Label cells	Agreement (%)	Mean TV
Prior	70	227	55.5	0.139
Question likelihood	80	750	43.6	0.202
Answer likelihood	100	1280	49.8	0.189
Soft map	600	1781	78.8	0.094

K SEMANTIC EQUIVALENCE METRIC

For tasks where no single canonical ground-truth answer exists but multiple phrasings can express the same underlying meaning, such as AR-Bench-SP, metrics such as character-level F1 similarity are inadequate: a random permutation of the ground-truth characters can achieve a perfect score while a faithful paraphrase may receive a low one.

Following Kuhn et al. (2023), let a denote the agent's predicted answer and a^* the ground-truth solution, both conditioned on the same prompt $\mathbf{p}$. We use a judge LLM $\mathcal{J}$ (`Qwen2.5-32B-Instruct`) to assess the directional entailment relation

$$\mathcal{J}(a_1, a_2 \mid \mathbf{p}) \in \{\texttt{entailment}, \texttt{neutral}, \texttt{contradiction}\},$$

which evaluates whether a_1 semantically entails a_2 in the context of $\mathbf{p}$. We query $\mathcal{J}$ in both directions, obtaining

$$e_{\rightarrow} = \mathcal{J}(a, a^* \mid \mathbf{p}), \qquad e_{\leftarrow} = \mathcal{J}(a^*, a \mid \mathbf{p}).$$

We declare a and a^* *semantically equivalent* (non strict) when neither direction is a contradiction and the pair is not jointly neutral:

$$\mathrm{Eq}(a, a^*) \;=\; \mathbf{1}[(\texttt{contradiction} \notin \{e_{\rightarrow}, e_{\leftarrow}\}) \;\wedge\; ((e_{\rightarrow}, e_{\leftarrow}) \neq (\texttt{neutral}, \texttt{neutral}))].$$

This criterion accepts predictions that are entailed by the reference in at least one direction, provided no direction is contradicted. It reflects the nature of such tasks, where multiple correct phrasings of the hidden explanation are possible and a partial match (one direction entailment, the other neutral) often corresponds to a correct answer that omits minor details.

L HYPERPARAMETERS

Fixed parameters are shared across all datasets. The label set is $\mathcal{L} = \{\texttt{likely}, \texttt{neutral}, \texttt{unlikely}\}$ with unnormalized weight map $\phi(\texttt{likely}) = 0.8$, $\phi(\texttt{neutral}) = 0.5$, $\phi(\texttt{unlikely}) = 0.2$. This mapping is chosen as simple and heuristically reasonable. The ablation in Table 5 shows it performs well in practice.

For Figure 4, the fixed configurations are DC (T_{ask}=25, α=0.3, p=5, $|\mathcal{Q}|$=10), SP (T_{ask}=25, α=0.3, β=0.5, p=5, $|\mathcal{Q}|$=10), and iCraft-MD (T_{ask}=25, α=0.3 (top) and 0.1 (bottom), p=1, $|\mathcal{Q}|$=2).

Table 4: Hyperparameter settings.

Parameter	DC/iCraft-MD	SP
Variable		
Confidence threshold (α)	$\{0.1, 0.3\}$	$\{0.1, 0.3\}$
Initial dims / questions ($(p, \lvert\mathcal{Q}\rvert)$)	$\{(1,2), (5,3), (5,5), (5,10)\}$	
Marginal convergence (β)	1.0 (fixed)	$\{0.5, 0.7\}$
Fixed		
Total round budget (T)	100	
Max ask rounds (T_{ask})	25	
Max values per dim	4	
Max choices per question	4	2 (yes/no)
Max new questions per EXPAND	4	
Top-entropy dims for EXPAND (m_{top})	2	
Expand multiplier (λ)	1.0	
Temperature / top-p	0.1 / 1.0	

Choosing hyperparameters. BALAR's hyperparameters include the confidence threshold α, the marginal convergence fraction β, the ASK budget T_{ask}, the initial number of dimensions p and questions $|\mathcal{Q}|$, and the expand multiplier λ. The appropriate values of these parameters are task-dependent. The threshold α should reflect the acceptable residual uncertainty: for high-stakes decisions (e.g. medical diagnosis), a small α is appropriate, whereas for lower-stakes tasks larger values reduce unnecessary questioning. The budget T_{ask} should be set to the maximum number of questions a user is willing to answer. $(p, |\mathcal{Q}|)$ trades off initialization cost against the expressiveness of the prior representation. Starting with $(p{=}1, |\mathcal{Q}|{=}2)$ and relying on EXPAND is a natural default when the structure of ambiguity is unknown. λ controls how aggressively BALAR triggers expansion. Smaller values expand more aggressively, whereas larger values favor asking from the current bank.

M PRIOR SENSITIVITY

Table 5: Prior sensitivity ablation on iCraft-MD / Qwen2.5-32B-Instruct. Config: T_{ask}=25, α=0.1, p=1, $|\mathcal{Q}|$=2.

Variant	Accuracy (%)
Default priors (0.8/0.5/0.2)	73.6 ± 3.7
Flat priors (0.6/0.5/0.4)	70.7 ± 3.8
Sharp priors (0.9/0.5/0.1)	52.1 ± 4.2

We study the effect of different prior mappings. Table 5 shows that BALAR is robust to moderate changes in the prior mapping: flatter priors incur only a 3.9% relative drop. However, overly sharp priors degrade performance substantially (29.1% relative drop), as they over-concentrate the initial belief.

N QUESTION QUALITY

We evaluate questions from iCraft-MD dialogues under `Qwen2.5-32B-Instruct` using a `GPT-5-nano` judge. For each question, the judge receives the public case, candidate diagnoses, the complete set of initial and expanded BALAR dimensions, and all previously asked questions. User responses are not included. The evaluation is post-hoc, and the judge's ratings do not influence BALAR's actions.

The judge assigns scores from 1 to 5 for three criteria: relevance to the case, expected informativeness for narrowing the candidate diagnoses, and non-redundancy against prior questions. Non-redundancy therefore concerns the previously asked questions, not the full dialogue including patient responses. The exact prompts appear in Appendix V.14.

The judge also identifies one primary target dimension per question, or none. We report the number of distinct assigned dimensions per dialogue, which uses the full set of initial and expanded dimensions. Table 6 reports dialogue-level means and standard errors.

The judge rates questions as highly relevant, with more moderate informativeness and non-redundancy. Each dialogue targets 6.14 distinct dimensions on average. These scores characterize perceived question quality rather than directly measuring realized information gain.

Table 6: Question-quality evaluation on iCraft-MD with `Qwen2.5-32B-Instruct`.

Metric	Mean ± SE
Relevance	4.47 ± 0.016
Informativeness	3.63 ± 0.014
Non-redundancy	3.69 ± 0.012
Distinct targeted dimensions	6.14 ± 0.140

O PREDICTIVE-BELIEF CALIBRATION

For each model-dataset pair in Table 7, we assess the final predictive answer distribution $\hat{p}(a)$ defined in Section 3.6. Motivated by recent work on calibration in agentic systems with explicit linguistic belief states (Murphy, 2026), we use leave-one-out temperature scaling. For each held-out instance, the temperature $\tau > 0$ is fit on the remaining instances by minimizing multinomial negative log-likelihood (NLL), and the held-out distribution becomes :

$$\hat{p}_{\text{cal}}(a) = \frac{\hat{p}(a)^{1/\tau}}{\sum_{a' \in \mathcal{A}} \hat{p}(a')^{1/\tau}} = \text{softmax}\left(\frac{\log \hat{p}(a)}{\tau}\right).$$

Thus $\tau > 1$ softens an overconfident distribution, whereas $\tau < 1$ sharpens an underconfident one. We report expected calibration error (ECE), NLL, and the multiclass Brier score. Lower values are better for all three.

Table 7: Calibration of BALAR's predictive answer belief using leave-one-out temperature scaling. "Cal." denotes the corresponding metric after calibration. Lower is better.

Dataset	Model	τ	ECE	Cal.	NLL	Cal.	Brier	Cal.
iCraft-MD	Llama-8B	3.256	0.2627	0.0439	1.6498	1.2771	0.7951	0.6909
iCraft-MD	Qwen-7B	3.909	0.3853	0.0863	1.9083	1.2691	0.8691	0.6829
iCraft-MD	Qwen-14B	2.762	0.2656	0.0862	1.4875	1.1160	0.6807	0.5997
iCraft-MD	Qwen-32B	1.531	0.1315	0.0668	0.9773	0.9211	0.5066	0.4926
iCraft-MD	QwQ-32B	1.514	0.1205	0.0839	1.0952	1.0501	0.5914	0.5728
AR-DC	Llama-8B	0.328	0.2836	0.1010	1.2497	1.0691	0.6414	0.5396
AR-DC	Qwen-32B	0.465	0.1469	0.1309	0.9500	0.8178	0.4943	0.4442
AR-DC	QwQ-32B	0.565	0.1721	0.1361	1.1176	1.0545	0.5838	0.5587

Temperature scaling improves all three metrics for every evaluated pair. On iCraft-MD, all fitted temperatures exceed one, indicating overconfidence before scaling, and scaling reduces ECE by factors ranging from 1.4 to 6.0. Calibration generally improves with model capacity: the two 32B models have the lowest unscaled ECE, although the trend is not strictly monotone across the smaller models. On AR-DC, every fitted temperature is below one, indicating underconfidence, with smaller but consistent improvements across all three metrics. These opposite directions also show that a single global temperature should not be assumed to transfer across tasks.

BALAR does not emit $\arg\max_a \hat{p}(a)$ directly. As described in Section 3.6, it passes the MAP latent state and dialogue history to a final LLM reasoning call. Table 8 compares the two outputs.

Table 8: Accuracy of the predictive-belief argmax and the answer emitted by BALAR's final LLM call.

Dataset	Model	Argmax (%)	Emitted (%)
iCraft-MD	Llama-8B	44.3	57.1
iCraft-MD	Qwen-7B	42.1	50.7
iCraft-MD	Qwen-14B	54.3	62.1
iCraft-MD	Qwen-32B	62.9	73.6
iCraft-MD	QwQ-32B	55.0	70.0
AR-DC	Llama-8B	62.0	61.0
AR-DC	Qwen-32B	63.0	47.0
AR-DC	QwQ-32B	58.0	51.0

The emitted answer is consistently more accurate on iCraft-MD, suggesting that the final reasoning call can use the structured state and dialogue context more effectively than a direct answer-belief argmax in that domain. This benefit does not extend to AR-DC, where the emitted answer is slightly worse for Llama-8B and both Qwen-32B models.

P Answer-Belief Dynamics

We examine concentration of the predictive answer belief $\hat{p}_t(a)$ in Equation (13) on iCraft-MD with `Qwen2.5-32B-Instruct`. At each saved belief state, we compute

$$H(t) = -\sum_{a \in \mathcal{A}} \hat{p}_t(a) \log \hat{p}_t(a), \tag{63}$$

$$C(t) = \max_{a \in \mathcal{A}} \hat{p}_t(a). \tag{64}$$

Entropy is measured in nats. These quantities describe the answer belief used for stopping, not the latent-state entropy or the final LLM-emitted answer.

Figure 12 plots instance means against cumulative ASK count. For each dialogue and ASK count, we use the last available belief state at that count, including any intervening expansion. Each point averages only dialogues reaching that count. Mean entropy decreases from approximately 1.4 to 0.82 nats and mean maximum probability increases from approximately 0.35 to 0.66. These trends indicate increasing belief concentration, not necessarily attainment of the stopping threshold or convergence to the correct answer.

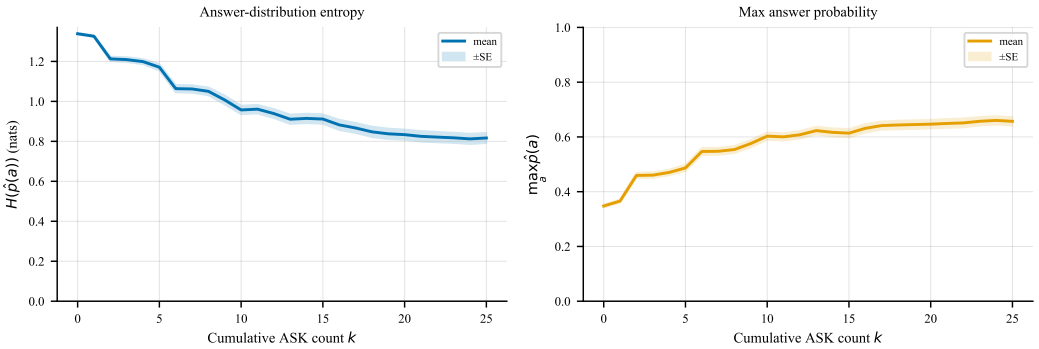


Figure 12: Answer-belief entropy (left) and maximum answer probability (right) versus cumulative ASK count on iCraft-MD with `Qwen2.5-32B-Instruct`. Curves show instance means with $\pm$ one standard error.

Q Sensitivity of Initial Dimensions

We evaluate how strongly the initial latent representation depends on the wording of the dimension-generation prompt. On iCraft-MD, we use `Qwen2.5-32B-Instruct` to generate five initial dimensions under three prompt variants:

- **v0**: the standard clinical prompt used in the main experiments,
- **v1**: the same physician system role with definitions and other task-specific scaffolding removed,
- **v2**: a domain-neutral formulation containing no clinical language.

For each pair of variants, a `GPT-5-nano` judge matches semantically equivalent dimensions and computes the Jaccard similarity between the resulting dimension sets. Table 9 reports the mean across evaluated cases with standard errors.

Table 9: Semantic overlap between sets of five initial dimensions generated under different prompt formulations on iCraft-MD using `Qwen2.5-32B-Instruct`.

Prompt comparison	**Mean Jaccard $\pm$ SE**
v0 vs. v1	0.678 ± 0.012
v0 vs. v2	0.564 ± 0.012
v1 vs. v2	0.620 ± 0.024

The dimension sets retain moderate semantic overlap across all three formulations. In particular, the overlap involving v2 indicates that the representation is not determined solely by clinical terminology or detailed prompt scaffolding. At the same time, v0 and v2 have the lowest agreement, showing measurable sensitivity to domain-specific framing.

The exact v1 and v2 prompts and the semantic-matching judge prompts are reproduced in Appendix V.12. v0 is the standard iCraft-MD prompt in Appendix V.2.

R Round Distribution

We analyze how BALAR allocates its interaction budget across ASK and EXPAND actions. Figure 13 shows the distribution of the number of ASK and EXPAND rounds per instance under `Qwen2.5-32B-Instruct`. On AR-Bench-DC, nearly all runs exhaust the full ASK budget with few EXPAND calls, suggesting the initial dimensions suffice. On AR-Bench-SP, EXPAND is called 1–5 times per run. On iCraft-MD, EXPAND is called 5–8 times consistently, indicating that clinical cases regularly require state refinement.

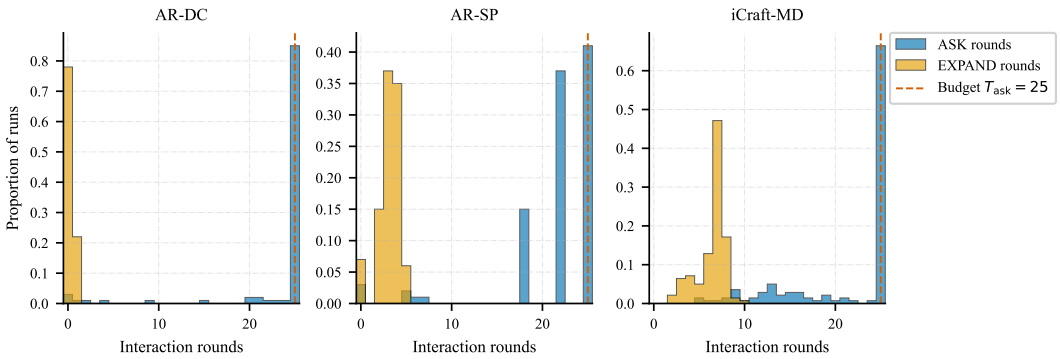


Figure 13: Distribution of ASK and EXPAND rounds for BALAR under `Qwen2.5-32B-Instruct` on each dataset. Each panel shows normalized histograms over runs, so bar heights correspond to proportions. The dashed vertical line indicates the selected T_{ask} budget. Fixed configs: DC (T_{ask}=25, α=0.3, p=5, $|\mathcal{Q}|$=10), SP (T_{ask}=25, α=0.3, β=0.5, p=5, $|\mathcal{Q}|$=10), iCraft-MD (T_{ask}=25, α=0.1, p=1, $|\mathcal{Q}|$=2).

S ORACLE

Oracle. The LLM is given the prompt, public context, and all users' private information, providing an upper bound on performance.

Oracle dimensions. The oracle returns a minimal set of ambiguity dimensions and their ground-truth values. Table 10 reports the average number of dimensions ($\pm$ SE), measuring intrinsic disambiguation complexity.

Table 10: Oracle Dimensions. Entries are the average number of oracle dimensions required to resolve a case, reported as mean $\pm$ SE over instances.

Model	AR-DC	AR-SP	iCraft-MD
Llama-3.1-8B-Instruct	4.7 ± 0.05	4.2 ± 0.1	5.3 ± 0.1
Qwen2.5-32B-Instruct	3.9 ± 0.03	2.4 ± 0.1	3.6 ± 0.1

Oracle accuracy. We report oracle accuracy for iCraft-MD in Table 11. For AR-Bench-DC and AR-Bench-SP, full private information deterministically fixes the answer, so oracle accuracy is trivially 100% and omitted.

Table 11: Oracle accuracy (%) on iCraft-MD: upper bound when the agent has access to all patient information.

Model	Accuracy (%)
Llama-3.1-8B-Instruct	72.7 ± 3.8
Qwen2.5-7B-Instruct	65.9 ± 4.2
Qwen2.5-14B-Instruct	78.6 ± 3.5
Qwen2.5-32B-Instruct	83.6 ± 3.1
QwQ-32B	83.6 ± 3.1

Table 10 shows that resolving an ambiguous prompt requires on average 3–5 disambiguating dimensions, with larger models proposing fewer, more precise dimensions. On iCraft-MD, oracle accuracy ranges from 65.9% to 83.6% in Table 11, reflecting irreducible ambiguity in the underlying clinical questions. BALAR with `Qwen2.5-32B-Instruct` achieves 73.6% without access to private information, reaching 88% of the oracle upper bound.

T DIALOGUE STATE TRACKING EVALUATION

We evaluate internal state tracking on iCraft-MD trajectories from `Qwen2.5-32B-Instruct`, using the oracle dimensions and reference values described above. These oracle annotations are LLM-generated evaluation references.

For each instance, a `GPT-5-nano` judge receives the public clinical case, the oracle dimensions and their optimal value annotations, all BALAR dimensions and value sets (including those added by EXPAND), and the complete list of asked questions with round numbers. For each oracle dimension, it returns an aligned BALAR dimension, or -1 if none exists, the BALAR value label equivalent to the oracle reference value, and the rounds of questions that probed that dimension. Oracle information is used only for this post-hoc evaluation and is not supplied to the BALAR agent. The exact judge prompts are in Appendix V.13.

For instance i, let m_i be its number of scored oracle dimensions and $c_i(k)$ the number whose aligned BALAR value in the joint MAP state after k cumulative ASK actions belongs to the judge's matching set. We define

$$\mathrm{JGA}_i(k) = \mathbf{1}[c_i(k) = m_i], \tag{65}$$

$$\mathrm{SER}_i(k) = 1 - \frac{c_i(k)}{m_i}. \tag{66}$$

Thus SER is an oracle-aligned slot-error fraction. It penalizes missing or incorrect oracle slots but not additional BALAR dimensions. JGA requires every scored oracle slot to be correct simultaneously.

Let $d_i(k)$ count oracle dimensions probed by at least one of the first k asked questions, as identified by the judge. Questioning coverage is

$$\mathrm{Coverage}_i(k) = \frac{d_i(k)}{m_i}. \tag{67}$$

A dimension can be probed without having an equivalent BALAR dimension, so this metric measures questioning breadth rather than representational coverage or value correctness.

Table 12: Oracle-aligned tracking and questioning coverage on iCraft-MD with `Qwen2.5-32B-Instruct`, for the 93 evaluated instances reaching 25 ASK actions. Entries are instance means $\pm$ standard errors.

Metric	Mean $\pm$ SE
Joint goal accuracy	0.280 ± 0.047
Oracle-aligned slot-error rate	0.380 ± 0.033
Questioning coverage	0.981 ± 0.008

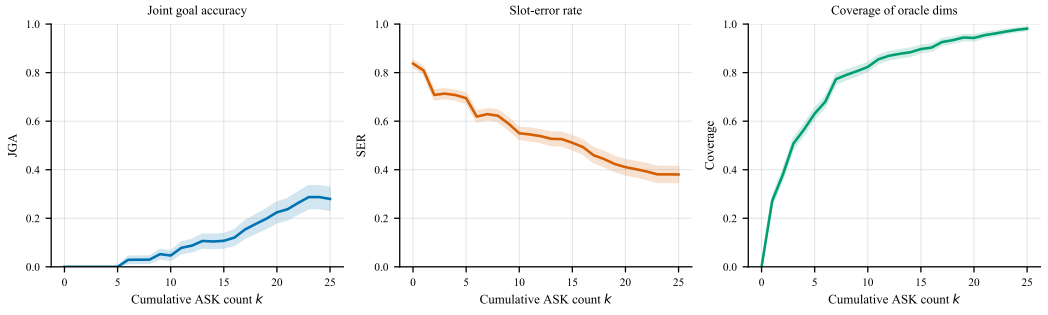


Figure 14: Oracle-aligned JGA, SER, and questioning coverage versus cumulative ASK count on iCraft-MD with `Qwen2.5-32B-Instruct`. Shaded bands indicate $\pm$ one standard error. The right panel's "Coverage of oracle dims" denotes dimensions probed by questions, not dimensions represented in BALAR's ontology.

U DISTILLATION

- **SFT :** given a dialogue prefix, the student learns to output BALAR's compact state and next question.
- **DPO :** given the same prefix and state, the student learns to prefer a question with high latent-state mutual information over one with low mutual information.
- **RL :** the student samples questions, a frozen Qwen model simulates each answer, and the student is rewarded when that answer reduces uncertainty over the clinical answer choices.

U.1 SETUP

Recall that $\mathbf{p}$ is the ambiguous prompt, and $\mathbf{c}$ is the public meta-context. Before turn t, let

$$\mathcal{H}_t = \{(q_1, a_1), \ldots, (q_{t-1}, a_{t-1})\} \tag{68}$$

be the interaction history and let s_{t-1} be the previous compact state. Every student input is

$$\mathcal{I}_t = (\mathbf{p}, \mathbf{c}, \mathcal{H}_t, s_{t-1}). \tag{69}$$

The offline training records use BALAR's previous state in $\mathcal{I}_t$. During evaluation, the student uses its own previous state. The student never sees the private patient record or the correct answer. Its output is

$$y_t = (s_t, q_t), \tag{70}$$

where s_t is a list of dimension–value–confidence triples and q_t is one question.

The **student** is `Llama-3.1-8B-Instruct`. Only its LoRA adapter is updated. The **BALAR teacher** uses `Qwen2.5-32B-Instruct` to create the offline states, questions, and mutual-information scores. In RL and evaluation, a separate frozen instance of the same Qwen model acts as the patient simulator and answer-distribution estimator. None of the Qwen parameters are updated.

For completeness, recall that $\pi_t(\theta)$ is BALAR's joint posterior over its active dimensions, and let

$$\hat{\theta}_t = \arg\max_{\theta \in \Theta} \pi_t(\theta) \tag{71}$$

be its joint MAP state. For dimension j, BALAR emits the value $\hat{\theta}_{t,j}$ and converts its marginal probability $\pi_t^{(j)}(\hat{\theta}_{t,j})$ to a label $\ell(\pi_t^{(j)}(\hat{\theta}_{t,j}))$. The compact state is then

$$s_t^* = \left\{(\theta_j, \hat{\theta}_{t,j}, \ell(\pi_t^{(j)}(\hat{\theta}_{t,j})))\right\}_{j=1}^{p}. \tag{72}$$

The value comes from the *joint* MAP assignment, while the confidence label comes from that value's *marginal* probability.

At test time, no BALAR loop generates the student's questions. The trained student predicts a state and question, the frozen Qwen patient simulator answers, and the next student turn receives that new dialogue plus the student's previous state.

U.2 SUPERVISED FINE-TUNING (SFT)

For every ASK turn in a training-patient BALAR trace, the data contain the input $\mathcal{I}_i$ and the teacher output

$$y_i^* = (s_i^*, q_i^{\mathrm{BALAR}}). \tag{73}$$

The target includes both the complete pre-question state and the question BALAR selected. If $y_{i,m}^*$ is target token m, SFT minimizes completion-only cross-entropy:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\sum_i \sum_m \log \pi_\theta\left(y_{i,m}^* \mid \mathcal{I}_i, y_{i,<m}^*\right). \tag{74}$$

This experiment trains the student to copy both BALAR's state update and BALAR's next question.

U.3 DIRECT PREFERENCE OPTIMIZATION (DPO)

DPO starts from the fold's SFT model. At each retained BALAR prefix, BALAR's question score is

$$\mathrm{MI}_t(q) = \mathbb{H}(\pi_t) - \mathbb{E}_{Y\sim p_t^{q,u}}\left[\mathbb{H}(\pi_t^{q,u,Y})\right] = I_t(\theta; Y \mid \mathcal{H}_t;\, q, u). \tag{75}$$

The chosen question q_i^+ is the available question with the largest recorded score. The rejected question q_i^- is the distinct available question with the smallest score. A pair is kept only when $\mathrm{MI}_i(q_i^+) - \mathrm{MI}_i(q_i^-) \geqslant 0.01$. Both completions contain the same teacher state:

$$y_i^+ = (s_i^*, q_i^+), \qquad y_i^- = (s_i^*, q_i^-). \tag{76}$$

Consequently, their only intended difference is the question. Let π_{ref} be the fixed SFT model and let π_θ be the trainable DPO model initialized from it. Define the sequence log-probability difference

$$\Delta_\pi(i) = \log \pi(y_i^+ \mid \mathcal{I}_i) - \log \pi(y_i^- \mid \mathcal{I}_i). \tag{77}$$

The DPO loss is

$$\mathcal{L}_{\mathrm{DPO}}(\theta) = -\sum_i \log \sigma(\beta\,[\Delta_{\pi_\theta}(i) - \Delta_{\pi_{\mathrm{ref}}}(i)]), \qquad \beta = 0.1. \tag{78}$$

This trains the student to assign more probability to the high-MI question than to the low-MI question, relative to the SFT model.

U.4 DENSE-REWARD REINFORCEMENT LEARNING (RL)

RL starts independently from the SFT model. At an input $\mathcal{I}_i$, the current student samples a group of G outputs

$$y_{i,g} = (s_{i,g}, q_{i,g}), \qquad g = 1, \ldots, G. \tag{79}$$

For each valid, non-repeated question, a frozen Qwen performs three steps:

1. Using only $(\mathbf{p}, \mathbf{c}, \mathcal{H}_i)$, estimate a distribution p_i^- over all K clinical answer choices.
2. Using the private patient record, simulate an answer $a_{i,g}$ to $q_{i,g}$.
3. Using only $(\mathbf{p}, \mathbf{c}, \mathcal{H}_i \cup \{(q_{i,g}, a_{i,g})\})$, estimate a new distribution $p_{i,g}^+$ over the same choices.

With $H(p) = -\sum_{k=1}^K p_k \log p_k$, the reward is

$$r_{i,g} = \frac{H(p_i^-) - H(p_{i,g}^+)}{\log K}. \tag{80}$$

We clip this value to $[-1, 1]$. A repeated question receives -0.1, a malformed JSON receives -0.25, and a question the patient cannot answer receives 0.

Within each group, rewards are standardized:

$$A_{i,g} = \frac{r_{i,g} - \bar{r}_i}{\sqrt{G^{-1}\sum_{g'=1}^G (r_{i,g'} - \bar{r}_i)^2}}. \tag{81}$$

If all G rewards are equal, every advantage is set to zero. For output token m, let

$$\rho_{i,g,m} = \exp(\log \pi_\theta(y_{i,g,m} \mid \cdot) - \log \pi_{\mathrm{old}}(y_{i,g,m} \mid \cdot)), \tag{82}$$

where π_{old} is the student that sampled the group. The token loss is

$$\mathcal{L}_{\mathrm{RL}} = -\min(\rho A, \mathrm{clip}(\rho, 1-\epsilon, 1+\epsilon)A) + \lambda_{\mathrm{KL}}\,\widehat{D}_{\mathrm{k3}}(\pi_\theta \| \pi_{\mathrm{ref}}), \tag{83}$$

$$\widehat{D}_{\mathrm{k3}}(\pi_\theta \| \pi_{\mathrm{ref}}) = \exp(\delta) - \delta - 1, \qquad \delta = \log \pi_{\mathrm{ref}} - \log \pi_\theta, \tag{84}$$

averaged over completion tokens and sampled outputs, with $\epsilon = 0.2$ and $\lambda_{\mathrm{KL}} = 0.02$. Here π_{ref} is the fixed SFT model, and $\widehat{D}_{\mathrm{k3}}$ is the per-token estimator of $D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}})$.

V PROMPTS

All BALAR prompts consist of a *system prompt* and a *user prompt*. Each subsection below covers one prompt role and contains, for each of the three domains (AR-Bench-DC, AR-Bench-SP, iCraft-MD), the system prompt followed by the user prompt(s). Every prompt requests a short `reason` field before the structured output, implementing the lightweight chain-of-thought verification described in Appendix C.

V.1 USER SIMULATOR

AR-BENCH-DC

System prompt.

```
You are simulating a suspect being interrogated in a murder investigation. You must stay in
     character and answer based only on your character's knowledge and story. If you are the
      murderer, you will try to deflect suspicion while remaining consistent with your story
     . If you are innocent, answer honestly based on what your character knows.
```

With discrete choices.

```
<USER_CONTEXT>
{user_context}
</USER_CONTEXT>

<QUESTION>
{question}
</QUESTION>

<CHOICES_WITH_IDS>
{choices_with_ids}
</CHOICES_WITH_IDS>

Task:
You are a suspect being interrogated by a detective. Answer <QUESTION> in a way that is
     consistent with your character described in <USER_CONTEXT>. If you are the murderer,
     you may deflect or be evasive, but you must remain consistent with your story.

What to generate:
- reason : a short one-sentence explanation of why you chose the answer you did.
- answer_id : the id of the choice you are selecting as your answer to the question.

Constraints:
Use ONLY information that is supported by <USER_CONTEXT>.
answer_id must be one of the ids provided in <CHOICES_WITH_IDS>.

Output format:
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "answer_id": string
}}
```

Without discrete choices (free-form).

```
<USER_CONTEXT>
{user_context}
</USER_CONTEXT>

<QUESTION>
{question}
</QUESTION>

Task:
You are a suspect being interrogated by a detective. Answer <QUESTION> in a way that is
     consistent with your character described in <USER_CONTEXT>. If you are the murderer,
     you may deflect or be evasive, but you must remain consistent with your story.

What to generate:
- reason : a short one-sentence explanation of why you chose the answer you did.
- answer : your answer to the question.

Constraints:
```

```
Use ONLY information that is supported by <USER_CONTEXT>.
answer must be a natural language answer to the question.

Output format:
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "answer": string
}}
```

AR-BENCH-SP

System prompt.

```
You are the host of a thinking puzzle (situation puzzle). You know the hidden explanation
     behind the puzzle. Answer the solver's questions truthfully based on the solution.
     Never reveal the full solution directly; only confirm or deny specific aspects when
     asked.
```

With discrete choices.

```
<USER_CONTEXT>
{user_context}
</USER_CONTEXT>

<QUESTION>
{question}
</QUESTION>

<CHOICES_WITH_IDS>
{choices_with_ids}
</CHOICES_WITH_IDS>

Task:
You are the host of a thinking puzzle. A solver is asking you <QUESTION> to try to figure
     out the hidden explanation. Answer based on your knowledge of the puzzle solution
     described in <USER_CONTEXT>. Give a truthful answer---do not mislead, but do not give
     extra information.

What to generate:
- reason : a short one-sentence explanation of why you chose the answer you did.
- answer_id : the id of the choice you are selecting as your answer to the question.

Constraints:
Use ONLY information that is supported by <USER_CONTEXT>.
answer_id must be one of the ids provided in <CHOICES_WITH_IDS>.

Output format:
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "answer_id": string
}}
```

Without discrete choices (free-form).

```
<USER_CONTEXT>
{user_context}
</USER_CONTEXT>

<QUESTION>
{question}
</QUESTION>

Task:
You are the host of a thinking puzzle. A solver is asking you <QUESTION> to try to figure
     out the hidden explanation. Answer based on your knowledge of the puzzle solution
     described in <USER_CONTEXT>. Give a truthful answer---do not mislead, but do not
     volunteer extra information.

What to generate:
- reason : a short one-sentence explanation of why you chose the answer you did.
- answer : your answer to the question.

Constraints:
Use ONLY information that is supported by <USER_CONTEXT>.
```

```
answer must be a natural language answer to the question.

Output format:
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "answer": string
}}
```

iCRAFT-MD

System prompt.

```
You are simulating a patient being examined by a physician. Answer questions based ONLY on
     your medical record and symptoms. Use facts from your medical record. Do not give
     information that was not specifically asked about.
```

With discrete choices.

```
<USER_CONTEXT>
{user_context}
</USER_CONTEXT>

<QUESTION>
{question}
</QUESTION>

<CHOICES_WITH_IDS>
{choices_with_ids}
</CHOICES_WITH_IDS>

Task:
You are a patient being examined by a physician. Answer <QUESTION> in a way that is
     consistent with your medical record and symptoms described in <USER_CONTEXT>. Use facts
      from your medical record. Do not give information that was not specifically asked
     about.

What to generate:
- reason : a short one-sentence explanation of why you chose the answer you did.
- answer_id : the id of the choice you are selecting as your answer to the question.

Constraints:
Use ONLY information that is supported by <USER_CONTEXT>.
answer_id must be one of the ids provided in <CHOICES_WITH_IDS>.

Output format:
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "answer_id": string
}}
```

Without discrete choices (free-form).

```
<USER_CONTEXT>
{user_context}
</USER_CONTEXT>

<QUESTION>
{question}
</QUESTION>

Task:
You are a patient being examined by a physician. Answer <QUESTION> in a way that is
     consistent with your medical record and symptoms described in <USER_CONTEXT>. Use facts
      from your medical record. Do not give information that was not specifically asked
     about.

What to generate:
- reason : a short one-sentence explanation of why you chose the answer you did.
- answer : your answer to the question.

Constraints:
Use ONLY information that is supported by <USER_CONTEXT>.
answer must be a natural language answer to the question.
```

```
Output format:
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "answer": string
}}
```

V.2 STEP 1 — DIMENSION PROPOSAL

AR-BENCH-DC

System prompt.

```
You are an experienced detective analyzing a murder case. Your goal is to identify the key
     investigative dimensions that must be resolved to determine who the real murderer is.
```

User prompt.

```
<CASE_QUESTION>
{ambiguous_prompt}
</CASE_QUESTION>

<CASE_BACKGROUND>
{meta_context}
</CASE_BACKGROUND>

Task:
Identify the key investigative dimensions in the <CASE_QUESTION> that must be resolved to
     determine who the real murderer is.

Definition:
An investigative dimension is a specific aspect of the murder case (e.g., motive, alibi,
     access to the murder weapon, relationship to the victim) where multiple suspects could
     plausibly be implicated, and resolving it would narrow down the true murderer. Once the
      dimension's value is known, the case moves toward identifying a single suspect as the
     murderer.

What to generate:
- Produce a minimal, non-overlapping set of investigative dimensions.
- Produce exactly {num_initial_dims} investigative dimensions.
- Each dimension must correspond to a distinct, investigative uncertainty.
- If <CASE_BACKGROUND> already resolves a dimension, do not include it.
- If <CASE_BACKGROUND> proposes some investigative dimensions, use them.

For each investigative dimension, provide :
- reason: a short one-sentence explanation of why this dimension is critical for identifying
      the murderer.
- name: a short, specific label (e.g., "Motive", "Alibi at time of death", "Access to murder
      weapon", etc.)
- values: a list of plausible values (e.g., one per suspect or per scenario), no larger than
      {max_num_values_per_dim}, that this dimension could take in the context of the case.

Constraints:
- Use ONLY the information provided in <CASE_QUESTION> and <CASE_BACKGROUND>.
- Do NOT answer the <CASE_QUESTION> itself. Focus ONLY on identifying investigative
     dimensions.
- Do NOT rewrite or restate the <CASE_QUESTION> or <CASE_BACKGROUND>.

Output format:
Return STRICT JSON only with the following schema:
{{
    "dimensions": [
        {{
            "reason": string,
            "name": string,
            "values": [string, ...]
        }},
        ...
    ]
}}
```

AR-BENCH-SP

System prompt.

```
You are an expert thinking puzzle solver. Your goal is to identify the hidden dimensions of
     the puzzle---the unstated aspects of the scenario whose true values would explain the
     strange situation presented.
```

User prompt.

```
<PUZZLE>
{ambiguous_prompt}
</PUZZLE>

<PUZZLE_CONTEXT>
{meta_context}
</PUZZLE_CONTEXT>

Task:
Identify the hidden dimensions of the thinking puzzle in <PUZZLE> that must be uncovered to
     explain the strange scenario.

Definition:
A puzzle dimension is a hidden aspect of the scenario---such as a non-obvious word meaning,
     an unstated context, a surprising identity, or an unusual causal mechanism---where
     knowing its true value would explain the puzzle. Once the dimension's value is known,
     the puzzle moves toward a single coherent explanation.

What to generate:
- Produce a minimal, non-overlapping set of puzzle dimensions.
- Produce exactly {num_initial_dims} puzzle dimensions.
- Each dimension must correspond to a distinct hidden aspect of the puzzle.
- If <PUZZLE_CONTEXT> already resolves a dimension, do not include it.
- If <PUZZLE_CONTEXT> proposes some puzzle dimensions, use them.

For each puzzle dimension, provide :
- reason: a short one-sentence explanation of why this dimension is a key unknown in the
     puzzle.
- name: a short, specific label
- values: a list of plausible interpretations, no larger than {max_num_values_per_dim}, that
      this dimension could take.

Constraints:
- Use ONLY the information provided in <PUZZLE> and <PUZZLE_CONTEXT>.
- Do NOT solve the <PUZZLE> itself. Focus ONLY on identifying hidden dimensions.
- Do NOT rewrite or restate the <PUZZLE> or <PUZZLE_CONTEXT>.

Output format:
Return STRICT JSON only with the following schema:
{{
    "dimensions": [
        {{
            "reason": string,
            "name": string,
            "values": [string, ...]
        }},
        ...
    ]
}}
```

ICRAFT-MD

System prompt.

```
You are an experienced physician performing a diagnosis. Your goal is to identify the key
     clinical dimensions that must be clarified to arrive at the correct diagnosis or
     clinical decision.
```

User prompt.

```
<CLINICAL_QUESTION>
{ambiguous_prompt}
</CLINICAL_QUESTION>

<PATIENT_INFORMATION>
{meta_context}
</PATIENT_INFORMATION>

Task:
```

```
Identify the key clinical dimensions in the <CLINICAL_QUESTION> that must be clarified to
     arrive at the correct diagnosis or clinical decision.

Definition:
A clinical dimension is a specific clinical factor (e.g., symptom characterization, lab
     finding, risk factor, past medical history) where different values would point toward
     different diagnoses or clinical decisions. Once the dimension's value is known, the
     diagnosis narrows toward a single correct answer.

What to generate:
- Produce a minimal, non-overlapping set of clinical dimensions.
- Produce exactly {num_initial_dims} clinical dimensions.
- Each dimension must correspond to a distinct clinical uncertainty.
- If <PATIENT_INFORMATION> already resolves a dimension, do not include it.
- If <PATIENT_INFORMATION> proposes some clinical dimensions, use them.

For each clinical dimension, provide :
- reason: a short one-sentence explanation of why this clinical factor is discriminating
     between diagnoses.
- name: a short, specific clinical label
- values: a list of clinically plausible values, no larger than {max_num_values_per_dim},
     that this dimension could take.

Constraints:
- Use ONLY the information provided in <CLINICAL_QUESTION> and <PATIENT_INFORMATION>.
- Do NOT answer the <CLINICAL_QUESTION> itself. Focus ONLY on identifying clinical
     dimensions.
- Do NOT rewrite or restate the <CLINICAL_QUESTION> or <PATIENT_INFORMATION>.

Output format:
Return STRICT JSON only with the following schema:
{{
    "dimensions": [
        {{
            "reason": string,
            "name": string,
            "values": [string, ...]
        }},
        ...
    ]
}}
```

V.3 Step 2 — Prior Elicitation

AR-Bench-DC

System prompt.

```
You are an experienced detective forming initial hypotheses about a murder case based on the
      available evidence and case background.
```

User prompt.

```
<CASE_QUESTION>
{ambiguous_prompt}
</CASE_QUESTION>

<CASE_BACKGROUND>
{meta_context}
</CASE_BACKGROUND>

<DIMENSION_NAME>
{dimension_name}
</DIMENSION_NAME>

<DIMENSION_VALUE>
{dimension_value}
</DIMENSION_VALUE>

Task:
Given <CASE_QUESTION> and <CASE_BACKGROUND>, judge how likely the investigative dimension <
     DIMENSION_NAME> takes on the value <DIMENSION_VALUE>.

What to generate:
- reason: a short one-sentence explanation of why the <DIMENSION_NAME> is likely, unlikely,
     or neutral to take on the value <DIMENSION_VALUE>.
- label: one of "likely", "unlikely", or "neutral" according to the following definitions:
```

```
    - likely: <DIMENSION_VALUE> is explicitly stated, strongly implied, or is the most
        natural assumption given the evidence in the <CASE_QUESTION> and <CASE_BACKGROUND>.
    - neutral: <DIMENSION_VALUE> is plausible but not implied or supported by specific
        evidence in the <CASE_QUESTION> or <CASE_BACKGROUND>.
    - unlikely: <DIMENSION_VALUE> is contradicted by the <CASE_QUESTION> or <CASE_BACKGROUND
        >, or would require assumptions that are inconsistent with the available evidence.

Constraints:
- Use ONLY the information provided in <CASE_QUESTION> and <CASE_BACKGROUND>.
- Do NOT answer the <CASE_QUESTION> itself. Focus ONLY on judging the likelihood of the
    dimension value.
- Do NOT rewrite or restate the <CASE_QUESTION> or <CASE_BACKGROUND>.
- label must be one of "likely", "unlikely", or "neutral".

Output format:
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "label": string
}}
```

AR-BENCH-SP

System prompt.

```
You are an expert thinking puzzle solver forming initial hypotheses about the hidden aspects
     of a puzzle based on the scenario description.
```

User prompt.

```
<PUZZLE>
{ambiguous_prompt}
</PUZZLE>

<PUZZLE_CONTEXT>
{meta_context}
</PUZZLE_CONTEXT>

<DIMENSION_NAME>
{dimension_name}
</DIMENSION_NAME>

<DIMENSION_VALUE>
{dimension_value}
</DIMENSION_VALUE>

Task:
Given <PUZZLE> and <PUZZLE_CONTEXT>, judge how likely the puzzle dimension <DIMENSION_NAME>
    takes on the value <DIMENSION_VALUE>.

What to generate:
- reason: a short one-sentence explanation of why the <DIMENSION_NAME> is likely, unlikely,
    or neutral to take on the value <DIMENSION_VALUE>.
- label: one of "likely", "unlikely", or "neutral" according to the following definitions:
    - likely: <DIMENSION_VALUE> is explicitly suggested by, strongly implied by, or is the
        most natural interpretation given the clues in the <PUZZLE> and <PUZZLE_CONTEXT>.
    - neutral: <DIMENSION_VALUE> is a plausible interpretation but not implied or supported
        by specific clues in the <PUZZLE> or <PUZZLE_CONTEXT>.
    - unlikely: <DIMENSION_VALUE> is contradicted by the <PUZZLE> or <PUZZLE_CONTEXT>, or
        would require assumptions that are inconsistent with the scenario.

Constraints:
- Use ONLY the information provided in <PUZZLE> and <PUZZLE_CONTEXT>.
- Do NOT solve the <PUZZLE> itself. Focus ONLY on judging the likelihood of the dimension
    value.
- Do NOT rewrite or restate the <PUZZLE> or <PUZZLE_CONTEXT>.
- label must be one of "likely", "unlikely", or "neutral".

Output format:
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "label": string
}}
```

iCraft-MD

System prompt.

```
You are an experienced physician forming initial clinical hypotheses based on the available
     patient information. Use your clinical knowledge to assess the likelihood of different
     clinical findings.
```

User prompt.

```
<CLINICAL_QUESTION>
{ambiguous_prompt}
</CLINICAL_QUESTION>

<PATIENT_INFORMATION>
{meta_context}
</PATIENT_INFORMATION>

<DIMENSION_NAME>
{dimension_name}
</DIMENSION_NAME>

<DIMENSION_VALUE>
{dimension_value}
</DIMENSION_VALUE>

Task:
Given <CLINICAL_QUESTION> and <PATIENT_INFORMATION>, judge how likely the clinical dimension
      <DIMENSION_NAME> takes on the value <DIMENSION_VALUE>.

What to generate:
- reason: a short one-sentence explanation of why the <DIMENSION_NAME> is likely, unlikely,
     or neutral to take on the value <DIMENSION_VALUE>.
- label: one of "likely", "unlikely", or "neutral" according to the following definitions:
    - likely: <DIMENSION_VALUE> is explicitly stated, strongly implied, or is the most
         natural clinical assumption given the patient information in <CLINICAL_QUESTION>
         and <PATIENT_INFORMATION>.
    - neutral: <DIMENSION_VALUE> is clinically plausible but not implied or supported by
         specific evidence in the <CLINICAL_QUESTION> or <PATIENT_INFORMATION>.
    - unlikely: <DIMENSION_VALUE> is contradicted by the <CLINICAL_QUESTION> or <
         PATIENT_INFORMATION>, or would require assumptions that are inconsistent with the
         patient's presentation.

Constraints:
- Use ONLY the information provided in <CLINICAL_QUESTION> and <PATIENT_INFORMATION>.
- Do NOT answer the <CLINICAL_QUESTION> itself. Focus ONLY on judging the likelihood of the
     dimension value.
- Do NOT rewrite or restate the <CLINICAL_QUESTION> or <PATIENT_INFORMATION>.
- label must be one of "likely", "unlikely", or "neutral".

Output format:
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "label": string
}}
```

V.4 Step 3 — Question Generation

AR-Bench-DC

System prompt.

```
You are an experienced detective preparing interrogation questions for suspects in a murder
     investigation. Your questions should be designed to reveal inconsistencies, uncover
     motives, and verify alibis.
```

User prompt.

```
<CASE_QUESTION>
{ambiguous_prompt}
</CASE_QUESTION>

<CASE_BACKGROUND>
{meta_context}
```

```
</CASE_BACKGROUND>

<INVESTIGATIVE_DIMENSIONS>
{dimensions_with_values}
</INVESTIGATIVE_DIMENSIONS>

Task:
Given <CASE_QUESTION>, <CASE_BACKGROUND>, and <INVESTIGATIVE_DIMENSIONS>, generate exactly {
     num_initial_questions} interrogation questions to ask the suspects that would help
     identify the real murderer. Each question should target one or more <
     INVESTIGATIVE_DIMENSIONS> and have multiple-choice answers.

Definition:
<INVESTIGATIVE_DIMENSIONS> is a list of investigative dimensions, where each dimension has a
      name and a list of possible values it could take. An investigative dimension is a
     specific aspect of the murder case where multiple suspects could plausibly be
     implicated, and resolving it would narrow down the true murderer.

What to generate:
For each of the {num_initial_questions} questions, provide:
- reason: a short one-sentence explanation of why this question would help identify the
     murderer.
- question: the text of the interrogation question.
- choices: a list of multiple-choice answer options for the question, no larger than {
     max_choices_per_question}.

Constraints:
- Use ONLY the information provided in <CASE_QUESTION>, <CASE_BACKGROUND>, and <
     INVESTIGATIVE_DIMENSIONS>.
- Do NOT answer the <CASE_QUESTION> itself. Focus ONLY on generating interrogation questions
     .
- Do NOT rewrite or restate the <CASE_QUESTION> or <CASE_BACKGROUND>.
- Each question must be designed to elicit information about one or more of the dimensions
     in <INVESTIGATIVE_DIMENSIONS>.
- Each question must have multiple-choice answers.

Output format:
Return STRICT JSON only with the following schema:
{{
    "questions": [
        {{
            "reason": string,
            "question": string,
            "choices": [string, ...]
        }},
        ...
    ]
}}
```

AR-BENCH-SP

System prompt.

```
You are an expert thinking puzzle solver. Generate clarifying questions to ask the puzzle
     host that will help you uncover the hidden explanation. Good puzzle questions test
     specific hypotheses about what is really going on in the scenario.
```

User prompt.

```
<PUZZLE>
{ambiguous_prompt}
</PUZZLE>

<PUZZLE_CONTEXT>
{meta_context}
</PUZZLE_CONTEXT>

<PUZZLE_DIMENSIONS>
{dimensions_with_values}
</PUZZLE_DIMENSIONS>

Task:
Given <PUZZLE>, <PUZZLE_CONTEXT>, and <PUZZLE_DIMENSIONS>, generate exactly {
     num_initial_questions} clarifying questions to ask the puzzle host that would help
     uncover the hidden explanation. Each question should target one or more <
     PUZZLE_DIMENSIONS> and have "yes"/"no" answers.
```

```
Definition:
<PUZZLE_DIMENSIONS> is a list of puzzle dimensions, where each dimension has a name and a
     list of possible values it could take. A puzzle dimension is a hidden aspect of the
     scenario where knowing its true value would explain the puzzle.

What to generate:
For each of the {num_initial_questions} questions, provide:
- reason: a short one-sentence explanation of why this question would help solve the puzzle.
- question: the text of the clarifying question to ask the puzzle host.
- choices: ["yes", "no"] as the {max_choices_per_question} multiple-choice answer options
     for the question.

Constraints:
- Use ONLY the information provided in <PUZZLE>, <PUZZLE_CONTEXT>, and <PUZZLE_DIMENSIONS>.
- Do NOT solve the <PUZZLE> itself. Focus ONLY on generating clarifying questions.
- Do NOT rewrite or restate the <PUZZLE> or <PUZZLE_CONTEXT>.
- Each question must be designed to elicit information about one or more of the dimensions
     in <PUZZLE_DIMENSIONS>.
- Each question must have "yes"/"no" answers.
- Keep each question short: at most 20 words.
- Keep each reason short: at most 15 words.

Output format:
Return STRICT JSON only with the following schema:
{{
    "questions": [
        {{
            "reason": string,
            "question": string,
            "choices": ["yes", "no"]
        }},
        ...
    ]
}}
```

iCRAFT-MD

System prompt.

```
You are an experienced physician conducting a patient interview to gather clinical
     information for a diagnosis. Your questions should be targeted, clinically relevant,
     and designed to discriminate between competing diagnoses.
```

User prompt.

```
<CLINICAL_QUESTION>
{ambiguous_prompt}
</CLINICAL_QUESTION>

<PATIENT_INFORMATION>
{meta_context}
</PATIENT_INFORMATION>

<CLINICAL_DIMENSIONS>
{dimensions_with_values}
</CLINICAL_DIMENSIONS>

Task:
Given <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, and <CLINICAL_DIMENSIONS>, generate
     exactly {num_initial_questions} clinical questions to ask the patient that would help
     arrive at the correct diagnosis. Each question should target one or more <
     CLINICAL_DIMENSIONS> and have multiple-choice answers.

Definition:
<CLINICAL_DIMENSIONS> is a list of clinical dimensions, where each dimension has a name and
     a list of possible values it could take. A clinical dimension is a specific clinical
     factor where different values would point toward different diagnoses.

What to generate:
For each of the {num_initial_questions} questions, provide:
- reason: a short one-sentence explanation of why this question would help narrow the
     diagnosis.
- question: the text of the clinical question to ask the patient.
- choices: a list of multiple-choice answer options for the question, no larger than {
     max_choices_per_question}.

Constraints:
```

```
- Use ONLY the information provided in <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, and <
    CLINICAL_DIMENSIONS>.
- Do NOT answer the <CLINICAL_QUESTION> itself. Focus ONLY on generating clinical questions.
- Do NOT rewrite or restate the <CLINICAL_QUESTION> or <PATIENT_INFORMATION>.
- Each question must be designed to elicit information about one or more of the dimensions
    in <CLINICAL_DIMENSIONS>.
- Each question must have multiple-choice answers.

Output format:
Return STRICT JSON only with the following schema:
{{
    "questions": [
        {{
            "reason": string,
            "question": string,
            "choices": [string, ...]
        }},
        ...
    ]
}}
```

V.5 Step 4 — Likelihood Table Construction

AR-Bench-DC

System prompt.

```
You are an experienced detective evaluating how a suspect would likely respond to an
    interrogation question under different assumptions about the case. Consider that guilty
     suspects may deflect, lie, or give evasive answers, while innocent suspects will
    answer based on their genuine knowledge.
```

Without conversation history.

```
<CASE_QUESTION>
{ambiguous_prompt}
</CASE_QUESTION>

<CASE_BACKGROUND>
{meta_context}
</CASE_BACKGROUND>

<SUSPECT_INFO>
{user_info}
</SUSPECT_INFO>

<DIMENSION_NAME>
{dimension_name}
</DIMENSION_NAME>

<DIMENSION_VALUES_WITH_IDS>
{dimension_values_with_ids}
</DIMENSION_VALUES_WITH_IDS>

<QUESTION>
{question_text}
</QUESTION>

<QUESTION_CHOICES_WITH_IDS>
{question_choices_with_ids}
</QUESTION_CHOICES_WITH_IDS>

Definition:
An investigative dimension is a specific aspect of the murder case where multiple suspects
    could plausibly be implicated, and resolving it would narrow down the true murderer.
<DIMENSION_VALUES_WITH_IDS> is a list of dicts with "id" and "text" fields. Each dict
    corresponds to a possible value that the <DIMENSION_NAME> could take.
<QUESTION_CHOICES_WITH_IDS> is a list of dicts with "id" and "text" fields. Each dict
    corresponds to a multiple-choice answer option for the question.
Let values[i] be the i-th element of <DIMENSION_VALUES_WITH_IDS>.
Let choices[j] be the j-th element of <QUESTION_CHOICES_WITH_IDS>.

Task (row-major order):
For i = 0..len(values)-1:
  For j = 0..len(choices)-1:
    - Assume the true state of the case is <DIMENSION_NAME> = values[i]["text"].
    - Impersonate the suspect described in <SUSPECT_INFO>.
```

```
    - Judge how likely it is that this suspect would answer the question <QUESTION> with
        choices[j]["text"] under that assumption. Consider that a guilty suspect may try to
         deflect or mislead.

What to generate:
For i = 0..len(values)-1:
  For j = 0..len(choices)-1:
    - question_choice_id: the id of the question choice being evaluated, i.e. choices[j]["id
        "]
    - dimension_value_id: the id of the dimension value being evaluated, i.e. values[i]["id
        "]
    - reason: a short one-sentence explanation of why choices[j] is labeled likely/neutral/
        unlikely and why the other two labels were not chosen.
    - label: one of "likely", "neutral", or "unlikely" according to the following
        definitions:
      - "likely": Given <DIMENSION_NAME> = values[i]["text"] and the suspect acts according
          to <SUSPECT_INFO>, the suspect is expected to give choices[j]["text"] for <
          QUESTION>.
      - "neutral": Given <DIMENSION_NAME> = values[i]["text"] and the suspect acts according
           to <SUSPECT_INFO>, choices[j]["text"] is plausible but not specifically
          supported; there is insufficient evidence to say that the suspect would or would
          not prefer it over other choices.
      - "unlikely": Given <DIMENSION_NAME> = values[i]["text"] and the suspect acts
          according to <SUSPECT_INFO>, the suspect is not expected to give choices[j]["text
          "] for <QUESTION>.

Constraints:
- Use ONLY the information provided in <CASE_QUESTION>, <CASE_BACKGROUND>, and <SUSPECT_INFO
    >.
- Do NOT answer the <CASE_QUESTION> itself. Focus ONLY on judging the likelihood of the
    question choices under different assumptions about the dimension value.
- Do NOT rewrite or restate the <CASE_QUESTION>, <CASE_BACKGROUND>, or <SUSPECT_INFO>.
- label must be one of "likely", "neutral", or "unlikely".
- The output must include an entry for every combination of dimension value and question
    choice.

Output format:
Return STRICT JSON only with the following schema:
{{
    "evaluations": [
        [
            {{
                "question_choice_id": string,
                "dimension_value_id": string,
                "reason": string,
                "label": string
            }},
            ... // one object for each question choice
        ],
        ... // one array for each dimension value
    ]
}}
The "evaluations" field must contain exactly {num_dimension_values} arrays (one per
    dimension value).
Each inner array must contain exactly {num_question_choices} objects (one per question
    choice).
```

With conversation history.

```
<CASE_QUESTION>
{ambiguous_prompt}
</CASE_QUESTION>

<CASE_BACKGROUND>
{meta_context}
</CASE_BACKGROUND>

<SUSPECT_INFO>
{user_info}
</SUSPECT_INFO>

<INTERROGATION_LOG>
{conversation_log}
</INTERROGATION_LOG>

<DIMENSION_NAME>
{dimension_name}
</DIMENSION_NAME>
```

```
<DIMENSION_VALUES_WITH_IDS>
{dimension_values_with_ids}
</DIMENSION_VALUES_WITH_IDS>

<QUESTION>
{question_text}
</QUESTION>

<QUESTION_CHOICES_WITH_IDS>
{question_choices_with_ids}
</QUESTION_CHOICES_WITH_IDS>

Definition:
An investigative dimension is a specific aspect of the murder case where multiple suspects
     could plausibly be implicated, and resolving it would narrow down the true murderer.
<DIMENSION_VALUES_WITH_IDS> is a list of dicts with "id" and "text" fields. Each dict
     corresponds to a possible value that the <DIMENSION_NAME> could take.
<QUESTION_CHOICES_WITH_IDS> is a list of dicts with "id" and "text" fields. Each dict
     corresponds to a multiple-choice answer option for the question.
<INTERROGATION_LOG> is a list of dicts with "question_text", "user_name", and "user_answer"
     fields, representing the history of the interrogation between the detective and the
     suspects up to this point. This information may provide additional context and reveal
     inconsistencies.
Let values[i] be the i-th element of <DIMENSION_VALUES_WITH_IDS>.
Let choices[j] be the j-th element of <QUESTION_CHOICES_WITH_IDS>.

Task (row-major order):
For i = 0..len(values)-1:
  For j = 0..len(choices)-1:
    - Assume the true state of the case is <DIMENSION_NAME> = values[i]["text"].
    - Impersonate the suspect described in <SUSPECT_INFO>.
    - Judge how likely it is that this suspect would answer the question <QUESTION> with
        choices[j]["text"] under that assumption. Consider that a guilty suspect may try to
         deflect or mislead.

What to generate:
For i = 0..len(values)-1:
  For j = 0..len(choices)-1:
    - question_choice_id: the id of the question choice being evaluated, i.e. choices[j]["id
        "]
    - dimension_value_id: the id of the dimension value being evaluated, i.e. values[i]["id
        "]
    - reason: a short one-sentence explanation of why choices[j] is labeled likely/neutral/
        unlikely and why the other two labels were not chosen.
    - label: one of "likely", "neutral", or "unlikely" according to the following
        definitions:
      - "likely": Given <DIMENSION_NAME> = values[i]["text"] and the suspect acts according
          to <SUSPECT_INFO>, the suspect is expected to give choices[j]["text"] for <
          QUESTION>.
      - "neutral": Given <DIMENSION_NAME> = values[i]["text"] and the suspect acts according
           to <SUSPECT_INFO>, choices[j]["text"] is plausible but not specifically
          supported; there is insufficient evidence to say that the suspect would or would
          not prefer it over other choices.
      - "unlikely": Given <DIMENSION_NAME> = values[i]["text"] and the suspect acts
          according to <SUSPECT_INFO>, the suspect is not expected to give choices[j]["text
          "] for <QUESTION>.

Constraints:
- Use ONLY the information provided in <CASE_QUESTION>, <CASE_BACKGROUND>, <SUSPECT_INFO>,
    and <INTERROGATION_LOG>.
- Do NOT answer the <CASE_QUESTION> itself. Focus ONLY on judging the likelihood of the
    question choices under different assumptions about the dimension value.
- Do NOT rewrite or restate the <CASE_QUESTION>, <CASE_BACKGROUND>, <SUSPECT_INFO>, or <
    INTERROGATION_LOG>.
- label must be one of "likely", "neutral", or "unlikely".
- The output must include an entry for every combination of dimension value and question
    choice.

Output format:
Return STRICT JSON only with the following schema:
{{
    "evaluations": [
        [
            {{
                "question_choice_id": string,
                "dimension_value_id": string,
                "reason": string,
                "label": string
            }},
            ... // one object for each question choice
        ],
```

```
        ... // one array for each dimension value
    ]
}}
The "evaluations" field must contain exactly {num_dimension_values} arrays (one per
     dimension value).
Each inner array must contain exactly {num_question_choices} objects (one per question
     choice).
```

AR-BENCH-SP

System prompt.

```
You are an expert thinking puzzle analyst. Evaluate how a puzzle host who knows the hidden
     explanation would likely respond to a solver's question under different assumptions
     about the puzzle's hidden aspects. The host answers truthfully, without giving extra
     information.
```

Without conversation history.

```
<PUZZLE>
{ambiguous_prompt}
</PUZZLE>

<PUZZLE_CONTEXT>
{meta_context}
</PUZZLE_CONTEXT>

<HOST_INFO>
{user_info}
</HOST_INFO>

<DIMENSION_NAME>
{dimension_name}
</DIMENSION_NAME>

<DIMENSION_VALUES_WITH_IDS>
{dimension_values_with_ids}
</DIMENSION_VALUES_WITH_IDS>

<QUESTION>
{question_text}
</QUESTION>

<QUESTION_CHOICES_WITH_IDS>
{question_choices_with_ids}
</QUESTION_CHOICES_WITH_IDS>

Definition:
A puzzle dimension is a hidden aspect of the scenario where knowing its true value would
     explain the puzzle.
<DIMENSION_VALUES_WITH_IDS> is a list of dicts with "id" and "text" fields. Each dict
     corresponds to a possible value that the <DIMENSION_NAME> could take.
<QUESTION_CHOICES_WITH_IDS> is a list of dicts with "id" and "text" fields. Each dict
     corresponds to a multiple-choice answer option for the question.
Let values[i] be the i-th element of <DIMENSION_VALUES_WITH_IDS>.
Let choices[j] be the j-th element of <QUESTION_CHOICES_WITH_IDS>.

Task (row-major order):
For i = 0..len(values)-1:
  For j = 0..len(choices)-1:
    - Assume the hidden explanation is such that <DIMENSION_NAME> = values[i]["text"].
    - Impersonate the puzzle host described in <HOST_INFO>, who knows the hidden explanation
        .
    - Judge how likely it is that this host would answer the question <QUESTION> with
        choices[j]["text"] under that assumption. The host answers truthfully.

What to generate:
For i = 0..len(values)-1:
  For j = 0..len(choices)-1:
    - question_choice_id: the id of the question choice being evaluated, i.e. choices[j]["id
        "]
    - dimension_value_id: the id of the dimension value being evaluated, i.e. values[i]["id
        "]
    - reason: a short one-sentence explanation of why choices[j] is labeled likely/neutral/
        unlikely and why the other two labels were not chosen.
    - label: one of "likely", "neutral", or "unlikely" according to the following
        definitions:
```

```
      - "likely": Given <DIMENSION_NAME> = values[i]["text"] and the host acts according to
          <HOST_INFO>, the host is expected to give choices[j]["text"] for <QUESTION>.
      - "neutral": Given <DIMENSION_NAME> = values[i]["text"] and the host acts according to
           <HOST_INFO>, choices[j]["text"] is plausible but not specifically supported;
          there is insufficient evidence to say that the host would or would not prefer it
          over other choices.
      - "unlikely": Given <DIMENSION_NAME> = values[i]["text"] and the host acts according
          to <HOST_INFO>, the host is not expected to give choices[j]["text"] for <QUESTION
          >.

Constraints:
- Use ONLY the information provided in <PUZZLE>, <PUZZLE_CONTEXT>, and <HOST_INFO>.
- Do NOT solve the <PUZZLE> itself. Focus ONLY on judging the likelihood of the question
    choices under different assumptions about the dimension value.
- Do NOT rewrite or restate the <PUZZLE>, <PUZZLE_CONTEXT>, or <HOST_INFO>.
- label must be one of "likely", "neutral", or "unlikely".
- The output must include an entry for every combination of dimension value and question
    choice.

Output format:
Return STRICT JSON only with the following schema:
{{
    "evaluations": [
        [
            {{
                "question_choice_id": string,
                "dimension_value_id": string,
                "reason": string,
                "label": string
            }},
            ... // one object for each question choice
        ],
        ... // one array for each dimension value
    ]
}}
The "evaluations" field must contain exactly {num_dimension_values} arrays (one per
    dimension value).
Each inner array must contain exactly {num_question_choices} objects (one per question
    choice).
```

With conversation history.

```
<PUZZLE>
{ambiguous_prompt}
</PUZZLE>

<PUZZLE_CONTEXT>
{meta_context}
</PUZZLE_CONTEXT>

<HOST_INFO>
{user_info}
</HOST_INFO>

<CONVERSATION_LOG>
{conversation_log}
</CONVERSATION_LOG>

<DIMENSION_NAME>
{dimension_name}
</DIMENSION_NAME>

<DIMENSION_VALUES_WITH_IDS>
{dimension_values_with_ids}
</DIMENSION_VALUES_WITH_IDS>

<QUESTION>
{question_text}
</QUESTION>

<QUESTION_CHOICES_WITH_IDS>
{question_choices_with_ids}
</QUESTION_CHOICES_WITH_IDS>

Definition:
A puzzle dimension is a hidden aspect of the scenario where knowing its true value would
    explain the puzzle.
<DIMENSION_VALUES_WITH_IDS> is a list of dicts with "id" and "text" fields. Each dict
    corresponds to a possible value that the <DIMENSION_NAME> could take.
```

```
<QUESTION_CHOICES_WITH_IDS> is a list of dicts with "id" and "text" fields. Each dict
     corresponds to a multiple-choice answer option for the question.
<CONVERSATION_LOG> is a list of dicts with "question_text", "user_name", and "user_answer"
     fields, representing the history of the conversation between the solver and the host up
      to this point. This information may provide additional clues.
Let values[i] be the i-th element of <DIMENSION_VALUES_WITH_IDS>.
Let choices[j] be the j-th element of <QUESTION_CHOICES_WITH_IDS>.

Task (row-major order):
For i = 0..len(values)-1:
  For j = 0..len(choices)-1:
    - Assume the hidden explanation is such that <DIMENSION_NAME> = values[i]["text"].
    - Impersonate the puzzle host described in <HOST_INFO>, who knows the hidden explanation
        .
    - Judge how likely it is that this host would answer the question <QUESTION> with
        choices[j]["text"] under that assumption. The host answers truthfully.

What to generate:
For i = 0..len(values)-1:
  For j = 0..len(choices)-1:
    - question_choice_id: the id of the question choice being evaluated, i.e. choices[j]["id
        "]
    - dimension_value_id: the id of the dimension value being evaluated, i.e. values[i]["id
        "]
    - reason: a short one-sentence explanation of why choices[j] is labeled likely/neutral/
        unlikely and why the other two labels were not chosen.
    - label: one of "likely", "neutral", or "unlikely" according to the following
        definitions:
      - "likely": Given <DIMENSION_NAME> = values[i]["text"] and the host acts according to
          <HOST_INFO>, the host is expected to give choices[j]["text"] for <QUESTION>.
      - "neutral": Given <DIMENSION_NAME> = values[i]["text"] and the host acts according to
           <HOST_INFO>, choices[j]["text"] is plausible but not specifically supported;
          there is insufficient evidence to say that the host would or would not prefer it
          over other choices.
      - "unlikely": Given <DIMENSION_NAME> = values[i]["text"] and the host acts according
          to <HOST_INFO>, the host is not expected to give choices[j]["text"] for <QUESTION
          >.

Constraints:
- Use ONLY the information provided in <PUZZLE>, <PUZZLE_CONTEXT>, <HOST_INFO>, and <
     CONVERSATION_LOG>.
- Do NOT solve the <PUZZLE> itself. Focus ONLY on judging the likelihood of the question
     choices under different assumptions about the dimension value.
- Do NOT rewrite or restate the <PUZZLE>, <PUZZLE_CONTEXT>, <HOST_INFO>, or <
     CONVERSATION_LOG>.
- label must be one of "likely", "neutral", or "unlikely".
- The output must include an entry for every combination of dimension value and question
     choice.

Output format:
Return STRICT JSON only with the following schema:
{{
    "evaluations": [
        [
            {{
                "question_choice_id": string,
                "dimension_value_id": string,
                "reason": string,
                "label": string
            }},
            ... // one object for each question choice
        ],
        ... // one array for each dimension value
    ]
}}
The "evaluations" field must contain exactly {num_dimension_values} arrays (one per
     dimension value).
Each inner array must contain exactly {num_question_choices} objects (one per question
     choice).
```

ICRAFT-MD

System prompt.

```
You are an experienced physician evaluating how a patient would likely respond to a clinical
      question under different assumptions about their underlying condition.
```

Without conversation history.

```
<CLINICAL_QUESTION>
{ambiguous_prompt}
</CLINICAL_QUESTION>

<PATIENT_INFORMATION>
{meta_context}
</PATIENT_INFORMATION>

<PATIENT_PROFILE>
{user_info}
</PATIENT_PROFILE>

<DIMENSION_NAME>
{dimension_name}
</DIMENSION_NAME>

<DIMENSION_VALUES_WITH_IDS>
{dimension_values_with_ids}
</DIMENSION_VALUES_WITH_IDS>

<QUESTION>
{question_text}
</QUESTION>

<QUESTION_CHOICES_WITH_IDS>
{question_choices_with_ids}
</QUESTION_CHOICES_WITH_IDS>

Definition:
A clinical dimension is a specific clinical factor where different values would point toward
      different diagnoses or clinical decisions.
<DIMENSION_VALUES_WITH_IDS> is a list of dicts with "id" and "text" fields. Each dict
     corresponds to a possible value that the <DIMENSION_NAME> could take.
<QUESTION_CHOICES_WITH_IDS> is a list of dicts with "id" and "text" fields. Each dict
     corresponds to a multiple-choice answer option for the question.
Let values[i] be the i-th element of <DIMENSION_VALUES_WITH_IDS>.
Let choices[j] be the j-th element of <QUESTION_CHOICES_WITH_IDS>.

Task (row-major order):
For i = 0..len(values)-1:
  For j = 0..len(choices)-1:
    - Assume the patient's true clinical state is <DIMENSION_NAME> = values[i]["text"].
    - Impersonate the patient described in <PATIENT_PROFILE>.
    - Judge how likely it is that this patient would answer the question <QUESTION> with
        choices[j]["text"] under that assumption.

What to generate:
For i = 0..len(values)-1:
  For j = 0..len(choices)-1:
    - question_choice_id: the id of the question choice being evaluated, i.e. choices[j]["id
        "]
    - dimension_value_id: the id of the dimension value being evaluated, i.e. values[i]["id
        "]
    - reason: a short one-sentence explanation of why choices[j] is labeled likely/neutral/
        unlikely and why the other two labels were not chosen.
    - label: one of "likely", "neutral", or "unlikely" according to the following
        definitions:
      - "likely": Given <DIMENSION_NAME> = values[i]["text"] and the patient acts according
          to <PATIENT_PROFILE>, the patient is expected to give choices[j]["text"] for <
          QUESTION>.
      - "neutral": Given <DIMENSION_NAME> = values[i]["text"] and the patient acts according
           to <PATIENT_PROFILE>, choices[j]["text"] is plausible but not specifically
          supported; there is insufficient evidence to say that the patient would or would
          not prefer it over other choices.
      - "unlikely": Given <DIMENSION_NAME> = values[i]["text"] and the patient acts
          according to <PATIENT_PROFILE>, the patient is not expected to give choices[j]["
          text"] for <QUESTION>.

Constraints:
- Use ONLY the information provided in <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, and <
    PATIENT_PROFILE>.
- Do NOT answer the <CLINICAL_QUESTION> itself. Focus ONLY on judging the likelihood of the
    question choices under different assumptions about the dimension value.
- Do NOT rewrite or restate the <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, or <
    PATIENT_PROFILE>.
- label must be one of "likely", "neutral", or "unlikely".
- The output must include an entry for every combination of dimension value and question
    choice.
```

```
Output format:
Return STRICT JSON only with the following schema:
{{
    "evaluations": [
        [
            {{
                "question_choice_id": string,
                "dimension_value_id": string,
                "reason": string,
                "label": string
            }},
            ... // one object for each question choice
        ],
        ... // one array for each dimension value
    ]
}}
The "evaluations" field must contain exactly {num_dimension_values} arrays (one per
     dimension value).
Each inner array must contain exactly {num_question_choices} objects (one per question
     choice).
```

With conversation history.

```
<CLINICAL_QUESTION>
{ambiguous_prompt}
</CLINICAL_QUESTION>

<PATIENT_INFORMATION>
{meta_context}
</PATIENT_INFORMATION>

<PATIENT_PROFILE>
{user_info}
</PATIENT_PROFILE>

<CLINICAL_INTERVIEW_LOG>
{conversation_log}
</CLINICAL_INTERVIEW_LOG>

<DIMENSION_NAME>
{dimension_name}
</DIMENSION_NAME>

<DIMENSION_VALUES_WITH_IDS>
{dimension_values_with_ids}
</DIMENSION_VALUES_WITH_IDS>

<QUESTION>
{question_text}
</QUESTION>

<QUESTION_CHOICES_WITH_IDS>
{question_choices_with_ids}
</QUESTION_CHOICES_WITH_IDS>

Definition:
A clinical dimension is a specific clinical factor where different values would point toward
      different diagnoses or clinical decisions.
<DIMENSION_VALUES_WITH_IDS> is a list of dicts with "id" and "text" fields. Each dict
     corresponds to a possible value that the <DIMENSION_NAME> could take.
<QUESTION_CHOICES_WITH_IDS> is a list of dicts with "id" and "text" fields. Each dict
     corresponds to a multiple-choice answer option for the question.
<CLINICAL_INTERVIEW_LOG> is a list of dicts with "question_text", "user_name", and "
     user_answer" fields, representing the history of the clinical interview between the
     physician and the patient up to this point. This information may reveal additional
     symptoms or clinical details.
Let values[i] be the i-th element of <DIMENSION_VALUES_WITH_IDS>.
Let choices[j] be the j-th element of <QUESTION_CHOICES_WITH_IDS>.

Task (row-major order):
For i = 0..len(values)-1:
  For j = 0..len(choices)-1:
    - Assume the patient's true clinical state is <DIMENSION_NAME> = values[i]["text"].
    - Impersonate the patient described in <PATIENT_PROFILE>.
    - Judge how likely it is that this patient would answer the question <QUESTION> with
        choices[j]["text"] under that assumption.

What to generate:
For i = 0..len(values)-1:
```

```
  For j = 0..len(choices)-1:
    - question_choice_id: the id of the question choice being evaluated, i.e. choices[j]["id
        "]
    - dimension_value_id: the id of the dimension value being evaluated, i.e. values[i]["id
        "]
    - reason: a short one-sentence explanation of why choices[j] is labeled likely/neutral/
        unlikely and why the other two labels were not chosen.
    - label: one of "likely", "neutral", or "unlikely" according to the following
        definitions:
      - "likely": Given <DIMENSION_NAME> = values[i]["text"] and the patient acts according
          to <PATIENT_PROFILE>, the patient is expected to give choices[j]["text"] for <
          QUESTION>.
      - "neutral": Given <DIMENSION_NAME> = values[i]["text"] and the patient acts according
           to <PATIENT_PROFILE>, choices[j]["text"] is plausible but not specifically
          supported; there is insufficient evidence to say that the patient would or would
          not prefer it over other choices.
      - "unlikely": Given <DIMENSION_NAME> = values[i]["text"] and the patient acts
          according to <PATIENT_PROFILE>, the patient is not expected to give choices[j]["
          text"] for <QUESTION>.

Constraints:
- Use ONLY the information provided in <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, <
    PATIENT_PROFILE>, and <CLINICAL_INTERVIEW_LOG>.
- Do NOT answer the <CLINICAL_QUESTION> itself. Focus ONLY on judging the likelihood of the
    question choices under different assumptions about the dimension value.
- Do NOT rewrite or restate the <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, <PATIENT_PROFILE
    >, or <CLINICAL_INTERVIEW_LOG>.
- label must be one of "likely", "neutral", or "unlikely".
- The output must include an entry for every combination of dimension value and question
    choice.

Output format:
Return STRICT JSON only with the following schema:
{{
    "evaluations": [
        [
            {{
                "question_choice_id": string,
                "dimension_value_id": string,
                "reason": string,
                "label": string
            }},
            ... // one object for each question choice
        ],
        ... // one array for each dimension value
    ]
}}
The "evaluations" field must contain exactly {num_dimension_values} arrays (one per
    dimension value).
Each inner array must contain exactly {num_question_choices} objects (one per question
    choice).
```

V.6 SOFT-MAP SCORING

AR-BENCH-DC

System prompt.

```
You are an experienced detective analyzing a suspect's response to an interrogation question
    . Your task is to judge how well the suspect's answer maps to each of the predefined
    answer choices.
```

User prompt.

```
<QUESTION>
{question}
</QUESTION>

<CHOICES_WITH_IDS>
{choices_with_ids}
</CHOICES_WITH_IDS>

<SUSPECT_ANSWER>
{user_answer}
</SUSPECT_ANSWER>

Definition:
```

```
<CHOICES_WITH_IDS> is a list of dicts with "id" and "value" fields. Each dict corresponds to
      a multiple-choice answer option for the question.
Let choices[i] be the i-th element of <CHOICES_WITH_IDS>.

Task:
Judge how well the <SUSPECT_ANSWER> maps to each of the choices in <CHOICES_WITH_IDS> for
     the question <QUESTION>.

What to generate:
For i = 0..len(choices)-1:
- choice_id: the id of the question choice being evaluated, i.e. choices[i]["id"]
- reason: a short one-sentence explanation of choices[i]["value"] is likely/neutral/unlikely
      given the <SUSPECT_ANSWER>.
- label: one of "likely", "neutral", or "unlikely" according to the following definitions:
    -"likely": choices[i]["value"] aligns well with the <SUSPECT_ANSWER> and fits it better
         than most other choices.
    -"neutral": choices[i]["value"] is neither clearly supported nor clearly contradicted by
          the <SUSPECT_ANSWER>.
    -"unlikely": choices[i]["value"] fits the <SUSPECT_ANSWER> worse than other choices, or
         conflicts with the meaning of the <SUSPECT_ANSWER>.

Constraints:
- Use ONLY the information provided in <QUESTION>, <CHOICES_WITH_IDS>, and <SUSPECT_ANSWER>.
- Do NOT answer the <QUESTION> itself. Focus ONLY on judging how well the <SUSPECT_ANSWER>
     maps to the provided choices.
- Do NOT rewrite or restate the <QUESTION> or <SUSPECT_ANSWER>.
- label must be one of "likely", "neutral", or "unlikely".
- The output must include an entry for every choice in <CHOICES_WITH_IDS>.

Output format:
Return STRICT JSON only with the following schema:
{{
    "scores": [
        {{
            "choice_id": string,
            "reason": string,
            "label": string
        }},
        ...
    ]
}}
```

AR-BENCH-SP

System prompt.

```
You are an expert thinking puzzle analyst. Your task is to judge how well a puzzle host's
     response maps to each of the predefined answer choices. The host gives truthful answers
      ("yes" or "no").
```

User prompt.

```
<QUESTION>
{question}
</QUESTION>

<CHOICES_WITH_IDS>
{choices_with_ids}
</CHOICES_WITH_IDS>

<HOST_ANSWER>
{user_answer}
</HOST_ANSWER>

Definition:
<CHOICES_WITH_IDS> is a list of dicts with "id" and "value" fields. Each dict corresponds to
      a multiple-choice answer option for the question.
Let choices[i] be the i-th element of <CHOICES_WITH_IDS>.

Task:
Judge how well the <HOST_ANSWER> maps to each of the choices in <CHOICES_WITH_IDS> for the
     question <QUESTION>.

What to generate:
For i = 0..len(choices)-1:
- choice_id: the id of the question choice being evaluated, i.e. choices[i]["id"]
```

```
- reason: a short one-sentence explanation of choices[i]["value"] is likely/neutral/unlikely
      given the <HOST_ANSWER>.
- label: one of "likely", "neutral", or "unlikely" according to the following definitions:
    -"likely": choices[i]["value"] aligns well with the <HOST_ANSWER> and fits it better
         than most other choices.
    -"neutral": choices[i]["value"] is neither clearly supported nor clearly contradicted by
          the <HOST_ANSWER>.
    -"unlikely": choices[i]["value"] fits the <HOST_ANSWER> worse than other choices, or
         conflicts with the meaning of the <HOST_ANSWER>.

Constraints:
- Use ONLY the information provided in <QUESTION>, <CHOICES_WITH_IDS>, and <HOST_ANSWER>.
- Do NOT answer the <QUESTION> itself. Focus ONLY on judging how well the <HOST_ANSWER> maps
      to the provided choices.
- Do NOT rewrite or restate the <QUESTION> or <HOST_ANSWER>.
- label must be one of "likely", "neutral", or "unlikely".
- The output must include an entry for every choice in <CHOICES_WITH_IDS>.

Output format:
Return STRICT JSON only with the following schema:
{{
    "scores": [
        {{
            "choice_id": string,
            "reason": string,
            "label": string
        }},
        ...
    ]
}}
```

iCRAFT-MD

System prompt.

```
You are an experienced physician interpreting a patient's response to a clinical question.
     Your task is to judge how well the patient's answer maps to each of the predefined
     answer choices.
```

User prompt.

```
<QUESTION>
{question}
</QUESTION>

<CHOICES_WITH_IDS>
{choices_with_ids}
</CHOICES_WITH_IDS>

<PATIENT_ANSWER>
{user_answer}
</PATIENT_ANSWER>

Definition:
<CHOICES_WITH_IDS> is a list of dicts with "id" and "value" fields. Each dict corresponds to
      a multiple-choice answer option for the question.
Let choices[i] be the i-th element of <CHOICES_WITH_IDS>.

Task:
Judge how well the <PATIENT_ANSWER> maps to each of the choices in <CHOICES_WITH_IDS> for
     the question <QUESTION>.

What to generate:
For i = 0..len(choices)-1:
- choice_id: the id of the question choice being evaluated, i.e. choices[i]["id"]
- reason: a short one-sentence explanation of choices[i]["value"] is likely/neutral/unlikely
      given the <PATIENT_ANSWER>.
- label: one of "likely", "neutral", or "unlikely" according to the following definitions:
    -"likely": choices[i]["value"] aligns well with the <PATIENT_ANSWER> and fits it better
         than most other choices.
    -"neutral": choices[i]["value"] is neither clearly supported nor clearly contradicted by
          the <PATIENT_ANSWER>.
    -"unlikely": choices[i]["value"] fits the <PATIENT_ANSWER> worse than other choices, or
         conflicts with the meaning of the <PATIENT_ANSWER>.

Constraints:
- Use ONLY the information provided in <QUESTION>, <CHOICES_WITH_IDS>, and <PATIENT_ANSWER>.
```

```
- Do NOT answer the <QUESTION> itself. Focus ONLY on judging how well the <PATIENT_ANSWER>
    maps to the provided choices.
- Do NOT rewrite or restate the <QUESTION> or <PATIENT_ANSWER>.
- label must be one of "likely", "neutral", or "unlikely".
- The output must include an entry for every choice in <CHOICES_WITH_IDS>.

Output format:
Return STRICT JSON only with the following schema:
{{
    "scores": [
        {{
            "choice_id": string,
            "reason": string,
            "label": string
        }},
        ...
    ]
}}
```

V.7 EXPAND — NEW DIMENSION PROPOSAL

AR-BENCH-DC

System prompt.

```
You are an experienced detective who needs to explore a new line of investigation in a
    murder case. The current investigative dimensions have not been sufficient to identify
    the murderer, so you must identify a new aspect of the case to investigate.
```

User prompt.

```
<CASE_QUESTION>
{ambiguous_prompt}
</CASE_QUESTION>

<CASE_BACKGROUND>
{meta_context}
</CASE_BACKGROUND>

<PAST_INVESTIGATIVE_DIMENSIONS>
{past_dimensions}
</PAST_INVESTIGATIVE_DIMENSIONS>

<INTERROGATION_LOG>
{conversation_log}
</INTERROGATION_LOG>

Definition:
An investigative dimension is a specific aspect of the murder case where multiple suspects
    could plausibly be implicated, and resolving it would narrow down the true murderer.
<PAST_INVESTIGATIVE_DIMENSIONS> is a list of dicts with "name" fields, representing the
    investigative dimensions that have already been explored in the investigation.
<INTERROGATION_LOG> is a list of dicts with "question_text", "user_name", and "user_answer"
    fields, representing the history of the interrogation between the detective and the
    suspects up to this point.

Task:
Identify a new investigative dimension in the murder case described by <CASE_QUESTION> that
    has not been previously identified in <PAST_INVESTIGATIVE_DIMENSIONS>. Use insights
    from the <INTERROGATION_LOG> to guide your choice.

What to generate:
- reason: a short one-sentence explanation of why this dimension is a critical new line of
    investigation.
- name: a short, specific label for this investigative dimension (e.g., "Forensic evidence",
     "Financial motive", "Witness credibility", etc.).
- values: a list of plausible values, no larger than {max_num_values_per_dim}, that this
    dimension could take in the context of the case.

Constraints:
- Use ONLY the information provided in <CASE_QUESTION>, <CASE_BACKGROUND>, <
    PAST_INVESTIGATIVE_DIMENSIONS>, and <INTERROGATION_LOG>.
- Do NOT answer the <CASE_QUESTION> itself. Focus ONLY on identifying a new investigative
    dimension.
- Do NOT rewrite or restate the <CASE_QUESTION>, <CASE_BACKGROUND>, <
    PAST_INVESTIGATIVE_DIMENSIONS>, or <INTERROGATION_LOG>.
```

```
- The generated dimension name must not be the same as any of the names in <
    PAST_INVESTIGATIVE_DIMENSIONS>.

Output format:
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "name": string,
    "values": [string, ...]
}}
```

AR-BENCH-SP

System prompt.

```
You are an expert thinking puzzle solver. The puzzle dimensions explored so far have not
    been sufficient to explain the puzzle, so you must identify a new hidden aspect of the
    scenario to investigate.
```

User prompt.

```
<PUZZLE>
{ambiguous_prompt}
</PUZZLE>

<PUZZLE_CONTEXT>
{meta_context}
</PUZZLE_CONTEXT>

<PAST_PUZZLE_DIMENSIONS>
{past_dimensions}
</PAST_PUZZLE_DIMENSIONS>

<CONVERSATION_LOG>
{conversation_log}
</CONVERSATION_LOG>

Definition:
A puzzle dimension is a hidden aspect of the scenario where knowing its true value would
    explain the puzzle.
<PAST_PUZZLE_DIMENSIONS> is a list of dicts with "name" fields, representing the puzzle
    dimensions that have already been explored.
<CONVERSATION_LOG> is a list of dicts with "question_text", "user_name", and "user_answer"
    fields, representing the history of the conversation between the solver and the host up
     to this point.

Task:
Identify a new puzzle dimension in <PUZZLE> that has not been previously identified in <
    PAST_PUZZLE_DIMENSIONS>. Use insights from the <CONVERSATION_LOG> to guide your choice.

What to generate:
- reason: a short one-sentence explanation of why this dimension is a key unknown in the
    puzzle.
- name: a short, specific label for this puzzle dimension.
- values: a list of plausible interpretations, no larger than {max_num_values_per_dim}, that
     this dimension could take.

Constraints:
- Use ONLY the information provided in <PUZZLE>, <PUZZLE_CONTEXT>, <PAST_PUZZLE_DIMENSIONS>,
     and <CONVERSATION_LOG>.
- Do NOT solve the <PUZZLE> itself. Focus ONLY on identifying a new puzzle dimension.
- Do NOT rewrite or restate the <PUZZLE>, <PUZZLE_CONTEXT>, <PAST_PUZZLE_DIMENSIONS>, or <
    CONVERSATION_LOG>.
- The generated dimension name must not be the same as any of the names in <
    PAST_PUZZLE_DIMENSIONS>.

Output format:
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "name": string,
    "values": [string, ...]
}}
```

iCRAFT-MD

System prompt.

```
You are an experienced physician who needs to explore a new line of clinical inquiry. The
     clinical dimensions investigated so far have not been sufficient to arrive at a
     definitive diagnosis, so you must identify a new clinical factor to assess.
```

User prompt.

```
<CLINICAL_QUESTION>
{ambiguous_prompt}
</CLINICAL_QUESTION>

<PATIENT_INFORMATION>
{meta_context}
</PATIENT_INFORMATION>

<PAST_CLINICAL_DIMENSIONS>
{past_dimensions}
</PAST_CLINICAL_DIMENSIONS>

<CLINICAL_INTERVIEW_LOG>
{conversation_log}
</CLINICAL_INTERVIEW_LOG>

Definition:
A clinical dimension is a specific clinical factor where different values would point toward
      different diagnoses or clinical decisions.
<PAST_CLINICAL_DIMENSIONS> is a list of dicts with "name" fields, representing the clinical
     dimensions that have already been assessed in the patient interview.
<CLINICAL_INTERVIEW_LOG> is a list of dicts with "question_text", "user_name", and "
     user_answer" fields, representing the history of the clinical interview between the
     physician and the patient up to this point.

Task:
Identify a new clinical dimension relevant to the <CLINICAL_QUESTION> that has not been
     previously identified in <PAST_CLINICAL_DIMENSIONS>. Use insights from the <
     CLINICAL_INTERVIEW_LOG> to guide your choice---the patient's answers may reveal the
     need to investigate additional clinical factors.

What to generate:
- reason: a short one-sentence explanation of why this clinical factor is important for
     narrowing the diagnosis.
- name: a short, specific clinical label for this dimension.
- values: a list of clinically plausible values, no larger than {max_num_values_per_dim},
     that this dimension could take.

Constraints:
- Use ONLY the information provided in <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, <
     PAST_CLINICAL_DIMENSIONS>, and <CLINICAL_INTERVIEW_LOG>.
- Do NOT answer the <CLINICAL_QUESTION> itself. Focus ONLY on identifying a new clinical
     dimension.
- Do NOT rewrite or restate the <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, <
     PAST_CLINICAL_DIMENSIONS>, or <CLINICAL_INTERVIEW_LOG>.
- The generated dimension name must not be the same as any of the names in <
     PAST_CLINICAL_DIMENSIONS>.

Output format:
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "name": string,
    "values": [string, ...]
}}
```

V.8 EXPAND — PRIOR ELICITATION

AR-BENCH-DC

System prompt.

```
You are an experienced detective forming a hypothesis about a newly identified aspect of a
     murder case, taking into account both the case background and what has been revealed
     during the interrogation so far.
```

User prompt.

```
<CASE_QUESTION>
{ambiguous_prompt}
</CASE_QUESTION>

<CASE_BACKGROUND>
{meta_context}
</CASE_BACKGROUND>

<INTERROGATION_LOG>
{conversation_log}
</INTERROGATION_LOG>

<DIMENSION_NAME>
{dimension_name}
</DIMENSION_NAME>

<DIMENSION_VALUE>
{dimension_value}
</DIMENSION_VALUE>

Definition:
An investigative dimension is a specific aspect of the murder case where multiple suspects
     could plausibly be implicated, and resolving it would narrow down the true murderer.
<INTERROGATION_LOG> is a list of dicts with "question_text", "user_name", and "user_answer"
     fields, representing the history of the interrogation between the detective and the
     suspects up to this point.

Task:
Given <CASE_QUESTION>, <CASE_BACKGROUND>, <INTERROGATION_LOG>, and a specific investigative
     dimension defined by <DIMENSION_NAME> and <DIMENSION_VALUE>, judge how likely it is
     that the <DIMENSION_NAME> takes on the value <DIMENSION_VALUE>.

What to generate:
- reason: a short one-sentence explanation of why the <DIMENSION_NAME> is likely, unlikely,
     or neutral to take on the value <DIMENSION_VALUE>.
- label: one of "likely", "unlikely", or "neutral" according to the following definitions:
    - likely: <DIMENSION_VALUE> is explicitly stated, strongly implied, or is the most
         natural assumption given the evidence in the <CASE_QUESTION>, <CASE_BACKGROUND>,
         and <INTERROGATION_LOG>.
    - neutral: <DIMENSION_VALUE> is plausible but not implied or supported by specific
         evidence in the <CASE_QUESTION>, <CASE_BACKGROUND>, or <INTERROGATION_LOG>.
    - unlikely: <DIMENSION_VALUE> is contradicted by the <CASE_QUESTION>, <CASE_BACKGROUND>
         or <INTERROGATION_LOG>, or would require assumptions that are inconsistent with the
          available evidence.

Constraints:
- Use ONLY the information provided in <CASE_QUESTION>, <CASE_BACKGROUND>, and <
     INTERROGATION_LOG>.
- Do NOT answer the <CASE_QUESTION> itself. Focus ONLY on judging the likelihood of the
     dimension value.
- Do NOT rewrite or restate the <CASE_QUESTION>, <CASE_BACKGROUND>, or <INTERROGATION_LOG>.
- label must be one of "likely", "unlikely", or "neutral".

Output format:
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "label": string
}}
```

AR-BENCH-SP

System prompt.

```
You are an expert thinking puzzle solver forming a hypothesis about a newly identified
     hidden aspect of the puzzle, taking into account both the puzzle scenario and what the
     host has revealed so far.
```

User prompt.

```
<PUZZLE>
{ambiguous_prompt}
</PUZZLE>

<PUZZLE_CONTEXT>
{meta_context}
```

```
</PUZZLE_CONTEXT>

<CONVERSATION_LOG>
{conversation_log}
</CONVERSATION_LOG>

<DIMENSION_NAME>
{dimension_name}
</DIMENSION_NAME>

<DIMENSION_VALUE>
{dimension_value}
</DIMENSION_VALUE>

Definition:
A puzzle dimension is a hidden aspect of the scenario where knowing its true value would
     explain the puzzle.
<CONVERSATION_LOG> is a list of dicts with "question_text", "user_name", and "user_answer"
     fields, representing the history of the conversation between the solver and the host up
      to this point.

Task:
Given <PUZZLE>, <PUZZLE_CONTEXT>, <CONVERSATION_LOG>, and a specific puzzle dimension
     defined by <DIMENSION_NAME> and <DIMENSION_VALUE>, judge how likely it is that the <
     DIMENSION_NAME> takes on the value <DIMENSION_VALUE>.

What to generate:
- reason: a short one-sentence explanation of why the <DIMENSION_NAME> is likely, unlikely,
     or neutral to take on the value <DIMENSION_VALUE>.
- label: one of "likely", "unlikely", or "neutral" according to the following definitions:
    - likely: <DIMENSION_VALUE> is explicitly suggested by, strongly implied by, or is the
         most natural interpretation given the clues in the <PUZZLE>, <PUZZLE_CONTEXT>, and
         <CONVERSATION_LOG>.
    - neutral: <DIMENSION_VALUE> is plausible but not implied or supported by specific clues
          in the <PUZZLE>, <PUZZLE_CONTEXT>, or <CONVERSATION_LOG>.
    - unlikely: <DIMENSION_VALUE> is contradicted by the <PUZZLE>, <PUZZLE_CONTEXT> or <
         CONVERSATION_LOG>, or would require assumptions that are inconsistent with the
         scenario.

Constraints:
- Use ONLY the information provided in <PUZZLE>, <PUZZLE_CONTEXT>, and <CONVERSATION_LOG>.
- Do NOT solve the <PUZZLE> itself. Focus ONLY on judging the likelihood of the dimension
     value.
- Do NOT rewrite or restate the <PUZZLE>, <PUZZLE_CONTEXT>, or <CONVERSATION_LOG>.
- label must be one of "likely", "unlikely", or "neutral".

Output format:
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "label": string
}}
```

ICRAFT-MD

System prompt.

```
You are an experienced physician forming a clinical hypothesis about a newly identified
     clinical factor, taking into account the patient's baseline information and what has
     been revealed during the clinical interview so far.
```

User prompt.

```
<CLINICAL_QUESTION>
{ambiguous_prompt}
</CLINICAL_QUESTION>

<PATIENT_INFORMATION>
{meta_context}
</PATIENT_INFORMATION>

<CLINICAL_INTERVIEW_LOG>
{conversation_log}
</CLINICAL_INTERVIEW_LOG>

<DIMENSION_NAME>
{dimension_name}
```

```
</DIMENSION_NAME>

<DIMENSION_VALUE>
{dimension_value}
</DIMENSION_VALUE>

Definition:
A clinical dimension is a specific clinical factor where different values would point toward
      different diagnoses or clinical decisions.
<CLINICAL_INTERVIEW_LOG> is a list of dicts with "question_text", "user_name", and "
     user_answer" fields, representing the history of the clinical interview between the
     physician and the patient up to this point.

Task:
Given <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, <CLINICAL_INTERVIEW_LOG>, and a specific
     clinical dimension defined by <DIMENSION_NAME> and <DIMENSION_VALUE>, judge how likely
     it is that the <DIMENSION_NAME> takes on the value <DIMENSION_VALUE>.

What to generate:
- reason: a short one-sentence explanation of why the <DIMENSION_NAME> is likely, unlikely,
     or neutral to take on the value <DIMENSION_VALUE>.
- label: one of "likely", "unlikely", or "neutral" according to the following definitions:
    - likely: <DIMENSION_VALUE> is explicitly stated, strongly implied, or is the most
         natural clinical assumption given the patient information in <CLINICAL_QUESTION>, <
         PATIENT_INFORMATION>, and <CLINICAL_INTERVIEW_LOG>.
    - neutral: <DIMENSION_VALUE> is clinically plausible but not implied or supported by
         specific evidence in the <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, or <
         CLINICAL_INTERVIEW_LOG>.
    - unlikely: <DIMENSION_VALUE> is contradicted by the <CLINICAL_QUESTION>, <
         PATIENT_INFORMATION> or <CLINICAL_INTERVIEW_LOG>, or would require assumptions that
          are inconsistent with the patient's presentation.

Constraints:
- Use ONLY the information provided in <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, and <
     CLINICAL_INTERVIEW_LOG>.
- Do NOT answer the <CLINICAL_QUESTION> itself. Focus ONLY on judging the likelihood of the
     dimension value.
- Do NOT rewrite or restate the <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, or <
     CLINICAL_INTERVIEW_LOG>.
- label must be one of "likely", "unlikely", or "neutral".

Output format:
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "label": string
}}
```

V.9 EXPAND — QUESTION GENERATION

AR-BENCH-DC

System prompt.

```
You are an experienced detective preparing new interrogation questions based on a newly
     discovered line of investigation and unresolved aspects of the murder case.
```

User prompt.

```
<CASE_QUESTION>
{ambiguous_prompt}
</CASE_QUESTION>

<CASE_BACKGROUND>
{meta_context}
</CASE_BACKGROUND>

<INTERROGATION_LOG>
{conversation_log}
</INTERROGATION_LOG>

<NEW_INVESTIGATIVE_DIMENSION>
{new_dimension_with_values}
</NEW_INVESTIGATIVE_DIMENSION>

<UNRESOLVED_INVESTIGATIVE_DIMENSIONS>
{high_uncertainty_dimensions_with_values}
```

```
</UNRESOLVED_INVESTIGATIVE_DIMENSIONS>

Definition:
An investigative dimension is a specific aspect of the murder case where multiple suspects
     could plausibly be implicated, and resolving it would narrow down the true murderer.
<INTERROGATION_LOG> is a list of dicts with "question_text", "user_name", and "user_answer"
     fields, representing the history of the interrogation between the detective and the
     suspects up to this point.
<NEW_INVESTIGATIVE_DIMENSION> is a dict with "name" and "values" fields, representing the
     newly identified line of investigation along with its possible values.
<UNRESOLVED_INVESTIGATIVE_DIMENSIONS> is a list of dicts with "name" and "values" fields,
     representing the investigative dimensions that currently have the highest uncertainty.
     They do not include the new dimension in <NEW_INVESTIGATIVE_DIMENSION>.

Task:
Given <CASE_QUESTION>, <CASE_BACKGROUND>, <INTERROGATION_LOG>, a newly identified
     investigative dimension in <NEW_INVESTIGATIVE_DIMENSION>, and the most uncertain
     dimensions in <UNRESOLVED_INVESTIGATIVE_DIMENSIONS>, generate interrogation questions
     that would help identify the murderer by targeting the new dimension and/or the
     unresolved dimensions.

What to generate:
Generate at most {max_new_questions_per_round} interrogation questions. For each question,
     provide:
- reason: a short one-sentence explanation of why this question would help identify the
     murderer.
- question: the text of the interrogation question.
- choices: a list of multiple-choice answer options for the question, no larger than {
     max_choices_per_question}.

Constraints:
- Use ONLY the information provided in <CASE_QUESTION>, <CASE_BACKGROUND>, <
     INTERROGATION_LOG>, <NEW_INVESTIGATIVE_DIMENSION>, and <
     UNRESOLVED_INVESTIGATIVE_DIMENSIONS>.
- Do NOT answer the <CASE_QUESTION> itself. Focus ONLY on generating interrogation questions
     .
- Do NOT rewrite or restate the <CASE_QUESTION>, <CASE_BACKGROUND>, <INTERROGATION_LOG>, <
     NEW_INVESTIGATIVE_DIMENSION>, or <UNRESOLVED_INVESTIGATIVE_DIMENSIONS>.
- Each question must be designed to elicit information about the new dimension in <
     NEW_INVESTIGATIVE_DIMENSION> and/or the unresolved dimensions in <
     UNRESOLVED_INVESTIGATIVE_DIMENSIONS>.
- Each question must have multiple-choice answers.
- Generate at most {max_new_questions_per_round} questions.

Output format:
Return STRICT JSON only with the following schema:
{{
    "questions": [
        {{
            "reason": string,
            "question": string,
            "choices": [string, ...]
        }},
        ...
    ]
}}
```

AR-BENCH-SP

System prompt.

```
You are an expert thinking puzzle solver generating new questions to ask the puzzle host.
     Focus on testing specific hypotheses about the hidden explanation, especially targeting
      newly discovered or still-uncertain aspects of the puzzle.
```

User prompt.

```
<PUZZLE>
{ambiguous_prompt}
</PUZZLE>

<PUZZLE_CONTEXT>
{meta_context}
</PUZZLE_CONTEXT>

<CONVERSATION_LOG>
{conversation_log}
```

```
</CONVERSATION_LOG>

<NEW_PUZZLE_DIMENSION>
{new_dimension_with_values}
</NEW_PUZZLE_DIMENSION>

<UNCERTAIN_PUZZLE_DIMENSIONS>
{high_uncertainty_dimensions_with_values}
</UNCERTAIN_PUZZLE_DIMENSIONS>

Definition:
A puzzle dimension is a hidden aspect of the scenario where knowing its true value would
     explain the puzzle.
<CONVERSATION_LOG> is a list of dicts with "question_text", "user_name", and "user_answer"
     fields, representing the history of the conversation between the solver and the host up
      to this point.
<NEW_PUZZLE_DIMENSION> is a dict with "name" and "values" fields, representing the newly
     identified hidden aspect of the puzzle along with its possible interpretations.
<UNCERTAIN_PUZZLE_DIMENSIONS> is a list of dicts with "name" and "values" fields,
     representing the puzzle dimensions that currently have the highest uncertainty. They do
      not include the new dimension in <NEW_PUZZLE_DIMENSION>.

Task:
Given <PUZZLE>, <PUZZLE_CONTEXT>, <CONVERSATION_LOG>, a newly identified puzzle dimension in
      <NEW_PUZZLE_DIMENSION>, and the most uncertain dimensions in <
     UNCERTAIN_PUZZLE_DIMENSIONS>, generate questions to ask the puzzle host that would help
      uncover the hidden explanation by targeting the new dimension and/or the uncertain
     dimensions.

What to generate:
Generate at most {max_new_questions_per_round} clarifying questions. For each question,
     provide:
- reason: a short one-sentence explanation of why this question would help solve the puzzle.
- question: the text of the clarifying question to ask the puzzle host.
- choices: ["yes", "no"] as the {max_choices_per_question} multiple-choice answer options
     for the question.

Constraints:
- Use ONLY the information provided in <PUZZLE>, <PUZZLE_CONTEXT>, <CONVERSATION_LOG>, <
     NEW_PUZZLE_DIMENSION>, and <UNCERTAIN_PUZZLE_DIMENSIONS>.
- Do NOT solve the <PUZZLE> itself. Focus ONLY on generating clarifying questions.
- Do NOT rewrite or restate the <PUZZLE>, <PUZZLE_CONTEXT>, <CONVERSATION_LOG>, <
     NEW_PUZZLE_DIMENSION>, or <UNCERTAIN_PUZZLE_DIMENSIONS>.
- Each question must be designed to elicit information about the new dimension in <
     NEW_PUZZLE_DIMENSION> and/or the uncertain dimensions in <UNCERTAIN_PUZZLE_DIMENSIONS>.
- Each question must have multiple-choice answers.
- Generate at most {max_new_questions_per_round} questions.
- Keep each question short: at most 20 words.
- Keep each reason short: at most 15 words.

Output format:
Return STRICT JSON only with the following schema:
{{
    "questions": [
        {{
            "reason": string,
            "question": string,
            "choices": ["yes", "no"]
        }},
        ...
    ]
}}
```

ICRAFT-MD

System prompt.

```
You are an experienced physician preparing additional clinical questions for a patient
     interview based on a newly identified clinical factor and unresolved aspects of the
     diagnosis.
```

User prompt.

```
<CLINICAL_QUESTION>
{ambiguous_prompt}
</CLINICAL_QUESTION>
```

```
<PATIENT_INFORMATION>
{meta_context}
</PATIENT_INFORMATION>

<CLINICAL_INTERVIEW_LOG>
{conversation_log}
</CLINICAL_INTERVIEW_LOG>

<NEW_CLINICAL_DIMENSION>
{new_dimension_with_values}
</NEW_CLINICAL_DIMENSION>

<UNRESOLVED_CLINICAL_DIMENSIONS>
{high_uncertainty_dimensions_with_values}
</UNRESOLVED_CLINICAL_DIMENSIONS>

Definition:
A clinical dimension is a specific clinical factor where different values would point toward
      different diagnoses or clinical decisions.
<CLINICAL_INTERVIEW_LOG> is a list of dicts with "question_text", "user_name", and "
     user_answer" fields, representing the history of the clinical interview between the
     physician and the patient up to this point.
<NEW_CLINICAL_DIMENSION> is a dict with "name" and "values" fields, representing the newly
     identified clinical factor along with its possible values.
<UNRESOLVED_CLINICAL_DIMENSIONS> is a list of dicts with "name" and "values" fields,
     representing the clinical dimensions that currently have the highest diagnostic
     uncertainty. They do not include the new dimension in <NEW_CLINICAL_DIMENSION>.

Task:
Given <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, <CLINICAL_INTERVIEW_LOG>, a newly
     identified clinical dimension in <NEW_CLINICAL_DIMENSION>, and the most uncertain
     dimensions in <UNRESOLVED_CLINICAL_DIMENSIONS>, generate clinical questions to ask the
     patient that would help narrow the diagnosis by targeting the new dimension and/or the
     unresolved dimensions.

What to generate:
Generate at most {max_new_questions_per_round} clinical questions. For each question,
     provide:
- reason: a short one-sentence explanation of why this question would help narrow the
     diagnosis.
- question: the text of the clinical question to ask the patient.
- choices: a list of multiple-choice answer options for the question, no larger than {
     max_choices_per_question}.

Constraints:
- Use ONLY the information provided in <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, <
     CLINICAL_INTERVIEW_LOG>, <NEW_CLINICAL_DIMENSION>, and <UNRESOLVED_CLINICAL_DIMENSIONS
     >.
- Do NOT answer the <CLINICAL_QUESTION> itself. Focus ONLY on generating clinical questions.
- Do NOT rewrite or restate the <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, <
     CLINICAL_INTERVIEW_LOG>, <NEW_CLINICAL_DIMENSION>, or <UNRESOLVED_CLINICAL_DIMENSIONS>.
- Each question must be designed to elicit information about the new dimension in <
     NEW_CLINICAL_DIMENSION> and/or the unresolved dimensions in <
     UNRESOLVED_CLINICAL_DIMENSIONS>.
- Each question must have multiple-choice answers.
- Generate at most {max_new_questions_per_round} questions.

Output format:
Return STRICT JSON only with the following schema:
{{
    "questions": [
        {{
            "reason": string,
            "question": string,
            "choices": [string, ...]
        }},
        ...
    ]
}}
```

V.10 FINAL ANSWER

AR-BENCH-DC

System prompt.

```
You are an experienced detective concluding a murder investigation. Based on all the
     evidence gathered from interrogating the suspects and your analysis of the case, you
     must now identify the real murderer.
```

Without discrete choices.

```
<CASE_QUESTION>
{ambiguous_prompt}
</CASE_QUESTION>

<CASE_BACKGROUND>
{meta_context}
</CASE_BACKGROUND>

<INTERROGATION_LOG>
{conversation_log}
</INTERROGATION_LOG>

<INVESTIGATION_CONCLUSION>
{map_state}
</INVESTIGATION_CONCLUSION>

Definition:
<INTERROGATION_LOG> is a list of dicts with "question_text", "user_name", and "user_answer"
     fields, representing the full history of the interrogation between the detective and
     the suspects.
An investigative dimension is a specific aspect of the murder case where multiple suspects
     could plausibly be implicated, and resolving it would narrow down the true murderer.
<INVESTIGATION_CONCLUSION> is a structured representation of the detective's current
     understanding of the case, where each investigative dimension is mapped to its most
     likely value. This represents the detective's best assessment of the true state of the
     case based on the investigation so far.

Task:
Given <CASE_QUESTION>, <CASE_BACKGROUND>, <INTERROGATION_LOG>, and <INVESTIGATION_CONCLUSION
     >, identify the real murderer.

What to generate:
- reason: a short one-sentence explanation of why the identified suspect is the real
     murderer given the evidence in <CASE_QUESTION>, <CASE_BACKGROUND>, <INTERROGATION_LOG>,
      and <INVESTIGATION_CONCLUSION>.
- final_answer: the name of the suspect you are identifying as the real murderer.

Constraints:
- The final answer must be consistent with the evidence in <CASE_QUESTION>, <CASE_BACKGROUND
     >, <INTERROGATION_LOG>, and <INVESTIGATION_CONCLUSION>.
- Do NOT rewrite or restate the <CASE_QUESTION>, <CASE_BACKGROUND>, <INTERROGATION_LOG>, or
     <INVESTIGATION_CONCLUSION>.
- answer must be a natural language answer to the <CASE_QUESTION>.

Output format:
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "final_answer": string
}}
```

With discrete choices.

```
<CASE_QUESTION>
{ambiguous_prompt}
</CASE_QUESTION>

<CASE_BACKGROUND>
{meta_context}
</CASE_BACKGROUND>

<INTERROGATION_LOG>
{conversation_log}
</INTERROGATION_LOG>

<INVESTIGATION_CONCLUSION>
{map_state}
</INVESTIGATION_CONCLUSION>

<SUSPECTS_WITH_IDS>
{possible_answers_with_ids}
</SUSPECTS_WITH_IDS>
```

```
Definition:
<INTERROGATION_LOG> is a list of dicts with "question_text", "user_name", and "user_answer"
     fields, representing the full history of the interrogation between the detective and
     the suspects.
An investigative dimension is a specific aspect of the murder case where multiple suspects
     could plausibly be implicated, and resolving it would narrow down the true murderer.
<INVESTIGATION_CONCLUSION> is a structured representation of the detective's current
     understanding of the case, where each investigative dimension is mapped to its most
     likely value. This represents the detective's best assessment of the true state of the
     case based on the investigation so far.
<SUSPECTS_WITH_IDS> is a list of dicts with "id" and "value" fields. Each dict corresponds
     to a suspect who could be the murderer.
Let possible_answers[i] be the i-th element of <SUSPECTS_WITH_IDS>.

Task:
Given <CASE_QUESTION>, <CASE_BACKGROUND>, <INTERROGATION_LOG>, and <INVESTIGATION_CONCLUSION
     >, identify which suspect from <SUSPECTS_WITH_IDS> is the real murderer.

What to generate:
- reason: a short one-sentence explanation of why the identified suspect is the real
     murderer given the evidence in <CASE_QUESTION>, <CASE_BACKGROUND>, <INTERROGATION_LOG>,
      and <INVESTIGATION_CONCLUSION>.
- final_answer_id: the id of the suspect in <SUSPECTS_WITH_IDS> that you are identifying as
     the real murderer.

Constraints:
- The final answer must be consistent with the evidence in <CASE_QUESTION>, <CASE_BACKGROUND
     >, <INTERROGATION_LOG>, and <INVESTIGATION_CONCLUSION>.
- Do NOT rewrite or restate the <CASE_QUESTION>, <CASE_BACKGROUND>, <INTERROGATION_LOG>, or
     <INVESTIGATION_CONCLUSION>.
- final_answer_id must be one of the ids (i.e possible_answers[i]["id"]) provided in <
     SUSPECTS_WITH_IDS>.

Output format:
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "final_answer_id": string
}}
```

AR-BENCH-SP

System prompt.

```
You are an expert thinking puzzle solver. Based on all the clues gathered from the puzzle
     host's responses and your analysis of the puzzle dimensions, you must now provide the
     hidden explanation of the puzzle.
```

Without discrete choices.

```
<PUZZLE>
{ambiguous_prompt}
</PUZZLE>

<PUZZLE_CONTEXT>
{meta_context}
</PUZZLE_CONTEXT>

<CONVERSATION_LOG>
{conversation_log}
</CONVERSATION_LOG>

<PUZZLE_SOLUTION_STATE>
{map_state}
</PUZZLE_SOLUTION_STATE>

Definition:
<CONVERSATION_LOG> is a list of dicts with "question_text", "user_name", and "user_answer"
     fields, representing the full history of the conversation between the solver and the
     puzzle host.
A puzzle dimension is a hidden aspect of the scenario where knowing its true value would
     explain the puzzle.
<PUZZLE_SOLUTION_STATE> is a structured representation of the solver's current understanding
      of the puzzle, where each puzzle dimension is mapped to its most likely value. This
     represents the solver's best guess of the hidden explanation based on the clues
     gathered so far.
```

```
Task:
Given <PUZZLE>, <PUZZLE_CONTEXT>, <CONVERSATION_LOG>, and <PUZZLE_SOLUTION_STATE>, provide
     the hidden explanation of the puzzle.

What to generate:
- reason: a short one-sentence explanation of why this is the correct explanation given the
     clues in <PUZZLE>, <PUZZLE_CONTEXT>, <CONVERSATION_LOG>, and <PUZZLE_SOLUTION_STATE>.
- final_answer: the hidden explanation of the puzzle.

Constraints:
- The final answer must be consistent with the clues in <PUZZLE>, <PUZZLE_CONTEXT>, <
     CONVERSATION_LOG>, and <PUZZLE_SOLUTION_STATE>.
- Do NOT rewrite or restate the <PUZZLE>, <PUZZLE_CONTEXT>, <CONVERSATION_LOG>, or <
     PUZZLE_SOLUTION_STATE>.
- answer must be a natural language explanation of the puzzle.

Output format:
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "final_answer": string
}}
```

With discrete choices.

```
<PUZZLE>
{ambiguous_prompt}
</PUZZLE>

<PUZZLE_CONTEXT>
{meta_context}
</PUZZLE_CONTEXT>

<CONVERSATION_LOG>
{conversation_log}
</CONVERSATION_LOG>

<PUZZLE_SOLUTION_STATE>
{map_state}
</PUZZLE_SOLUTION_STATE>

<POSSIBLE_EXPLANATIONS_WITH_IDS>
{possible_answers_with_ids}
</POSSIBLE_EXPLANATIONS_WITH_IDS>

Definition:
<CONVERSATION_LOG> is a list of dicts with "question_text", "user_name", and "user_answer"
     fields, representing the full history of the conversation between the solver and the
     puzzle host.
A puzzle dimension is a hidden aspect of the scenario where knowing its true value would
     explain the puzzle.
<PUZZLE_SOLUTION_STATE> is a structured representation of the solver's current understanding
      of the puzzle, where each puzzle dimension is mapped to its most likely value. This
     represents the solver's best guess of the hidden explanation based on the clues
     gathered so far.
<POSSIBLE_EXPLANATIONS_WITH_IDS> is a list of dicts with "id" and "value" fields. Each dict
     corresponds to a possible explanation of the puzzle.
Let possible_answers[i] be the i-th element of <POSSIBLE_EXPLANATIONS_WITH_IDS>.

Task:
Given <PUZZLE>, <PUZZLE_CONTEXT>, <CONVERSATION_LOG>, and <PUZZLE_SOLUTION_STATE>, select
     the correct explanation from <POSSIBLE_EXPLANATIONS_WITH_IDS>.

What to generate:
- reason: a short one-sentence explanation of why this is the correct explanation given the
     clues in <PUZZLE>, <PUZZLE_CONTEXT>, <CONVERSATION_LOG>, and <PUZZLE_SOLUTION_STATE>.
- final_answer_id: the id of the explanation in <POSSIBLE_EXPLANATIONS_WITH_IDS> that you
     are selecting as the hidden explanation of the puzzle.

Constraints:
- The final answer must be consistent with the clues in <PUZZLE>, <PUZZLE_CONTEXT>, <
     CONVERSATION_LOG>, and <PUZZLE_SOLUTION_STATE>.
- Do NOT rewrite or restate the <PUZZLE>, <PUZZLE_CONTEXT>, <CONVERSATION_LOG>, or <
     PUZZLE_SOLUTION_STATE>.
- final_answer_id must be one of the ids (i.e possible_answers[i]["id"]) provided in <
     POSSIBLE_EXPLANATIONS_WITH_IDS>.

Output format:
```

```
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "final_answer_id": string
}}
```

iCraft-MD

System prompt.

```
You are an experienced physician concluding a clinical assessment. Based on all the clinical
     information gathered from the patient interview and your analysis of the clinical
    dimensions, you must now provide your diagnosis or clinical decision.
```

Without discrete choices.

```
<CLINICAL_QUESTION>
{ambiguous_prompt}
</CLINICAL_QUESTION>

<PATIENT_INFORMATION>
{meta_context}
</PATIENT_INFORMATION>

<CLINICAL_INTERVIEW_LOG>
{conversation_log}
</CLINICAL_INTERVIEW_LOG>

<CLINICAL_ASSESSMENT>
{map_state}
</CLINICAL_ASSESSMENT>

Definition:
<CLINICAL_INTERVIEW_LOG> is a list of dicts with "question_text", "user_name", and "
    user_answer" fields, representing the full history of the clinical interview between
    the physician and the patient.
A clinical dimension is a specific clinical factor where different values would point toward
     different diagnoses or clinical decisions.
<CLINICAL_ASSESSMENT> is a structured representation of the physician's current
    understanding of the patient's condition, where each clinical dimension is mapped to
    its most likely value. This represents the physician's best assessment of the true
    clinical state based on the interview so far.

Task:
Given <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, <CLINICAL_INTERVIEW_LOG>, and <
    CLINICAL_ASSESSMENT>, provide your diagnosis or clinical decision.

What to generate:
- reason: a short one-sentence explanation of why this is the correct diagnosis or clinical
    decision given the clinical evidence in <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, <
    CLINICAL_INTERVIEW_LOG>, and <CLINICAL_ASSESSMENT>.
- final_answer: your diagnosis or clinical decision.

Constraints:
- The final answer must be consistent with the clinical evidence in <CLINICAL_QUESTION>, <
    PATIENT_INFORMATION>, <CLINICAL_INTERVIEW_LOG>, and <CLINICAL_ASSESSMENT>.
- Do NOT rewrite or restate the <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, <
    CLINICAL_INTERVIEW_LOG>, or <CLINICAL_ASSESSMENT>.
- answer must be a natural language answer to the <CLINICAL_QUESTION>.

Output format:
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "final_answer": string
}}
```

With discrete choices.

```
<CLINICAL_QUESTION>
{ambiguous_prompt}
</CLINICAL_QUESTION>

<PATIENT_INFORMATION>
{meta_context}
</PATIENT_INFORMATION>
```

```
<CLINICAL_INTERVIEW_LOG>
{conversation_log}
</CLINICAL_INTERVIEW_LOG>

<CLINICAL_ASSESSMENT>
{map_state}
</CLINICAL_ASSESSMENT>

<DIAGNOSTIC_OPTIONS_WITH_IDS>
{possible_answers_with_ids}
</DIAGNOSTIC_OPTIONS_WITH_IDS>

Definition:
<CLINICAL_INTERVIEW_LOG> is a list of dicts with "question_text", "user_name", and "
    user_answer" fields, representing the full history of the clinical interview between
    the physician and the patient.
A clinical dimension is a specific clinical factor where different values would point toward
     different diagnoses or clinical decisions.
<CLINICAL_ASSESSMENT> is a structured representation of the physician's current
    understanding of the patient's condition, where each clinical dimension is mapped to
    its most likely value. This represents the physician's best assessment of the true
    clinical state based on the interview so far.
<DIAGNOSTIC_OPTIONS_WITH_IDS> is a list of dicts with "id" and "value" fields. Each dict
    corresponds to a possible diagnosis or clinical decision.
Let possible_answers[i] be the i-th element of <DIAGNOSTIC_OPTIONS_WITH_IDS>.

Task:
Given <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, <CLINICAL_INTERVIEW_LOG>, and <
    CLINICAL_ASSESSMENT>, select the correct diagnosis or clinical decision from <
    DIAGNOSTIC_OPTIONS_WITH_IDS>.

What to generate:
- reason: a short one-sentence explanation of why this is the correct diagnosis or clinical
    decision given the clinical evidence in <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, <
    CLINICAL_INTERVIEW_LOG>, and <CLINICAL_ASSESSMENT>.
- final_answer_id: the id of the choice in <DIAGNOSTIC_OPTIONS_WITH_IDS> that you are
    selecting as your diagnosis or clinical decision.

Constraints:
- The final answer must be consistent with the clinical evidence in <CLINICAL_QUESTION>, <
    PATIENT_INFORMATION>, <CLINICAL_INTERVIEW_LOG>, and <CLINICAL_ASSESSMENT>.
- Do NOT rewrite or restate the <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, <
    CLINICAL_INTERVIEW_LOG>, or <CLINICAL_ASSESSMENT>.
- final_answer_id must be one of the ids (i.e possible_answers[i]["id"]) provided in <
    DIAGNOSTIC_OPTIONS_WITH_IDS>.

Output format:
Return STRICT JSON only with the following schema:
{{
    "reason": string,
    "final_answer_id": string
}}
```

V.11 ANSWER LIKELIHOOD

AR-BENCH-DC

System prompt.

```
You are an experienced detective evaluating how likely each suspect is to be the real
    murderer under different assumptions about the state of the case.
```

User prompt.

```
<CASE_QUESTION>
{ambiguous_prompt}
</CASE_QUESTION>

<CASE_BACKGROUND>
{meta_context}
</CASE_BACKGROUND>

<DIMENSION_NAME>
{dimension_name}
</DIMENSION_NAME>
```

```
<DIMENSION_VALUES_WITH_IDS>
{dimension_values_with_ids}
</DIMENSION_VALUES_WITH_IDS>

<SUSPECTS_WITH_IDS>
{possible_answers_with_ids}
</SUSPECTS_WITH_IDS>

Definition:
An investigative dimension is a specific aspect of the murder case where multiple suspects
     could plausibly be implicated, and resolving it would narrow down the true murderer.
<DIMENSION_VALUES_WITH_IDS> is a list of dicts with "id" and "text" fields. Each dict
     corresponds to a possible value that the <DIMENSION_NAME> could take.
<SUSPECTS_WITH_IDS> is a list of dicts with "id" and "text" fields. Each dict corresponds to
      a suspect who could be the real murderer.
Let values[i] be the i-th element of <DIMENSION_VALUES_WITH_IDS>.
Let answers[j] be the j-th element of <SUSPECTS_WITH_IDS>.

Task (row-major order):
For i = 0..len(values)-1:
For j = 0..len(answers)-1:
    - Assume the true state of the case is <DIMENSION_NAME> = values[i]["text"].
    - Judge how likely it is that answers[j]["text"] is the real murderer given that
        assumption and the evidence in <CASE_BACKGROUND>.

What to generate:
For i = 0..len(values)-1:
For j = 0..len(answers)-1:
    - answer_id: the id of the suspect being evaluated, i.e. answers[j]["id"]
    - dimension_value_id: the id of the dimension value being evaluated, i.e. values[i]["id
        "]
    - reason: a short one-sentence explanation of why answers[j] is likely/neutral/unlikely
        to be the real murderer given <DIMENSION_NAME> = values[i]["text"].
    - label: one of "likely", "neutral", or "unlikely" according to the following
        definitions:
        - "likely": Given <DIMENSION_NAME> = values[i]["text"] and the evidence in <
            CASE_BACKGROUND>, answers[j]["text"] is the expected real murderer.
        - "neutral": Given <DIMENSION_NAME> = values[i]["text"] and the evidence in <
            CASE_BACKGROUND>, answers[j]["text"] is a plausible suspect but not
            specifically implicated; there is insufficient evidence to say this suspect is
            more or less guilty than others.
        - "unlikely": Given <DIMENSION_NAME> = values[i]["text"] and the evidence in <
            CASE_BACKGROUND>, answers[j]["text"] is not expected to be the real murderer.

Constraints:
- Use ONLY the information provided in <CASE_QUESTION>, <CASE_BACKGROUND>, and the assumed
     dimension value.
- Do NOT answer the <CASE_QUESTION> itself. Focus ONLY on judging how likely each suspect is
      to be the real murderer under different assumptions about the investigative dimension.
- Do NOT rewrite or restate the <CASE_QUESTION> or <CASE_BACKGROUND>.
- label must be one of "likely", "neutral", or "unlikely".
- The output must include an entry for every combination of dimension value and suspect.

Output format:
Return STRICT JSON only with the following schema:
{{
    "evaluations": [
        [
            {{
                "answer_id": string,
                "dimension_value_id": string,
                "reason": string,
                "label": string
            }},
            ... // one object for each suspect
        ],
        ... // one array for each dimension value
    ]
}}
The "evaluations" field must contain exactly {num_dimension_values} arrays (one per
     dimension value).
Each inner array must contain exactly {num_possible_answers} objects (one per suspect).
```

AR-BENCH-SP

System prompt.

```
You are an expert thinking puzzle analyst. Evaluate how likely each candidate explanation is
     to be the correct hidden explanation of the puzzle under different assumptions about
    the puzzle's hidden dimensions.
```

User prompt.

```
<PUZZLE>
{ambiguous_prompt}
</PUZZLE>

<PUZZLE_CONTEXT>
{meta_context}
</PUZZLE_CONTEXT>

<DIMENSION_NAME>
{dimension_name}
</DIMENSION_NAME>

<DIMENSION_VALUES_WITH_IDS>
{dimension_values_with_ids}
</DIMENSION_VALUES_WITH_IDS>

<POSSIBLE_EXPLANATIONS_WITH_IDS>
{possible_answers_with_ids}
</POSSIBLE_EXPLANATIONS_WITH_IDS>

Definition:
A puzzle dimension is a hidden aspect of the scenario where knowing its true value would
    explain the puzzle.
<DIMENSION_VALUES_WITH_IDS> is a list of dicts with "id" and "text" fields. Each dict
    corresponds to a possible value that the <DIMENSION_NAME> could take.
<POSSIBLE_EXPLANATIONS_WITH_IDS> is a list of dicts with "id" and "text" fields. Each dict
    corresponds to a candidate hidden explanation of the <PUZZLE>.
Let values[i] be the i-th element of <DIMENSION_VALUES_WITH_IDS>.
Let answers[j] be the j-th element of <POSSIBLE_EXPLANATIONS_WITH_IDS>.

Task (row-major order):
For i = 0..len(values)-1:
For j = 0..len(answers)-1:
    - Assume the hidden explanation is such that <DIMENSION_NAME> = values[i]["text"].
    - Judge how likely it is that answers[j]["text"] is the correct hidden explanation of
        the <PUZZLE> under that assumption.

What to generate:
For i = 0..len(values)-1:
For j = 0..len(answers)-1:
    - answer_id: the id of the candidate explanation being evaluated, i.e. answers[j]["id"]
    - dimension_value_id: the id of the dimension value being evaluated, i.e. values[i]["id
        "]
    - reason: a short one-sentence explanation of why answers[j] is likely/neutral/unlikely
        to be the correct hidden explanation given <DIMENSION_NAME> = values[i]["text"].
    - label: one of "likely", "neutral", or "unlikely" according to the following
        definitions:
        - "likely": Given <DIMENSION_NAME> = values[i]["text"] and the clues in <
            PUZZLE_CONTEXT>, answers[j]["text"] is the expected correct hidden explanation
            of the <PUZZLE>.
        - "neutral": Given <DIMENSION_NAME> = values[i]["text"] and the clues in <
            PUZZLE_CONTEXT>, answers[j]["text"] is a plausible explanation but not
            specifically supported; there is insufficient evidence to say it is more or
            less correct than other explanations.
        - "unlikely": Given <DIMENSION_NAME> = values[i]["text"] and the clues in <
            PUZZLE_CONTEXT>, answers[j]["text"] is not expected to be the correct hidden
            explanation of the <PUZZLE>.

Constraints:
- Use ONLY the information provided in <PUZZLE>, <PUZZLE_CONTEXT>, and the assumed dimension
     value.
- Do NOT solve the <PUZZLE> itself. Focus ONLY on judging how likely each candidate
    explanation is to be correct under different assumptions about the puzzle dimension.
- Do NOT rewrite or restate the <PUZZLE> or <PUZZLE_CONTEXT>.
- label must be one of "likely", "neutral", or "unlikely".
- The output must include an entry for every combination of dimension value and candidate
    explanation.

Output format:
Return STRICT JSON only with the following schema:
{{
    "evaluations": [
        [
```

```
            {{
                "answer_id": string,
                "dimension_value_id": string,
                "reason": string,
                "label": string
            }},
            ... // one object for each candidate explanation
        ],
        ... // one array for each dimension value
    ]
}}
The "evaluations" field must contain exactly {num_dimension_values} arrays (one per
     dimension value).
Each inner array must contain exactly {num_possible_answers} objects (one per candidate
     explanation).
```

iCRAFT-MD

System prompt.

```
You are an experienced physician evaluating how likely each candidate diagnosis or clinical
     decision is to be correct under different assumptions about the patient's clinical
     state.
```

User prompt.

```
<CLINICAL_QUESTION>
{ambiguous_prompt}
</CLINICAL_QUESTION>

<PATIENT_INFORMATION>
{meta_context}
</PATIENT_INFORMATION>

<DIMENSION_NAME>
{dimension_name}
</DIMENSION_NAME>

<DIMENSION_VALUES_WITH_IDS>
{dimension_values_with_ids}
</DIMENSION_VALUES_WITH_IDS>

<DIAGNOSTIC_OPTIONS_WITH_IDS>
{possible_answers_with_ids}
</DIAGNOSTIC_OPTIONS_WITH_IDS>

Definition:
A clinical dimension is a specific clinical factor where different values would point toward
      different diagnoses or clinical decisions.
<DIMENSION_VALUES_WITH_IDS> is a list of dicts with "id" and "text" fields. Each dict
     corresponds to a possible value that the <DIMENSION_NAME> could take.
<DIAGNOSTIC_OPTIONS_WITH_IDS> is a list of dicts with "id" and "text" fields. Each dict
     corresponds to a candidate diagnosis or clinical decision for the <CLINICAL_QUESTION>.
Let values[i] be the i-th element of <DIMENSION_VALUES_WITH_IDS>.
Let answers[j] be the j-th element of <DIAGNOSTIC_OPTIONS_WITH_IDS>.

Task (row-major order):
For i = 0..len(values)-1:
For j = 0..len(answers)-1:
    - Assume the patient's true clinical state is <DIMENSION_NAME> = values[i]["text"].
    - Judge how likely it is that answers[j]["text"] is the correct diagnosis or clinical
         decision for <CLINICAL_QUESTION> under that assumption.

What to generate:
For i = 0..len(values)-1:
For j = 0..len(answers)-1:
    - answer_id: the id of the candidate diagnosis being evaluated, i.e. answers[j]["id"]
    - dimension_value_id: the id of the dimension value being evaluated, i.e. values[i]["id
        "]
    - reason: a short one-sentence explanation of why answers[j] is likely/neutral/unlikely
         to be the correct diagnosis given <DIMENSION_NAME> = values[i]["text"].
    - label: one of "likely", "neutral", or "unlikely" according to the following
         definitions:
        - "likely": Given <DIMENSION_NAME> = values[i]["text"] and the patient information
             in <PATIENT_INFORMATION>, answers[j]["text"] is the expected correct diagnosis
             or clinical decision for <CLINICAL_QUESTION>.
```

```
        - "neutral": Given <DIMENSION_NAME> = values[i]["text"] and the patient information
             in <PATIENT_INFORMATION>, answers[j]["text"] is a plausible diagnosis but not
             specifically supported; there is insufficient clinical evidence to say it is
             more or less correct than other options.
        - "unlikely": Given <DIMENSION_NAME> = values[i]["text"] and the patient information
             in <PATIENT_INFORMATION>, answers[j]["text"] is not expected to be the correct
             diagnosis or clinical decision for <CLINICAL_QUESTION>.

Constraints:
- Use ONLY the information provided in <CLINICAL_QUESTION>, <PATIENT_INFORMATION>, and the
     assumed dimension value.
- Do NOT answer the <CLINICAL_QUESTION> itself. Focus ONLY on judging how likely each
     candidate diagnosis is to be correct under different assumptions about the clinical
     dimension.
- Do NOT rewrite or restate the <CLINICAL_QUESTION> or <PATIENT_INFORMATION>.
- label must be one of "likely", "neutral", or "unlikely".
- The output must include an entry for every combination of dimension value and candidate
     diagnosis.

Output format:
Return STRICT JSON only with the following schema:
{{
    "evaluations": [
        [
            {{
                "answer_id": string,
                "dimension_value_id": string,
                "reason": string,
                "label": string
            }},
            ... // one object for each candidate diagnosis
        ],
        ... // one array for each dimension value
    ]
}}
The "evaluations" field must contain exactly {num_dimension_values} arrays (one per
     dimension value).
Each inner array must contain exactly {num_possible_answers} objects (one per candidate
     diagnosis).
```

V.12 Prompt-Sensitivity Experiment

The v0 prompts are the standard iCraft-MD dimension-proposal prompts in Appendix V.2. Variant v1 retains the v0 system prompt but uses a minimally scaffolded user prompt. Variant v2 uses the generic dimension-proposal prompt.

Variant v1: Terse Clinical Prompt

System prompt.

```
You are an experienced physician performing a diagnosis. Your goal is to identify the key
     clinical dimensions that must be clarified to arrive at the correct diagnosis or
     clinical decision.
```

User prompt.

```
<CLINICAL_QUESTION>
{ambiguous_prompt}
</CLINICAL_QUESTION>

<PATIENT_INFORMATION>
{meta_context}
</PATIENT_INFORMATION>

List exactly {num_initial_dims} clinical dimensions whose values would discriminate between
     candidate diagnoses. For each: a one-sentence reason, a short name, and at most {
     max_num_values_per_dim} plausible values.

Return STRICT JSON:
{{
    "dimensions": [
        {{"reason": string, "name": string, "values": [string, ...]}},
        ...
    ]
}}
```

VARIANT V2: DOMAIN-NEUTRAL PROMPT

The system prompt is empty.

User prompt.

```
<AMBIGUOUS_PROMPT>
{ambiguous_prompt}
</AMBIGUOUS_PROMPT>

<META_CONTEXT>
{meta_context}
</META_CONTEXT>

Task:
Identify the dimensions of ambiguity in the <AMBIGUOUS_PROMPT>.

Definition:
A dimension of ambiguity is a specific aspect of the <AMBIGUOUS_PROMPT> that can be
     interpreted in multiple valid ways, leading to different possible meanings or outcomes.
      Once the dimension's value is known, the prompt moves toward a single dominant
     interpretation with a well-defined ground-truth answer.

What to generate:
- Produce a minimal, non-overlapping set of dimensions of ambiguity.
- Produce exactly {num_initial_dims} dimensions of ambiguity.
- Each dimension must correspond to a distinct, intent-level uncertainty.
- If <META_CONTEXT> already resolves a dimension, do not include it.
- If <META_CONTEXT> context proposes some dimensions of ambiguity, use them.

For each dimension of ambiguity, provide :
- reason: a short one-sentence explanation of why this dimension is a source of ambiguity in
      the prompt.
- name: a short, specific label
- values: a list of plausible values, no larger than {max_num_values_per_dim}, that this
     dimension could take in the context of the prompt.

Constraints:
- Use ONLY the information provided in <AMBIGUOUS_PROMPT> and <META_CONTEXT>.
- Do NOT answer the <AMBIGUOUS_PROMPT> itself. Focus ONLY on identifying dimensions of
     ambiguity.
- Do NOT rewrite or restate the <AMBIGUOUS_PROMPT> or <META_CONTEXT>.

Output format:
Return STRICT JSON only with the following schema:
{{
    "dimensions": [
        {{
            "reason": string,
            "name": string,
            "values": [string, ...]
        }},
        ...
    ]
}}
```

SEMANTIC-MATCHING JUDGE

System prompt.

```
You are comparing two sets of clinical dimensions proposed for the same clinical case.
     Identify which dimensions in Set A are semantically equivalent to dimensions in Set B.
```

User prompt.

```
<SET_A>
{a_block}
</SET_A>

<SET_B>
{b_block}
</SET_B>

Task: For each dimension in Set A, identify if there is a semantically equivalent dimension
     in Set B (same underlying clinical concept / observation / factor, even if worded
     differently). Return ONLY the matched pairs.
```

```
Constraints:
- Each dimension in A appears in AT MOST one matched pair.
- Each dimension in B appears in AT MOST one matched pair.
- Use the 0-based indices shown in <SET_A> and <SET_B>.

For each matched pair, provide:
- a_idx: integer index in Set A
- b_idx: integer index in Set B
- reason: one short sentence explaining the equivalence

Return STRICT JSON only.
```

V.13 ORACLE-ALIGNED DST EVALUATION JUDGE

These prompts are used only for post-hoc evaluation.

System prompt.

```
You are aligning two clinical dimension ontologies for the same patient case: (a) oracle
     dimensions derived from the gold patient state, and (b) BALAR's dynamically-proposed
     dimensions. For each oracle dimension, identify the semantically equivalent BALAR
     dimension (if any), the BALAR value tokens that semantically match the oracle's
     optimal_value, and which of BALAR's asked questions probed this oracle dimension.
```

User prompt.

```
<CLINICAL_CASE>
{amb}

{ctx}
</CLINICAL_CASE>

<ORACLE_DIMENSIONS>
{oracle_block}
</ORACLE_DIMENSIONS>

<BALAR_DIMENSIONS>
{balar_block}
</BALAR_DIMENSIONS>

<BALAR_ASKED_QUESTIONS>
{asked_block}
</BALAR_ASKED_QUESTIONS>

Task: For each oracle dimension (one record per oracle_dim_id, all exactly once), produce:
- oracle_dim_id: integer index from <ORACLE_DIMENSIONS> (0-based)
- aligned_balar_dim_id: id of the BALAR dimension that is semantically equivalent to this
     oracle dim, or -1 if BALAR proposed no such dimension
- matching_balar_values: BALAR value tokens (verbatim strings copied from the chosen BALAR
     dim's values list) that are semantically equivalent to this oracle dim's optimal_value;
      [] if no aligned BALAR dim
- probing_question_rounds: the round numbers from <BALAR_ASKED_QUESTIONS> of asked questions
      that probed this oracle dim; [] if none
- reason: one short sentence

Return STRICT JSON only.
```

V.14 QUESTION-QUALITY EVALUATION JUDGE

These prompts are used only for post-hoc evaluation.

System prompt.

```
You are evaluating clarifying questions asked by an expert clinician during a patient
     interview to disambiguate between candidate diagnoses. Score each question for
     relevance to the case, informativeness for the differential, non-redundancy vs prior
     questions, and classify it against a fixed list of proposed clinical dimensions.
```

User prompt.

```
<CLINICAL_CASE>
{amb}

{ctx}
</CLINICAL_CASE>

<CANDIDATE_DIAGNOSES>
{candidates_block}
</CANDIDATE_DIAGNOSES>

<PROPOSED_DIMENSIONS>
{dims_block}
</PROPOSED_DIMENSIONS>

<PRIOR_QUESTIONS>
{prior_block}
</PRIOR_QUESTIONS>

<CURRENT_QUESTION>
{question}
</CURRENT_QUESTION>

Rate the <CURRENT_QUESTION> on a 1-5 integer scale for each:
- relevance: does it relate to the <CLINICAL_CASE>? (1=unrelated, 5=highly relevant)
- informativeness: would the answer narrow the <CANDIDATE_DIAGNOSES>? (1=uninformative, 5=
    substantially narrows)
- non_redundancy: is it asking something new vs the <PRIOR_QUESTIONS>? (1=fully redundant,
    5=entirely novel)

Also identify:
- target_dim_id: integer id from <PROPOSED_DIMENSIONS> that this question primarily targets,
     or -1 if none.
- reason: one short sentence explaining your scoring.

Return STRICT JSON only.
```