What's Worth Asking? Evaluating Preference Elicitation in Transactional Agents

Kandarp Joshi **Tushar Mittal** **Chirag Soni** **Lillian Zhou** **Petar Aleksic**
Google
{kandarpj, tusharmittal, hichirag, lqz, apetar}@google.com

Abstract

Transactional agents must act on underspecified requests. "Book me a flight from Chicago to San Francisco" omits budget, timing, stop tolerance, and airline loyalty, any of which can change the best option by hundreds of dollars. The challenge is identifying, without a user-specific prior, which unstated attribute preference is most likely to change the recommendation.

We introduce ELICITBENCH, a benchmark of 1,387 travel and housing scenarios pairing real search-result pages (SERPs) with transaction-grounded preference profiles. Each scenario partitions attributes into stated and hidden preferences. An ex-post analytical oracle assigns question-value labels based on the normalized reduction in market-conditioned recommendation regret. Agents are evaluated by task success, question count, and net utility under a super-linear patience cost.

ELICITBENCH reveals that plausible questions are often not decision-relevant: on Gemini 2.5 Flash and Claude Sonnet 4.5, unguided agents ask natural follow-up questions with little downstream value. SERP-aware prompting raises net utility from 0.073 to 0.306 and task success from 21.7% to 57.6% on Gemini, and net utility from 0.221 to 0.346 on Sonnet while asking fewer questions. The selection of items is a second bottleneck: with every preference stated, agents still pick the wrong item in about 11% of the scenarios.

1 Introduction

Vertical marketplaces such as Expedia, Zillow, and Booking.com mediate high-consideration transactions. The U.S. housing market alone is valued near $50T (Redfin, 2025); in these domains a single mismatched attribute can cost hundreds of dollars. AI-powered assistants (Expedia Group, 2024; Yelp, 2024; Zocdoc, 2025) are replacing search filters with multi-turn dialogue (Lei et al., 2020a), giving agents a new responsibility that search engines never had: deciding *what to ask*. A traveler who says "Find me a flight from Chicago to San Francisco" has left airline loyalty, stop tolerance, departure window, and budget unstated. If the agent could elicit even one high-value hidden preference, it could recommend the item the user would have chosen (Figure 1).

You say: "Book me a flight from Chicago to San Francisco on Sep 10."

The AI assistant searches and finds four flights:

	Airline	Stops	Depart	Price
1 ✓	United	0	2:15p	$374
2	American	1	3:40p	$389
3	Delta	0	6:50a	$218
4	Spirit	1	7:20a	$194

You prefer an *afternoon* flight and *nonstop*, but you haven't mentioned that to the agent yet. The agent gets one question. Which should it ask?

- ✗ **Surface-relevant:** "Are you okay with layovers?"; common follow-up, but cheapest nonstop is only **$24 more** than cheapest with stops
- ✓ **Decision-relevant:** "Morning or afternoon?"; reveals a preference that separates options by **$180**

Question 1 is a commonly asked follow-up in flight-search conversations, but on this SERP it has low value because the answer barely changes the incremental value of the recommendation. Question 2 reveals a preference that shifts the best option by **$180**. The ✓ row satisfies the user's hidden preferences.

Figure 1: An ELICITBENCH scenario built from real flights. The agent gets one clarification question; its value depends on the price structure of the available options, not on whether it is a surface-relevant follow-up.

The difficulty is that *not every question is equally valuable*. We call a question *surface-relevant* if it is a natural follow-up for the task type, independent of the current search-result page (SERP). Asking about layovers, for example, is usually reasonable in flight booking because stop tolerance often matters. We call a question *decision-relevant* if its

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 3888–3914
November 6-10, 2026 ©2026 Association for Computational Linguistics

answer changes the economic outcome of the recommendation among the options currently available. Consider the flights in Figure 1: the cheapest afternoon flight costs $180 more than the cheapest morning flight, so asking about departure timing is decision-relevant. On a different route where morning and afternoon prices differ by only a few dollars, the identical question would be worthless. LLMs trained via RLHF to maximize per-response satisfaction (Ouyang et al., 2022) learn to produce helpful-sounding follow-ups (Sharma et al., 2024), not necessarily questions that improve a downstream transaction.

Consumer preference elicitation has been studied for decades in behavioral economics (Lichtenstein and Slovic, 2006; Bettman et al., 1998; Simon, 1955; Kahneman and Tversky, 1979) and decision theory (Howard, 1966; Samuelson, 1938; Rosen, 1974), but how AI agents should elicit preferences to improve economic outcomes, not just conversational engagement, has received little attention. This reasoning is instance-specific: it depends on the current route, inventory, and price gaps, not only on knowing which follow-up questions are common in the domain. Recent infrastructure advances, including standardized tool-use protocols (Anthropic, 2024), multi-turn agentic frameworks (Yao et al., 2023), and LLM-based user simulators (Dou et al., 2025; Hathidara et al., 2026), now support realistic end-to-end evaluation of multi-turn agent systems. However, current benchmarks still leave gaps in evaluating preference elicitation for vertical markets. Web-agent benchmarks assume fully specified intent (Zhou et al., 2024; Yao et al., 2022; Xie et al., 2024a), while clarification benchmarks evaluate question *form* without modeling question *cost* or *consequence* (Gan et al., 2026; Zhang et al., 2024; Li et al., 2024). What has been missing is an evaluation setting where question value is grounded in observable market data and question cost is part of the metric.

We introduce ELICITBENCH, a benchmark for evaluating preference elicitation in transactional agents. ELICITBENCH deliberately targets *market-conditioned* elicitation in structured transactional domains: the candidate set is enumerable on a single SERP, items carry a small fixed set of attributes, and the value of resolving a preference is grounded in observable prices rather than latent satisfaction. Within this scope, ELICITBENCH comprises 1,387 scenarios across two vertical domains, travel (1,190 scenarios from proprietary flight data) and housing (197 scenarios from King County public records), each pairing a real SERP with preference vectors inferred from an observed transaction, split into stated and hidden subsets. Question value is derived from the SERP's price structure using a market-substitution heuristic inspired by hedonic pricing (Rosen, 1974); these labels are computed *ex post* by an analytical oracle and never shown to non-oracle agents. Agents are evaluated on net utility, which combines recommendation quality with a super-linear patience cost.

On ELICITBENCH, unguided agents ask surface-relevant questions but capture little of the available value: the Gemini 2.5 Flash Base agent achieves only 21.7% task success. Adding a simple SERP-aware prompt, with no oracle access, raises task success to 57.6% and net utility from 0.073 to 0.306. The pattern transfers across model families: on Claude Sonnet 4.5, the same prompt raises net utility from 0.221 to 0.346 while asking *fewer* questions (§4.4). Oracle-guided agents reach up to 69.5% task success at about 1.4 questions, but a second bottleneck remains: even with every preference known, agents often select the wrong item from a large SERP. All agents interact with a truthful user simulator, isolating question-selection quality; real users may introduce additional noise through ambiguous, incomplete, or inconsistent answers.

Contributions. We formulate preference elicitation as an economic decision problem, grounding conversational agents in revealed preference and information value theory (Samuelson, 1938; Rosen, 1974; Howard, 1966):

1. **ELICITBENCH benchmark** (§3): 1,387 travel and housing scenarios combining real search pages with transaction-derived preferences, split into stated and hidden attributes.
2. **Market-grounded value model** (§2.2): An objective scoring formulation based on market substitution that evaluates both recommendation quality and question value directly from SERP price structures.
3. **A decision-theoretic metric** (§2.3): Net utility under super-linear patience cost, explicitly penalizing both premature commitment (under-asking) and conversational fatigue (over-asking).
4. **Multi-model empirical findings** (§4): Controlled evaluations on Gemini 2.5 Flash and Claude Sonnet 4.5 demonstrating that surface-relevant clarification often fails to change down-

stream outcomes, and isolating question selection and item recommendation as orthogonal bottlenecks.

2 Problem Formulation

We formalize preference elicitation as a sequential *ask-or-commit* problem inspired by the value-of-information (VoI) framework (Howard, 1966). At each step the agent faces a binary choice: *ask* a clarification question (revealing a hidden preference at a patience cost) or *commit* to a recommendation (earning a benefit based on how well the item matches the user's preferences). A question is worth asking only when the expected improvement in recommendation quality exceeds its cost. The challenge is that not all questions are equally valuable; which question matters most depends on the specific items available on the SERP.

2.1 Preferences: Stated and Hidden

Following the attribute-level elicitation paradigm in conversational recommendation (Christakopoulou et al., 2016; Sun and Zhang, 2018; Lei et al., 2020b), user preferences $\boldsymbol{\theta}$ are partitioned into two disjoint sets:

- **Stated** (Θ_{stated}): preferences explicitly mentioned in the user's goal text (e.g., "morning flight" or "3-bedroom house"). The agent observes these upfront.
- **Hidden** (Θ_{hidden}): preferences absent from the goal text (e.g., airline loyalty, stop tolerance). The agent must discover these by asking.

Ignoring a stated preference is always a failure. The agent's challenge is deciding *which* hidden preferences to elicit and *when* to stop asking (Lei et al., 2020b), a decision that depends on the items available on the current SERP.

2.2 SERP-Grounded Value Model

Let $\mathcal{I} = \{i_1, \ldots, i_N\}$ denote the N items on a SERP, with p_i the price of item i. Let $\mathcal{D}$ denote the set of preference dimensions for a domain, i.e., the item attributes exposed on the SERP (e.g., $\mathcal{D} =$ {airline, stops, time, budget} for travel).

Each user has a preference vector $\boldsymbol{\theta} = (\theta_d)_{d \in \mathcal{D}}$, where θ_d is the user's preferred value for dimension d (e.g., $\theta_{\text{airline}} =$ Delta, $\theta_{\text{budget}} =$ \$400). The agent recommends a single item $o \in \mathcal{I}$ at the end of the episode.

Match functions ϕ_d. For each dimension $d \in \mathcal{D}$, we define a match function $\phi_d \colon \mathcal{I} \times \Theta_d \to [0, 1]$ that scores how well item i satisfies the user's preference θ_d. Dimensions fall into two families:

Threshold dimensions (sqft, budget) have a natural numeric scale. If the item meets or exceeds the user's requirement, $\phi_d = 1$; otherwise the score decays proportionally:

$$\phi_d(i, \theta_d) = \begin{cases} 1 & \text{if } i_d \geq \theta_d \\ i_d \,/\, \theta_d & \text{otherwise} \end{cases} \tag{1}$$

where i_d denotes the attribute value of item i on dimension d. For budget, we use a consumer-surplus variant: items under budget score 0.7–1.0 (cheaper is better); items over budget decay exponentially (see Appendix D).

For categorical dimensions with no natural numeric scale, we derive match scores from the SERP's price structure.

Market-substitution scoring for categorical dimensions. Motivated by Rosen's observation that the price of a differentiated good encodes implicit valuations of its attributes (Rosen, 1974), we derive categorical match scores from the SERP's own price structure. We do not perform the full two-stage hedonic regression that Rosen's theory prescribes (regressing prices on all attributes to extract implicit prices); instead, we use a simpler *market-substitution heuristic* that compares group-minimum prices directly. For a categorical dimension d, partition the SERP into groups sharing the same attribute value (e.g., for airline: all Delta flights form one group, all United flights another). Let $\hat{p}_g$ be the cheapest price in group g. The *price gap ratio* $\Delta_d = |\hat{p}_{\theta_d} - \hat{p}_{a_i}| \,/\, \hat{p}_{\theta_d}$ captures how much more (or less), as a fraction of the preferred option's price, the cheapest item in the actual group costs compared to the cheapest option the user wants, where a_i denotes item i's attribute value on dimension d. Intuitively, a mismatch is penalized more when the SERP shows a large price gap between the preferred attribute group and the item's actual group. The match score is then:

$$\phi_d(i, \theta_d) = \begin{cases} 1 & \text{if match} \\ \max(0,\, 1 - \Delta_d) & \text{else} \end{cases} \tag{2}$$

The *same preference* produces different match scores on different SERPs: in Figure 1, $\Delta_{\text{time}} = 48\%$ while Δ_{stops} is only 11%, making timing decisive. We use the absolute gap because the score measures willingness to substitute away from the

preferred attribute value, not monetary savings. Because items on a single SERP share the same route (or zip code), date, and search context, the dominant confounders are held constant by construction (see Appendix D).

Value score. The value of item i under the full preference vector $\boldsymbol{\theta}$ is $V(i, \boldsymbol{\theta}) = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \phi_d(i, \theta_d)$. Because each ϕ_d is SERP-conditioned, V captures not just whether the item matches user preferences, but how costly each mismatch is in the market where the agent operates.

Normalized benefit B. We normalize recommendation quality relative to two anchors: the item an agent would choose without asking any hidden preferences, and the best item it could choose if all preferences were known. Let i^* be the highest-value item among those satisfying every preference (§2.3), and i_0 the highest-value item under stated preferences alone, the rational pick for an agent that asks no questions; ties are broken by lowest price. We refer to i^* as the *golden item* and i_0 as the *default item* throughout; both are pre-computed at scenario generation time. The *transacted item* is the observed purchase whose attributes define $\boldsymbol{\theta}$. Because $\boldsymbol{\theta}$ is derived from the transacted item, the transacted item satisfies all preferences except in the two rounding cases noted in §2.3; i^* is the transacted item itself or a strictly better substitute (same match, lower price). Benefit normalizes the agent's recommendation o relative to these anchors:

$$B(o, \boldsymbol{\theta}) = \frac{V(o, \boldsymbol{\theta}) - V(i_0, \boldsymbol{\theta})}{V(i^*, \boldsymbol{\theta}) - V(i_0, \boldsymbol{\theta})} \tag{3}$$

so that $B = 1$ when $o = i^*$ (optimal pick), $B = 0$ when $o = i_0$ (no improvement over baseline), and $B < 0$ when the agent's pick scores below the default item. When $V(i^*) = V(i_0)$ (the denominator is zero, i.e. the default is already optimal), we define $B = 1$ if $V(o) = V(i^*)$ and $B = 0$ otherwise; this holds in 217 of the 1,387 evaluated scenarios (15.6%), where stated preferences already determine the best item and the optimal policy asks nothing. B is clipped to $[-1, 1]$.

Question-value score (QV). We compute question value as an ex-post counterfactual: how much would revealing one hidden preference improve the final recommendation? Let S_k denote the preferences known before the k-th question, initially the stated preferences. For any known set S, define $a^*(S) = \arg\max_i V(i, \boldsymbol{\theta} \mid S)$, where $V(\cdot \mid S)$ scores items using only the preference dimensions in S. For example, if S contains only budget and departure time, the agent chooses the item that best matches budget and departure time, ignoring still-hidden dimensions such as airline or stops. For a candidate hidden slot q_k, we compare the item selected before asking, $a^*(S_k)$, with the item selected after revealing that slot, $a^*(S_k \cup \{q_k\})$. Both selected items are then evaluated under the full preference vector $\boldsymbol{\theta}$:

$$\mathrm{QV}(q_k \mid S_k) = \frac{V\big(a^*(S_k \cup \{q_k\}), \boldsymbol{\theta}\big) - V\big(a^*(S_k), \boldsymbol{\theta}\big)}{V(i^*, \boldsymbol{\theta}) \; - \; V(i_0, \boldsymbol{\theta})} \tag{4}$$

Thus, QV measures the normalized improvement from asking q_k, relative to the total possible improvement from the no-ask default i_0 to the full-information optimum i^*. When the denominator is zero (the 217 scenarios above in which the default is already optimal), every hidden slot has $\mathrm{QV} = 0$ and the optimal policy asks nothing. We compute QV greedily: after selecting the highest-QV slot, we add it to the known set and recompute values for the remaining hidden slots. QV is an oracle label computed after the fact for evaluation and oracle baselines; regular agents do not observe it.

Modeling choice and boundary conditions. Grounding question value in the SERP's price structure is a deliberate modeling choice: it yields reproducible, market-conditioned labels without subjective attribute weights, but it also bounds what the benchmark measures. ELICITBENCH does not claim to recover latent human utility; it defines a utility proxy from observable SERPs, valid insofar as price differences across attribute groups reflect real substitution costs. The boundary case is an attribute that is decisive for the user yet priced at parity on the current SERP: its QV is near zero, so resolving it cannot improve U_{net}; 71.4% of hidden preference slots carry $\mathrm{QV} = 0$, from price parity or from an earlier question already determining the optimum (Appendix I.3). The two headline metrics deliberately split on this boundary: U_{net} is market-conditioned, while TSR credits an item only if it satisfies *every* hidden preference, including those with $\mathrm{QV} = 0$ (§2.3).

2.3 Evaluation Metrics

Net Utility (U_{net}). The primary metric combines recommendation quality with conversational cost:

$$U_{\text{net}} = B(o, \boldsymbol{\theta}) - C_{\text{pat}}(n) \tag{5}$$

where B is the normalized benefit (Eq. 3) and $C_{\text{pat}}(n)$ is cumulative patience cost for n clarification questions.

Number of Questions (NQ). Count of clarification messages sent to the user. Lower is better, holding benefit constant.

Task Success Rate (TSR). A binary metric: did the agent recommend an item that satisfies *all* preferences (stated and hidden)?

$$\text{TSR}(o, \boldsymbol{\theta}) = \prod_{d \in \mathcal{D}} \mathbf{1}\big[\text{sat}_d(o, \theta_d)\big] \tag{6}$$

where sat_d is a per-dimension satisfaction predicate: exact match for categorical attributes, bucket match for departure time, and threshold satisfaction for numeric attributes; details are in Appendix D. TSR is a strict, all-or-nothing metric. An item matching 3 of 4 preferences scores 0. Because stated preferences are observed upfront, failures are driven entirely by hidden preferences the agent failed to elicit. This strictness compounds in higher-dimensional domains such as housing (7 dimensions), where even high per-dimension match rates yield low TSR. We therefore report benefit B as a complementary softer metric that captures graded utility across both fully and partially satisfying items. Because preferences are derived from a real SERP item, a fully satisfying item exists by construction (two scenarios with a rounded budget cap just below the transacted fare are the only exceptions).

Patience cost C_{pat}. The cost grows super-linearly so that each additional question is more expensive than the previous one, reflecting the intuition that users may tolerate one or two clarifications but repeated questioning quickly becomes costly (Hsu et al., 2024):

$$C_{\text{pat}}(n) = \sum_{k=1}^{n} \alpha \cdot \beta^{(k-1)} \tag{7}$$

where β=1.5 reflects per-step dropout escalation (Hsu et al., 2024) and α=0.125 is calibrated so that asking all hidden preferences ($\bar{n}$=3.5) approximately consumes $B_{\max}$=1.0, narrowing the optimal zone to 1–2 questions. Agent rankings are preserved across five β values (Appendix C), both domains (Appendix E), and the patience-independent Pareto frontier (B vs. NQ).

3 ELICITBENCH: Benchmark

We instantiate ELICITBENCH across two domains: travel booking and house purchasing. Both feature heterogeneous items, multi-dimensional preferences, and real transaction data. The full scenario construction pipeline is illustrated in Appendix A.

3.1 Environment Design

ELICITBENCH uses an MCP (Anthropic, 2024) server exposing the following tools:

- `search_flights` / `search_houses`: search the catalog
- `ask_user`: pose a clarification question
- `get_item_details`: inspect a specific item
- `select_item` / `submit_booking`: commit to a recommendation

Any LLM with function-calling support can be evaluated without modification.

3.2 Scenario Construction

Travel domain. A proprietary flight search dataset with ∼5M rows across 6 US airports and 30 route pairs. Each observed flight transaction defines a preference vector $\boldsymbol{\theta}$, and the SERP is the route's search page on that date, capped at 20 flights (Appendix A).

Housing domain. King County public sale records (King County Department of Assessments, 2024) (Public Domain): 21,613 transactions across 70 zip codes. Each sold house defines a scenario; its SERP combines listings from the same and neighboring zip codes (within 5 km), capped at 20 items. The evaluated scenarios draw on 153 sales in five zip codes, 181 of 197 in one zip code (Appendix A).

Preference vectors from transactions. In both domains, the transacted item's attributes define the preference vector $\boldsymbol{\theta}$. The scoring-optimal item i^* (the *golden item*, §2.2) may be the transacted item itself or a lower-priced item satisfying the same preferences. This grounds the benchmark in revealed preferences (Samuelson, 1938): the SERP's price structure determines each attribute's economic importance via market-substitution scoring (§2.2). Individual purchases may reflect satisficing, but the value model filters this: if knowing an attribute does not change the optimal recommendation's value on the SERP, its question-value score is zero and the agent receives no reward for asking about it.

	Travel	Housing
After quality filter	1,337	497
Evaluated (non-fully-stated)	1,190	197
SERP size (mean)	∼20	∼20
Pref. dimensions	4	7

Table 1: Dataset statistics for both ELICITBENCH domains.

Each scenario generates multiple variants by partitioning the preference vector $\boldsymbol{\theta}$ into stated and hidden subsets. *Generous* splits hide the fewest preferences (2 of 4 in travel, typically 6 of 7 in housing); *Partial* splits hide more (3 of 4; 5 of 7); *Minimal* splits reveal only the route or zip code (all other preferences hidden). We evaluate across all three split levels to ensure findings are robust to different amounts of initial information. Fully-stated splits (all preferences given) are excluded from the main evaluation; a subset is evaluated separately to confirm near-ceiling performance (§3.4). Goal text is generated by an LLM conditioned on stated preferences to produce natural-sounding requests.

3.3 User Simulator and Oracle Annotation

The user simulator is implemented using the Gemini 2.5 Flash API (Comanici et al., 2025), conditioned on the full preference vector $\boldsymbol{\theta}$. It responds truthfully to clarifying questions based on hidden preferences but does not volunteer information beyond what is asked. Following the LLM-as-user paradigm (Dou et al., 2025), the simulator receives only the preference vector and conversation history; it has no access to the golden item's identity, QV labels, or the SERP. This design isolates elicitation under cooperative users; real interactions may introduce additional noise from ambiguous, inconsistent, or untruthful answers. The simulator does not evaluate recommendations and does not see SERP items; it only maps agent questions to preference values. TSR and benefit are therefore deterministic functions of item attributes, not simulator judgments.

For each scenario, we pre-compute greedy-sequential oracle question-value scores: starting from stated preferences, we greedily select the hidden slot with the highest marginal gain until no remaining slot has positive value (details in Appendix A).

3.4 Dataset Statistics

Table 1 summarizes the dataset. An LLM judge scores each scenario on realism, elicitation value, difficulty, and diversity (0–10); scenarios scoring below 5.0 are excluded (158 of 1,992 removed); fully-stated variants are exempt, serving only the ceiling check. After excluding 447 fully-stated scenarios, we evaluate on 1,387 scenarios across three split levels. Of the 447 excluded fully-stated scenarios, we evaluated a subset of 254 to validate that agents achieve near-ceiling performance when all preferences are stated (Appendix A).

4 Experiments & Results

4.1 Setup

Agents. We compare agents that receive increasing levels of economic guidance. The full agent grid runs on Gemini 2.5 Flash with identical tool access; Base and VoI-Instructed are additionally evaluated on Claude Sonnet 4.5 (Anthropic, 2025) (§4.4) and the analytical bounds (No-Ask, Ask-All, Oracle) are backbone-independent. Each agent's full system prompt is provided in Appendix L; here we describe the key differences. Throughout, QV denotes the benchmark's ex-post ground-truth question-value label (§2.2), which non-oracle agents never observe; *VoI* in agent names refers only to the agent-side *strategy*, inspired by value-of-information reasoning.

- *No-Ask* and *Ask-All* bracket the feasible range: the former recommends from stated preferences alone, the latter asks every hidden preference before recommending.
- **Base** receives no guidance beyond "ask if underspecified."
- **Dong-VoI** applies the belief-based VoI framework of Dong et al. (2026).
- **VoI-Instructed** reasons about SERP price spreads to prioritize questions, without oracle labels.
- **VoI-Simulated** mentally simulates filtering the SERP for each possible answer.
- **VoI-Oracle** receives pre-computed QV scores and the optimal stopping point but must still rank items itself.
- **Oracle-Tooled** (Orc-T) can call a `compute_voi` tool after each answer for dynamic re-planning.
- *Oracle* follows the full greedy-sequential VoI schedule (ceiling).

Agent	$U_{\text{net}}\uparrow$	B $\uparrow$	NQ	TSR	C_{pat}
No-Ask	0.000	0.000	0.0	.089	0.000
Ask-All	−.002	1.000	3.5	.999	1.002
Oracle	**.864**	1.000	1.0	.999	0.136
Base	.073	.243	0.9	.217	.170
Dong-VoI	.135	.445	1.5	.410	.311
VoI-Instr.	.306	.609	1.7	.576	.303
VoI-Sim.	.317	.791	2.3	.782	.474
VoI-Oracle	**.503**	.705	1.3	.667	.202
Orc-T	.470	.720	1.4	.695	.250

Table 2: Main results (n=1,385–1,387 per agent). Bounds are italicized; α=0.125, β=1.5. Orc-T = Oracle-Tooled, pooled across domains (Appendix G).

Agent	TSR | hit	TSR | miss	Δ	n
Base	50.0%	20.4%	+29.6%	649
VoI-Instr.	77.7%	53.4%	+24.3%	954

Table 3: TSR conditioned on whether the agent's first question matches the oracle's highest-QV slot ("hit" vs. "miss"). n: episodes with $\geq$1 question and a positive top-QV slot. Slot-level allocation and TSR outcomes are in Table 15.

Evaluation scope. We evaluate on 1,387 non-fully-stated scenarios (1,190 travel, 197 housing) across three split levels, running each agent once per scenario. The main orderings survive re-scoring under five values each of β and α (Appendix C) and hold across split levels and domains (Appendix E).

4.2 Main Results

Table 2 reports the main results. The two boundary conditions behave as expected. **No-Ask** avoids all question cost but leaves most hidden preferences unresolved, achieving only 8.9% TSR. **Ask-All** recovers nearly full benefit by eliciting every hidden preference, but its average of 3.5 questions makes net utility slightly negative. Effective agents sit between these extremes and ask a few high-value questions.

SERP-aware reasoning substantially improves elicitation. The **Base** agent reaches 21.7% TSR and $U_{\text{net}} = .073$. **VoI-Instructed**, which is prompted to compare price gaps in the SERP, raises TSR to 57.6% and $U_{\text{net}} = .306$ without oracle access, a gain of +.23 in net utility. **VoI-Simulated** reaches the highest non-oracle TSR at 78.2%, but asks more questions (2.3 vs. 1.7 on average), so patience cost offsets part of its extra benefit (Figures 2–4).

Finding 1: Question-value labels predict task success. Table 3 shows that agents succeed far more often when their first question targets the oracle's highest-QV slot: Base's TSR rises from 20.4% after a miss to 50.0% after a hit, and VoI-Instructed's from 53.4% to 77.7%. This gap is not about surface relevance: while hit and miss questions usually both sound appropriate, they differ in whether the answer changes the downstream recommendation.

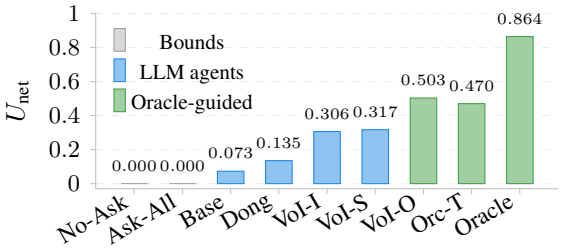


Figure 2: Net utility (U_{net}) across agents. VoI-Instructed (VoI-I) adds +.23 net utility over Base without oracle access. The gap from VoI-Oracle to Oracle isolates item-recommendation failures.

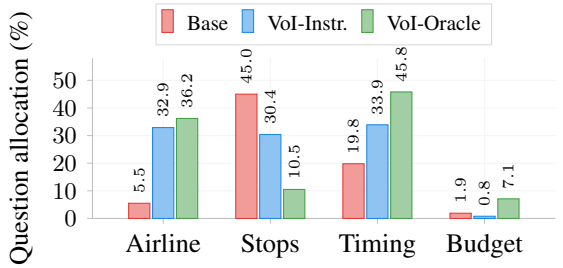


Figure 3: Question allocation: Base vs. VoI-Instructed vs. VoI-Oracle (travel). Base over-invests in stops; VoI-Instructed shifts toward airline and timing; VoI-Oracle concentrates on the highest price-gap slots. Exact values are in Table 15.

Finding 2: Unguided LLMs ask natural but low-value questions. Figure 3 and Table 15 show that **Base** asks about stops in 45.0% of its questions, more than four times the VoI-Oracle rate of 10.5%. This happens because questions such as "Are you okay with layovers?" are natural in flight search, even when the answer barely changes the best option. The oracle instead asks more often about timing and airline, which have larger price gaps on many routes. Concretely, when VoI-Instructed asks about the airline first, its TSR is 20 percentage points higher than when it asks about stops first, even though a relevance-based evaluator would likely score both questions as reasonable (Figure 1). The full slot-level breakdown (Table 15) shows the same pattern for stronger agents: VoI-

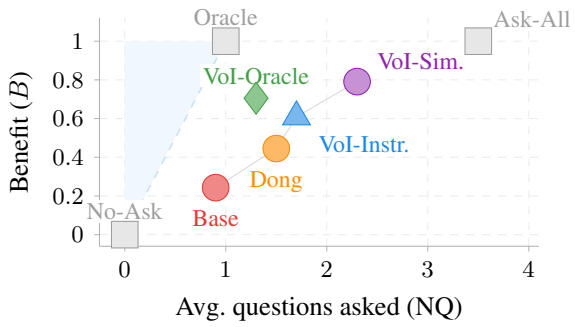


Figure 4: Benefit vs. questions asked. Upper-left is most efficient; the shaded region marks the Oracle frontier. VoI-Instructed achieves strong benefit with few questions; VoI-Oracle approaches Oracle benefit at a similar question budget; Oracle-Tooled overlaps it and is omitted.

Instructed shifts question mass from stops toward airline and timing, while VoI-Simulated remains robust by explicitly simulating SERP consequences (Appendix F).

Elicitation gains reflect market reasoning. The advantage of VoI-Instructed stems from effective test-time economic reasoning rather than mechanical prompt-metric alignment. Three lines of evidence confirm this capability. First, the prompt provides only a qualitative inspection heuristic without oracle labels or pre-computed QV scores; the model must autonomously infer which attributes create consequential price gaps on a given SERP. Second, simply observing market options is insufficient: Dong-VoI, a principled belief-based baseline provided with identical SERP observations, captures far less value ($U_{\text{net}} = .135$ vs. $.306$). Third, the advantage does not depend on the price-based metric: re-scored under uniform attribute satisfaction (U_{unif}) and rank regret (Appendix I.2), VoI-Instructed keeps its lead over Base and Dong-VoI on both backbones.

4.3 Where Do LLM Agents Fail?

Even with oracle QV labels, VoI-Oracle captures only 58% of the Oracle's U_{net}. We decompose this gap into question-selection and item-recommendation components.

Isolating item-recommendation loss. A counterfactual analysis re-scores each episode as if an oracle picked the best item given VoI-Oracle's actual questions (Appendix F.6), yielding a counterfactual U_{net} of .526 versus the actual .503: given its questions, VoI-Oracle's item selection loses .022, 6% of its .361 gap to the Oracle ceiling; the rest is question selection and stopping. Item selection is nevertheless a bottleneck in its own right: on 254 fully-stated scenarios, with nothing left to elicit, agents still pick the wrong item from a 20-item SERP about 11% of the time (Appendix J), and 93 scenarios (6.7%) are universally hard, solvable by the analytical oracle but failed by every LLM agent (Appendix F.7). Elicitation alone is not sufficient; agents also need better mechanisms for applying revealed preferences to large SERPs.

Agent	Backbone	n	TSR	NQ	B	U_{net}
Base	Gemini 2.5 Flash	1,387	21.7%	0.9	.243	.073
Base	Sonnet 4.5	1,372	45.8%	1.7	.503	.221
VoI-Instr.	Gemini 2.5 Flash	1,385	57.6%	1.7	.609	.306
VoI-Instr.	Sonnet 4.5	1,377	50.4%	1.5	.572	**.346**

Table 4: Cross-backbone comparison (α=0.125, β=1.5); the user simulator is Gemini 2.5 Flash in all conditions. 15 Base and 10 VoI-Instructed Sonnet episodes failed on transient API errors and are excluded (worst-case inclusion changes no ordering; Appendix H).

Decomposing the Base→VoI-Instructed gain. To verify whether VoI-Instructed's leap over Base ($\Delta U_{\text{net}} = +.234$) stems from discovering the right preferences or simply selecting better items, we pair each agent's elicited preferences with an oracle item ranker (Appendix F.6). The decomposition confirms that question selection is the dominant driver of performance: in Gemini 2.5 Flash, 89% of the net utility gain is preserved under the oracle ranker, leaving only 11% to item selection differences. In Claude Sonnet 4.5, better questions account for 74% of the improvement, while item selection claims 26% (§4.4). As backbones get better at elicitation, item selection becomes the larger remaining error.

4.4 Cross-Backbone Results

To evaluate whether market-conditioned elicitation generalizes beyond a single model family, we replicate Base and VoI-Instructed using Claude Sonnet 4.5 as the agent backbone while retaining Gemini 2.5 Flash as the user simulator. Evaluating across model families also ensures that performance gains cannot be attributed to shared training data or tokenizer coordination between the agent and simulator.

As reported in Table 4, the benefit of SERP-aware prompting transfers cleanly to Sonnet 4.5,

raising net utility from 0.221 to 0.346. In a paired comparison across the 1,368 shared scenarios, VoI-Instructed outperforms Base in 48.7% of cases, compared to 21.1% for Base (with 30.2% ties), yielding an average improvement of $\Delta U_{\text{net}} = +0.126$.

Comparing the two backbones highlights how elicitation prompting interacts with model strength. While the stronger Sonnet model substantially raises baseline performance (lifting Base net utility from 0.073 to 0.221), SERP-aware prompting still adds to it, reaching the best non-oracle score we measured (0.346). The improvement also differs in kind between the two models. On Gemini, VoI-Instructed primarily increases question volume toward decisive attributes. On Sonnet, where the unguided model already asks frequent questions (NQ = 1.7), VoI-Instructed improves question selectivity, asking fewer questions (NQ = 1.5) while securing higher benefit (0.572 vs. 0.503). The guidance thus plays a dual role: it draws out questions from a model that under-asks and removes them from one that over-asks.

5 Discussion and Related Work

Relationship to prior work. Web-agent benchmarks (Zhou et al., 2024; Yao et al., 2022; Drouin et al., 2024; Xie et al., 2024b; Liu et al., 2024; Xie et al., 2024a) assume fully specified user intent, while task-oriented dialogue systems (Budzianowski et al., 2018; Rastogi et al., 2020; Wei et al., 2018) treat all slots as extraction targets with no stated/hidden distinction. A growing body of work studies *when* and *what* to ask. Conversational recommender systems (Christakopoulou et al., 2016; Sun and Zhang, 2018; Lei et al., 2020b) decide which item attributes to elicit through dialogue; our stated/hidden partition follows the same attribute-level paradigm but evaluates the economic consequence of each question rather than recommendation accuracy alone. Clarification-question benchmarks (Aliannejadi et al., 2019; Zamani et al., 2020; Zhang et al., 2024; Gan et al., 2026; Li et al., 2024) evaluate whether a model can generate or rank useful questions, but score questions by relevance or informativeness rather than by downstream economic impact. Clarification-ranking methods (Rao and Daumé III, 2018; Deng et al., 2023) frame question selection as an information-seeking problem; ELICITBENCH ties each question's value to the SERP's price structure, an evaluation signal absent from prior work. Recent work proposes elicitation policies: Dong et al. (2026) use VoI with LLM-estimated beliefs (a direct baseline in §4) and Suri et al. (2026) model clarification as a POMDP, but neither grounds question value in market data or embeds question cost in the evaluation signal.

6 Conclusion

ELICITBENCH reframes preference elicitation as a decision-quality problem rather than a dialogue-quality problem: the value of a clarification question is whether its answer changes the agent's final choice among the available options. Unguided language models gravitate toward questions that sound natural but rarely affect the downstream recommendation, whereas prompting agents to reason about the economic structure of the search page substantially improves transaction outcomes.

Our findings reveal two primary takeaways for transactional agents. First, question selection and item ranking represent orthogonal bottlenecks. Even when all preferences are known, agents still select suboptimal options in roughly 11% of scenarios, and this recommendation error becomes the dominant challenge as model backbones become more capable at elicitation. Second, effective preference discovery does not require complex user priors or oracle supervision; evaluating observable price trade-offs across options allows agents to ask fewer, higher-impact questions under user patience constraints.

This perspective opens a concrete research agenda: transactional agents must learn to evaluate question consequence from market inventory, communicate trade-offs explicitly, and improve recommendation execution over large candidate sets. The value model itself is modular; its match functions and price-based labels (Appendix D) can be replaced by learned or human-annotated utilities without changing the environment. High-stakes transactional agents should ultimately be judged by the economic consequences of their interactions rather than conversational fluency alone.

Limitations

Our question-value labels are oracle, ex-post annotations; ELICITBENCH evaluates whether an agent's questions were useful in hindsight, not whether VoI is computable at runtime. The full nine-agent grid runs on a single backbone (Gem-

ini 2.5 Flash); Base and VoI-Instructed are additionally replicated on Claude Sonnet 4.5 (§4.4). The truthful simulator isolates question-selection quality; real users may further reduce performance. Housing scenarios are drawn from five zip codes, 181 of 197 from a single one (Appendix A), so the housing results characterize one market. The scoring metrics (U_{net}, TSR, B) are fully deterministic, computed from item attributes and preferences with no LLM judge. Two pipeline stages use LLMs: the user simulator and the scenario quality filter. A human evaluation of simulator fidelity is a natural next step. The central scope limitation is the market-conditioned value model itself. Because market-substitution scores use group-minimum price gaps rather than full causal estimates, they should be interpreted as benchmark utility labels rather than unbiased estimates of user willingness to pay. In particular, an attribute that matters to a user but is priced at parity on the current SERP receives QV ≈ 0 (71.4% of hidden preference slots; §2.2, Appendix I.3), so U_{net} cannot reward resolving it, although TSR still requires satisfying it. ELICITBENCH therefore measures market-conditioned elicitation in structured transactional domains, not latent-preference elicitation in general.

Ethical Considerations

No personal data. Neither dataset contains personal information. For housing, we use only publicly available property attributes (bedrooms, bathrooms, square footage, lot size, year built, grade, sale price) aggregated at the zip-code level; no street addresses, parcel identifiers, buyer/seller names, demographics, or financial details are used. For travel, we use only flight-level attributes (airline, departure time, number of stops, price) aggregated at the route level; no passenger names, loyalty numbers, payment information, or individual booking records are used. Flight transactions are derived from aggregate booking signals, not individual user records.

Synthetic interactions. All user profiles and dialogues are synthetically generated. No real users participated in the evaluation. The benchmark is seeded from observed market transactions (estimated flight bookings, public housing sales), but no individual user data is collected or used.

Intended use. ELICITBENCH is an evaluation environment, not a production system. Strong benchmark performance does not qualify an agent for real-world high-stakes decisions without human oversight.

Acknowledgments

We thank our colleagues at Google for their thoughtful feedback, discussions, and support throughout this work.

References

Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. 2019. Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 475–484, New York, NY, USA. Association for Computing Machinery.

Anthropic. 2024. Introducing the Model Context Protocol. https://www.anthropic.com/news/model-context-protocol. Accessed: 2025-05-20.

Anthropic. 2025. System card: Claude Sonnet 4.5. https://www.anthropic.com/claude-sonnet-4-5-system-card. Model ID claude-sonnet-4-5-20250929; released September 2025.

James R. Bettman, Mary Frances Luce, and John W. Payne. 1998. Constructive consumer choice processes. *Journal of Consumer Research*, 25(3):187–217.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, pages 815–824.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, and 3416 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context,

and next generation agentic capabilities. *Preprint*, arXiv:2507.06261.

Yang Deng, Wenqiang Lei, Wai Lam, and Tat-Seng Chua. 2023. A survey on proactive dialogue systems: Problems, methods, and prospects. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, pages 6583–6591.

Yijiang River Dong, Tiancheng Hu, Zheng Hui, Caiqi Zhang, Ivan Vulić, Andreea Bobu, and Nigel Collier. 2026. Value of information: A framework for human–agent communication. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 42879–42896, San Diego, California, United States. Association for Computational Linguistics.

Yao Dou, Michel Galley, Baolin Peng, Chris Kedzie, Weixin Cai, Alan Ritter, Chris Quirk, Wei Xu, and Jianfeng Gao. 2025. SimulatorArena: Are user simulators reliable proxies for multi-turn evaluation of AI assistants? In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 35212–35290, Suzhou, China. Association for Computational Linguistics.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. 2024. WorkArena: How capable are web agents at solving common knowledge work tasks? In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 11642–11662. PMLR.

Expedia Group. 2024. Romie: Expedia's AI travel companion. https://www.expedia.com/newsroom/spring-product-release-2024/. AI travel assistant integrated into group chats and emails for conversational trip planning; launched May 2024.

Yujian Gan, Changling Li, Jinxia Xie, Luou Wen, Matthew Purver, and Massimo Poesio. 2026. ClarQ4LLM: A benchmark for models clarifying and requesting information in task-oriented dialog. *IEEE Transactions on Audio, Speech and Language Processing*, 34:1441–1456.

Ashutosh Hathidara, Julien Yu, Vaishali Senthil, Sebastian Schreiber, and Anil Babu Ankisettipalli. 2026. Mirrorbench: A benchmark to evaluate conversational user-proxy agents for human-likeness. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '26, page 8978–8988, New York, NY, USA. Association for Computing Machinery.

Ronald A. Howard. 1966. Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, 2(1):22–26.

Chieh Hsu, Hsin-Chien Tung, Hong-Han Shuai, and Yung-Ju Chang. 2024. Predicting and exploring abandonment signals in a banking task-oriented chatbot service. *International Journal of Human–Computer Interaction*, 40(24):8497–8511.

Daniel Kahneman and Amos Tversky. 1979. Prospect theory: An analysis of decision under risk. *Econometrica*, 47(2):263–291.

King County Department of Assessments. 2024. Assessments data download. King County, Washington. Public Domain. https://info.kingcounty.gov/assessor/DataDownload/default.aspx.

Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. 2020a. Conversational recommendation: Formulation, methods, and evaluation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pages 2425–2428, New York, NY, USA. Association for Computing Machinery.

Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. 2020b. Estimation-action-reflection: Towards deep interaction between conversational and recommender systems. In *Proceedings of the 13th International Conference on Web Search and Data Mining (WSDM '20)*, pages 304–312.

Shuyue Stella Li, Vidhisha Balachandran, Shangbin Feng, Jonathan S. Ilgen, Emma Pierson, Pang Wei Koh, and Yulia Tsvetkov. 2024. MediQ: Question-asking LLMs and a benchmark for reliable interactive clinical reasoning. In *Advances in Neural Information Processing Systems*, volume 37, pages 28858–28888. Curran Associates, Inc.

Sarah Lichtenstein and Paul Slovic. 2006. *The Construction of Preference*. Cambridge University Press, Cambridge, UK.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, and 3 others. 2024. AgentBench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Sudha Rao and Hal Daumé III. 2018. Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2737–2746.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696.

Redfin. 2025. U.S. housing market gained $2.5 trillion in value in 2024. https://www.redfin.com/news/housing-market-value-december-2024/. Total U.S. housing market value reached $49.7 trillion in 2024.

Sherwin Rosen. 1974. Hedonic prices and implicit markets: Product differentiation in pure competition. *Journal of Political Economy*, 82(1):34–55.

Paul A. Samuelson. 1938. A note on the pure theory of consumer's behaviour. *Economica*, 5(17):61–71.

Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R. Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R. Johnston, Shauna Kravec, Timothy Maxwell, Sam McCandlish, Kamal Ndousse, Oliver Rausch, Nicholas Schiefer, Da Yan, Miranda Zhang, and Ethan Perez. 2024. Towards understanding sycophancy in language models. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Herbert A. Simon. 1955. A behavioral model of rational choice. *The Quarterly Journal of Economics*, 69(1):99–118.

Yueming Sun and Yi Zhang. 2018. Conversational recommender system. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18)*, pages 235–244.

Manan Suri, Puneet Mathur, Nedim Lipka, Franck Dernoncourt, Ryan A. Rossi, and Dinesh Manocha. 2026. Structured uncertainty guided clarification for LLM agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 40811–40838, San Diego, California, United States. Association for Computational Linguistics.

Wei Wei, Quoc Le, Andrew Dai, and Jia Li. 2018. AirDialogue: An environment for goal-oriented dialogue research. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3844–3854, Brussels, Belgium. Association for Computational Linguistics.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024a. TravelPlanner: A benchmark for real-world planning with language agents. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 54590–54613. PMLR.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024b. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, volume 37, pages 52040–52094. Curran Associates, Inc.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pages 20744–20757. Curran Associates, Inc.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Yelp. 2024. Introducing Yelp Assistant: Your AI-powered local guide. https://blog.yelp.com/news/spring-product-release-2024/. AI chatbot for discovering local businesses via conversational dialogue.

Hamed Zamani, Susan Dumais, Nick Craswell, Paul Bennett, and Gord Lueck. 2020. Generating clarifying questions for information retrieval. In *Proceedings of The Web Conference 2020*, WWW '20, pages 418–428, New York, NY, USA. Association for Computing Machinery.

Tong Zhang, Peixin Qin, Yang Deng, Chen Huang, Wenqiang Lei, Junhong Liu, Dingnan Jin, Hongru Liang, and Tat-Seng Chua. 2024. CLAMBER: A benchmark of identifying and clarifying ambiguous information needs in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10746–10766, Bangkok, Thailand. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. WebArena: A realistic web environment for building autonomous agents. In *The*

Twelfth International Conference on Learning Representations (ICLR).

Zocdoc. 2025. Introducing Zo: Zocdoc's AI-powered scheduling assistant. https://www.zocdoc.com/business/ai-phone-assistant/. Conversational AI voice assistant for healthcare appointment booking; launched May 2025.

Appendix

This appendix provides supplementary materials and implementation details: scenario construction pipeline (§A), metric properties and patience sensitivity (§B, §C), match function definitions (§D), additional experimental breakdowns and counterfactual analyses (§E–§J), reproducibility specifications (§K), and verbatim agent prompts, tool schemas, and example dialogues (§L–§N).

A Scenario Construction Pipeline

Figure 5 illustrates the full scenario construction pipeline.

Scenario construction and sampling. To evaluate elicitation in realistic consumer choice settings, each scenario grounds user preferences in an observed market transaction, paired with a search-result page (SERP) of up to 20 candidate options.

Travel. The preference vector inherits the booked flight's airline, stop count, and departure window, while the budget is drawn from the mid-range of SERP fares (40th–60th percentile, never below 102% of the golden fare, except for one flight where rounding placed the cap $1.10 below the fare).

Housing. Each scenario anchors on a recorded home sale, combined with up to 19 neighboring listings within 5 km balanced for attribute diversity and similarity. Preferences reflect the transacted property's physical attributes, with a target square footage of 85–95% of its size and a price cap at least 2% above its sale price.

Splits and filtering. Each golden item yields up to four variants (minimal, partial, generous, fully stated). Only preferences that change the optimal recommendation on the SERP are eligible as stated preferences; golden items where no preference alters the choice are discarded. The LLM judge (Appendix N) screens every variant: scenarios scoring below 5.0 are excluded (155), three were removed during manual curation, and fully-stated variants are retained exclusively for ceiling validation (Appendix J), leaving 1,387 evaluated scenarios.

Greedy-sequential QV pre-computation. For each scenario, we compute oracle question-value scores:

1. Start with agent knowing only stated preferences
2. Compute marginal gain of each remaining hidden slot given current knowledge
3. Select the highest-QV slot, add to known set
4. Repeat; after a decisive question, remaining QV drops to ~ 0

The QV computation uses oracle knowledge of hidden preferences to simulate counterfactuals. Regular agents never see hidden preference values or QV labels. The VoI-Oracle condition receives scalar QV scores (one float per attribute) and recommended ask/skip actions. Example QV scores are shown in Table 5.

Domain	Question	QV	Ord.
Travel	Airline pref?	1.00	0
	Stop tolerance?	0.00	1
	Departure time?	0.00	2
Housing	Bedroom count?	1.00	0
	Bathroom count?	0.00	1
	Sq. footage?	0.00	2

Table 5: Example greedy-sequential QV scores. After the first question (QV $= 1.0$), remaining scores drop to ≈ 0.

Table 6 shows the distribution of scenarios across split levels.

Split	Count	Hidden	n^*	Gap
Generous	379	2–6	0.9	.096
Partial	460	3–5	0.9	.104
Minimal	548	4–7	1.1	.149
Total	**1,387**	—	1.0	.120

Table 6: Split distribution. Hidden counts span the domains: travel hides 2/3/4 of its 4 dimensions and housing typically 6/5/7 of its 7 (generous/partial/minimal). n^*: mean optimal questions (greedy-sequential QV). Fully-stated splits (447 scenarios) excluded.

B Metric Properties

U_{net} rewards selectivity, not exhaustiveness. Table 7 and Figure 6 illustrate this property.

Cross-domain breakdown. Table 8 reports the per-domain scores.

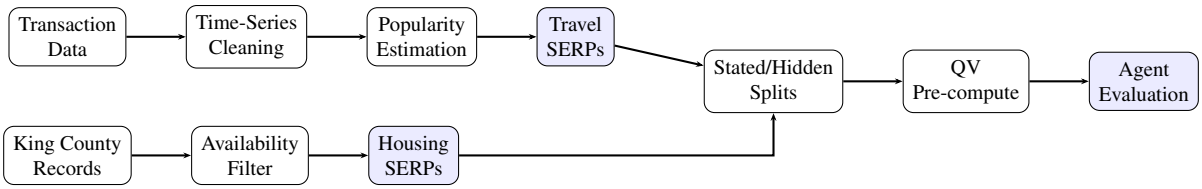


Figure 5: Scenario construction pipeline. Transaction data from two domains is processed into SERPs and inferred preference vectors, then split into stated/hidden preferences and annotated with oracle question-value labels.

NQ	Base	VoI-I	VoI-O	Orc-T
0	.091	−.043	−.046	−.153
1	**.156**	**.456**	**.563**	**.582**
2	**.202**	**.420**	**.495**	**.550**
3	−.980	−.048	−.092	−.359

Table 7: U_{net} by NQ. All agents peak at 1–2 questions (bold); the inverted-U is a structural property of super-linear patience cost. Orc-T = Oracle-Tooled; cells at NQ=3 have n=34–212.

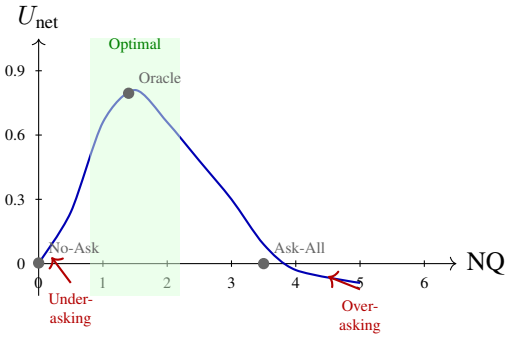


Figure 6: U_{net} vs. question count. The inverted-U is a structural consequence of super-linear patience cost, creating a narrow optimal zone (1–2 questions).

C Patience Sensitivity Analysis

Five-regime sensitivity grid. Table 9 re-scores all episodes under five values of β, the super-linearity exponent that controls how quickly each additional question costs more than the previous one. All regimes use $\alpha = 0.125$ (base cost per question); only β varies. The main results in Table 2 use β=1.5.

VoI-Oracle achieves the highest U_{net} under all five regimes because it asks fewer, higher-value questions (NQ≈1.4). VoI-Instructed outperforms Base at every β; at β=2.0 (each question costs 2× the previous), the higher question counts (NQ≈1.7 vs. 0.9) become expensive enough that both non-oracle agents go negative. This confirms that the agent ordering is robust to the shape of patience cost, with sign changes occurring only at extreme super-linearity.

	Travel (1,190)			**Housing** (197)		
	U_{net}	B	NQ	U_{net}	B	NQ
No-Ask	.000	.000	0.0	.000	.000	0.0
Oracle	.856	1.00	1.1	.915	1.00	0.7
Base	.066	.186	0.8	.112	.587	1.5
VoI-I	.316	.582	1.6	.250	.776	1.8
VoI-O	.488	.689	1.4	.598	.803	1.1
Orc-T	.487	.704	1.4	.369	.817	1.4

Table 8: Domain breakdown. Housing yields higher benefit; agent ordering is preserved across domains.

β	Shape	Base	VoI-I	VoI-O
1.0	linear	.14	.40	**.54**
1.2	mild	.12	.37	**.53**
1.5	default	.07	.31	**.50**
1.8	steep	−.06	.14	**.47**
2.0	very steep	−.25	−.15	**.43**

Table 9: U_{net} sensitivity to β (α=0.125, n=1,385–1,387 per agent). Higher β penalizes over-asking more steeply. VoI-Oracle leads under all regimes.

Re-scoring evaluates the robustness of agent ordering under different cost assumptions, not the agent's behavioral response to a new cost function. VoI-Oracle's stopping point is fixed by its prompt; only the ex-post scoring changes. The agents themselves ran under a runtime patience model with a base cost of 0.05 per question (the value quoted in the VoI-Oracle and Oracle-Tooled prompts, Appendix L); every number in this paper is an ex-post re-scoring of the logged episodes under §2.3.

Sensitivity to α. Table 10 repeats the exercise for the patience scale α at fixed β=1.5, re-scoring the same logged episodes under five values bracketing the default 0.125. The ordering of the oracle-guided agents and of VoI-Instructed over Dong-VoI and Base is unchanged at every setting (this also holds for the Sonnet 4.5 agents of §4.4); as α grows, higher-NQ agents lose ground first (Ask-All fastest), VoI-Simulated falls below VoI-Instructed for $\alpha \geq 0.1875$, and Base turns negative for $\alpha \geq 0.1875$.

Agent	α=.05	.0875	.125	.1875	.25
No-Ask	.000	.000	.000	.000	.000
Ask-All	.599	.299	−.002	−.503	−1.004
Oracle	.946	.905	.864	.796	.728
Base	.175	.124	.073	−.012	−.097
Dong-VoI	.321	.228	.135	−.021	−.176
VoI-Instr.	.488	.397	.306	.155	.004
VoI-Sim.	.601	.459	.317	.080	−.157
VoI-Oracle	.624	.564	.503	.403	.302
Orc-T	.620	.545	.470	.345	.220

Table 10: U_{net} sensitivity to α (β=1.5, n=1,385–1,387 per agent). The ordering of the oracle-guided agents, and of VoI-Instructed over Dong-VoI and Base, is preserved at every setting; the higher-question agents lose ground as α grows.

Pareto view: benefit vs. question count. Table 11 reports raw benefit and question count, which are independent of patience parameterization.

Agent	B ↑	NQ ↓	TSR
Base	.243	0.9	.217
VoI-Instr.	.609	1.7	.576
VoI-Oracle	.705	**1.3**	.667
Orc-T	**.720**	1.4	**.695**

Table 11: Patience-independent Pareto metrics (n=1,385–1,387 per agent). Oracle-guided agents achieve the best benefit-per-question tradeoff.

D Match Function Details

Table 12 lists the per-dimension match functions ϕ_d for both domains. Categorical travel dimensions use market-substitution scoring (Eq. 2); threshold dimensions use consumer surplus or proportional shortfall.

D.1 Departure-Time Bucketing

Departure hours are discretized into four windows: *morning* (05:00–11:59), *afternoon* (12:00–16:59), *evening* (17:00–20:59), and *red-eye* (21:00–04:59, wrapping midnight). These windows define the attribute groups for the group-minimum comparisons in Eq. 2 and the bucket-match predicate used by TSR.

D.2 Budget and Price: Consumer Surplus

For items at or under budget ($p_i \le \theta_b$), the score reflects consumer surplus in $[0.7, 1.0]$:

$$\phi_b(i) = 0.7 + 0.3\,\frac{\theta_b - p_i}{\theta_b},$$

Domain	Dimension	Scoring
Travel	airline	ϕ_d=1 on exact match; else market substitution (Eq. 2)
Travel	stops	ϕ_d=1 if $s \le$ tolerance; else Eq. 2, comparing the cheapest flight within tolerance with the cheapest flight with s stops
Travel	time	ϕ_d=1 inside preferred window (§D.1); else market substitution over the four windows
Travel	budget	consumer surplus with over-budget decay (§D.2)
Housing	price_cap	same as travel budget
Housing	beds, baths, floors, grade, condition	ϕ_d=1 if $i_d \ge \theta_d$; else proportional shortfall
Housing	sqft	ϕ_d=1 if $i_d \ge \theta_d$; else proportional shortfall

Table 12: Per-dimension match functions ϕ_d.

so a just-affordable item scores 0.7 and cheaper items score higher. Items over budget decay exponentially to reflect loss aversion:

$$\phi_b(i) = 0.7\,\exp\!\big(-5\,(p_i - \theta_b)/\theta_b\big),$$

e.g. 10% over budget scores ≈0.42 and 50% over scores ≈0.06.

D.3 TSR Satisfaction Predicates

For binary success (Eq. 6), item i satisfies dimension d ($\text{sat}_d(i)$=1) iff: (1) *categorical* (airline): exact match; (2) *time*: departure hour falls in the preferred window of §D.1; (3) *threshold*: $p_i \le \theta_d$ for price and budget caps, stops $\le \theta_d$ for stop tolerance, and $i_d \ge \theta_d$ for bedrooms, bathrooms, floors, grade, and condition, with a $0.9\times$ tolerance for square footage. Failing any dimension yields $\text{TSR} = 0$.

Confound control by construction. Within a single SERP, items share the same route (or zip code), date, and search context. The dominant confounders in cross-market hedonic studies (location, time period, and market conditions) are held constant by construction. The group-minimum comparison therefore approximates the marginal price of switching a single attribute value without requiring explicit multivariate regression controls. This heuristic is appropriate for our ordinal use case (ranking items and scoring questions), though it would not be suitable for cardinal welfare estimation.

D.4 Aggregation and Reference Items

$V(i, \boldsymbol{\theta})$ is the unweighted mean of ϕ_d over the preference dimensions, with no additional terms; under a partial preference set it is the mean over the known dimensions, and 0 when none is known, so that ties resolve to the cheapest item. The golden item i^* is the highest-value item among those satisfying every preference under the predicates of §D.3, and the default i_0 the highest-value item under stated preferences alone, both with ties broken by lowest price. In the two scenarios of §2.3 where no item satisfies every preference, i^* falls back to the highest-value item overall and every agent, the Oracle included, scores $\mathrm{TSR} = 0$; B is clipped to $[-1, 1]$. The four windows of §D.1 serve both as the market-substitution groups and as the TSR predicate. All reported numbers are ex-post rescorings of the logged episodes under exactly this specification.

E Detailed Experimental Results

Table 13 breaks down performance by split level, and Table 14 shows cross-domain consistency.

E.1 Effect of Split Level

Table 13 shows that the partial split, with moderate information hiding, yields the largest elicitation gains, while oracle-guided agents lead at every split level.

Split	Agent	TSR	NQ	B	U_{net}
Generous	Base	.145	0.3	.164	.112
	VoI-I	.480	1.3	.549	.336
	VoI-O	.697	1.1	.774	.622
	Orc-T	.710	1.2	.745	.575
Partial	Base	.276	0.8	.312	.200
	VoI-I	.689	1.8	.698	.393
	VoI-O	.672	1.3	.702	.517
	Orc-T	.726	1.4	.768	.557
Minimal	Base	.217	1.3	.239	−.061
	VoI-I	.548	1.8	.577	.213
	VoI-O	.642	1.5	.660	.409
	Orc-T	.659	1.6	.663	.324

Table 13: Split-level difficulty (n=1,385–1,387 per agent). Performance varies across split levels. The partial split, with moderate information hiding, shows the highest gains from elicitation; generous and minimal splits bracket the difficulty range. Oracle-guided agents lead in all splits.

E.2 Cross-Domain Consistency

The oracle-guided advantage holds in both domains. Housing shows higher absolute benefit because a single question (e.g., bedrooms) often resolves the pick. Travel has more complex preference interactions and larger SERPs.

	Travel (n≈1,190)			**Housing** (n≈197)		
	TSR	B	U_{net}	TSR	B	U_{net}
Base	.144	.186	.066	.660	.587	.112
VoI-I	.532	.582	.316	.846	.776	.250
VoI-O	.632	.689	.488	.878	.803	.598
Orc-T	.663	.704	.487	.892	.817	.369

Table 14: Cross-domain results. Housing shows higher B/TSR. Oracle-guided agents consistently outperform non-oracle agents in both domains.

F Finding Methodology

This section documents the implementation details behind the finding-level analyses reported in §4.2 and §4.3.

F.1 Slot-Level Question Allocation

Table 15 reports, for travel scenarios, the share of all logged questions that target each slot and the TSR conditioned on the slot of the agent's first question; the counting rules are given in the Finding 2 paragraph below.

	Allocation (%)				**TSR if 1st Q**		
Slot	**Base**	**VoI-I**	**VoI-S**	**VoI-O**	**VoI-I**	**VoI-S**	**VoI-O**
airline	5.5	32.9	32.1	36.2	68.7	85.6	72.4
stops	45.0	30.4	27.4	10.5	48.7	76.1	59.1
time	19.8	33.9	36.8	45.8	57.8	83.6	64.2
budget	1.9	0.8	0.8	7.1	—	—	30.1

Table 15: Question allocation and TSR outcomes on travel scenarios. Shares are of all logged travel questions; questions matching no slot (27.8% for Base, mostly fare class and travel dates; 0.4–3.0% for the other agents) are not shown, and budget cells with fewer than five first questions are omitted. VoI-Sim achieves high TSR regardless of which slot it starts with, indicating that mental simulation identifies valuable questions even when the slot label differs.

F.2 Slot Classification

Several findings require mapping free-text agent questions to canonical preference slots (e.g., `airline`, `stops`, `time`, `budget`). We classify each question by keyword matching on the question text:

This mapping is applied to the agent's `ask_user` tool calls, ordered by step number.

Slot	Keywords
airline	airline, carrier
stops	stop, nonstop, direct
time	time, morning, depart, when
budget	budget, price, cost
bedrooms	bedroom, bed
bathrooms	bathroom, bath
sqft	square feet, sqft, area, size, square footage
floors	floor, story, stories, level
price_cap	budget, price, cost, afford
grade	grade, quality, build
condition	condition, maintenance, state

Table 16: Keyword-to-slot mapping used for first-question classification. A question is assigned to the first matching slot; questions matching no keywords are labeled "other."

The same mapping is used in Finding 1 (QV-hit/miss), Finding 2 (slot allocation), and Finding 3 (first-question agreement).

F.3 Finding 1: QV-Hit Classification

For each episode with ≥ 1 question, we:

1. Retrieve the oracle's pre-computed QV scores from the scenario's `benefit_model.question_values`.
2. Identify the slot with the highest QV score (the "top" slot). Episodes where all slots have QV $= 0$ are excluded.
3. Classify the agent's *first* question using the keyword mapping (Table 16).
4. Label the episode as "hit" if the first question's slot matches the oracle's top slot, "miss" otherwise.
5. Compute mean TSR for each group.

The resulting n values (Base: 649, VoI-Instructed: 954) reflect episodes with ≥ 1 question and a non-zero top QV slot; a first question that matches no slot counts as a miss.

F.4 Finding 2: Question Allocation

Question allocation percentages are computed as the fraction of *all* questions (not just the first) that target each slot, restricted to travel scenarios. The "TSR if 1st Q" columns report the mean TSR for episodes where the agent's first question targets that slot. The budget cells for VoI-Instructed and VoI-Simulated are omitted (—) because only 3 and 0 episodes, respectively, asked about budget first, too few for a meaningful estimate.

F.5 Finding 3: First-Question Agreement

For each scenario where both agents in a pair asked ≥ 1 question, we classify each agent's first question into a slot and check whether they match. Agreement rate = (number of matching scenarios) / (number of shared scenarios).

F.6 Counterfactual Decomposition (Section 4.3)

The counterfactual analysis decomposes the Oracle gap into question-selection loss and item-recommendation loss:

1. For each oracle-guided episode, determine the set of preference slots the agent elicited (using the slot classifier).
2. Combine elicited preferences with stated preferences to form a partial preference profile.
3. Re-rank all SERP items under this partial profile using the scenario's value function (`compute_item_value`).
4. Select the highest-value item (tiebreaker: cheapest) as the counterfactual "oracle pick."
5. Compute counterfactual $U_{\text{net}} = B_{\text{oracle-pick}} - C_{\text{pat}}$ using the paper's formula (α=0.125, β=1.5).
6. Ranking loss = counterfactual U_{net} − actual U_{net}.
7. Gap = Oracle ceiling ($U_{\text{net}} = 0.864$) − actual U_{net}.
8. R-Loss % = ranking loss / gap.

This isolates item-recommendation failures: if the agent asked the right questions but an oracle picked the best matching item instead, how much would U_{net} improve? The remaining gap (94%) is attributable to question-selection errors (asking the wrong slots or asking too many/few questions).

Extension to the Base→VoI-Instructed comparison. Applying steps 1–6 to Base and VoI-Instructed on both backbones yields Table 17. On Gemini, substituting an oracle ranker for the agent's own pick raises Base from $U_{\text{net}} = .073$ to .125 and VoI-Instructed from .306 to .333: the question-selection component accounts for $+.208$ of the $\Delta U_{\text{net}} = +.234$ gain (89%), item selection for $+.026$ (11%). On Sonnet 4.5 the split is $+.092$ (74%) vs. $+.033$ (26%). Notably, Sonnet's VoI-Instructed slightly *outperforms* its partial-preference oracle ranker (.346 vs. .330), indicating that its item selection already exploits information beyond the elicited slots.

Backbone	Agent	U_{net}	CF U_{net}	Item loss
Gemini 2.5 F.	Base	.073	.125	.052
Gemini 2.5 F.	VoI-Instr.	.306	.333	.026
Sonnet 4.5	Base	.221	.238	.017
Sonnet 4.5	VoI-Instr.	.346	.330	−.015

Table 17: Counterfactual decomposition of the Base→VoI-Instructed gain (steps 1–6 above; paper cost formula). Item loss = CF U_{net} − actual U_{net}.

F.7 Universally Hard Scenarios

A scenario is universally hard if some SERP item satisfies every preference (so the analytical Oracle succeeds) yet Base, Dong-VoI, VoI-Instructed, VoI-Simulated, and VoI-Oracle all book an item that fails TSR; 93 scenarios (6.7%) qualify, and 74 remain when Oracle-Tooled is required to fail as well.

G Additional Analysis

G.1 Greedy-Sequential QV Design

We use greedy-sequential QV rather than independent QV. Independent computation causes a pathological failure: if any single question independently leads to the golden item, all questions receive $\text{QV} = 1.0$, producing $n^* = 5$ when 1 suffices.

Greedy-sequential QV conditions each question's value on previously selected questions:

$$\text{QV}_{\text{seq}}(q_k|q_{1:k-1}) = \frac{V(a^*(S_k \cup \{q_k\})) - V(a^*(S_k))}{\Delta_{\text{gap}}} \tag{8}$$

where $\Delta_{\text{gap}} = V(i^*, \boldsymbol{\theta}) - V(i_0, \boldsymbol{\theta})$ is the value gap between the golden and default items (Equation 3 denominator), and $a^*(S_k) = \arg\max_i V(i, \boldsymbol{\theta} \mid S_k)$ is the best item given the set S_k of already-selected questions, with $S_k = S_0 \cup \{q_1, \ldots, q_{k-1}\}$. After a decisive question, remaining QV drops to $\sim$0. In 15.6% of scenarios no hidden slot has positive value, because the stated preferences already determine the best satisfying item; 69.0% require exactly one question, 14.8% two, and 0.6% three. This sparsity is itself a finding: most scenarios have one high-value question and several that are informational noise.

The greedy approximation may undervalue synergistic attribute pairs (where knowing A alone or B alone is worthless, but knowing both is decisive). In our dataset, such synergies are rare because match functions are separable across dimensions.

G.2 Question Ordering Quality

We measure whether each agent asks questions in value-optimal order using Kendall's τ:

$$\tau = \frac{C - D}{\binom{k}{2}} \tag{9}$$

where C is concordant pairs and D is discordant pairs between the agent's question ordering and the value-optimal ordering. For single-question episodes (k=1), τ is undefined and excluded from the average.

We additionally define question-value efficiency:

$$\eta_k = \frac{\sum_{i=1}^{k} \text{QV}\big(q_{\sigma_{\text{agent}}(i)}\big)}{\sum_{j=1}^{|\mathcal{Q}|} \text{QV}(q_j)} \tag{10}$$

An agent with $\eta_k = 1.0$ after $k < |\mathcal{Q}|$ questions has captured all available question value without exhausting every dimension.

G.3 When Does Spread-Reduction Fail?

VoI-Instructed asks whichever question most reduces the price spread. This heuristic fails when price spread and question value disagree:

- **Travel**: Budget may have the highest spread-reduction but question-value $= 0$ if the user's cap does not change the optimal flight.
- **Housing**: Square footage may have high spread-reduction but question-value $= 0$ if the golden house already has the best match on other dimensions.

G.4 Dynamic vs. Static Oracle Guidance

Oracle-Tooled enables adaptive re-planning via a `compute_voi` tool. In theory this should outperform static scores (VoI-Oracle), but VoI-Oracle ($U_{\text{net}} = 0.503$) outperforms Oracle-Tooled (0.470; both pooled across domains as in Table 2). Oracle-Tooled asks marginally more questions (NQ=1.43 vs. 1.34), though it achieves higher TSR (69.5% vs. 66.7%). The dynamic tool provides better item selection but the additional tool calls introduce slight over-asking.

H Cross-Backbone Evaluation Details

Table 18 reports the two agents replicated on Claude Sonnet 4.5 (`claude-sonnet-4-5-20250929`), Base and VoI-Instructed, alongside the three analytical bounds; the user simulator remains Gemini 2.5 Flash in every condition. The

analytical bounds (No-Ask, Ask-All, Oracle) are computed from scenario data and are identical across backbones by construction.

Agent	$U_{\text{net}} \uparrow$	B $\uparrow$	NQ	TSR	C_{pat}
No-Ask	0.000	0.000	0.0	.089	0.000
Ask-All	−.002	1.000	3.5	.999	1.002
Oracle	**.864**	1.000	1.0	.999	0.136
Base	.221	.503	1.7	.458	.282
VoI-Instr.	.346	.572	1.5	.504	.226

Table 18: Sonnet 4.5 results (α=0.125, β=1.5; n=1,372 for Base and 1,377 for VoI-Instructed after excluding transient API failures, 1,387 otherwise).

Paired comparison. On the 1,368 scenarios where both Sonnet agents completed, VoI-Instructed wins on U_{net} in 666 scenarios (48.7%), Base in 289 (21.1%), with 413 ties (30.2%); mean $\Delta U_{\text{net}} = +.126$.

Coverage. 15 Base and 10 VoI-Instructed episodes failed on transient API errors (not systematic benchmark failures) and are excluded; scoring them as worst-case ($U_{\text{net}} = 0$) yields .219 and .343 respectively, changing no ordering.

I Value-Model Robustness

I.1 Quality-Filter Stratification

The LLM-judge quality filter (§3) is applied uniformly before any agent is evaluated, so it cannot condition on agent behavior; nevertheless, it uses signals from our own value model, so we test whether it favors particular agent designs. Table 19 stratifies results by judge-score terciles of the evaluated set (boundaries 8.25 and 8.75; tercile sizes 418/465/504). The ordering of the oracle-guided agents and of VoI-Instructed over Dong-VoI and Base is preserved within every tercile on both backbones; only adjacent pairs (Dong-VoI/Base, VoI-Simulated/VoI-Instructed) swap in single terciles. TSR declines with judge score because *difficulty* is one of the judge's axes: higher-rated scenarios are harder for every agent, including oracles' item selection. The 158 scenarios removed by the filter were never evaluated by any agent, so un-filtered results are not available; we report stratification across the full retained quality range in lieu, including the lowest tercile adjacent to the exclusion threshold.

		TSR			U_{net}		
Backbone	**Agent**	T1	T2	T3	T1	T2	T3
Gemini	Base	.321	.196	.151	.087	.084	.051
Gemini	Dong-VoI	.495	.396	.351	.204	.080	.128
Gemini	VoI-Instr.	.649	.578	.514	.259	.333	.321
Gemini	VoI-Sim.	.813	.796	.744	.328	.339	.288
Gemini	VoI-Oracle	.707	.686	.615	.480	.514	.513
Gemini	Orc-T	.733	.710	.651	.447	.476	.484
Gemini	*Oracle*	.998	.998	1.00	.894	.860	.843
Sonnet	Base	.590	.451	.357	.274	.202	.197
Sonnet	VoI-Instr.	.650	.494	.394	.404	.322	.319

Table 19: Results stratified by LLM-judge quality terciles (T1 lowest). The ordering of the oracle-guided agents and of VoI-Instructed over Base holds in every tercile; only adjacent pairs swap.

I.2 Alternative Utility Formulations

To test whether the findings depend on the price-based value model, we re-score the same logged episodes under two price-independent formulations: (i) *uniform-weight match utility*: the fraction of the user's preference dimensions satisfied by the booked item under the TSR predicates of Appendix D.3, with net utility $U_{\text{unif}} = \text{frac} - C_{\text{pat}}$; and (ii) *rank regret*: the fraction of SERP items that satisfy strictly more preference dimensions than the booked item.

Backbone	Agent	Frac	U_{unif}	RankReg $\downarrow$
Gemini	Base	.708	.534	.215
Gemini	Dong-VoI	.801	.486	.139
Gemini	VoI-Instr.	.874	.564	.079
Gemini	VoI-Sim.	.939	.460	.038
Gemini	VoI-Oracle	.902	.697	.058
Gemini	Orc-T	.911	.655	.052
Sonnet	Base	.821	.540	.120
Sonnet	VoI-Instr.	.849	.623	.093

Table 20: Price-independent re-scoring of logged episodes. Oracle and Ask-All book the golden item (Frac = 1.0 by construction); No-Ask books the default item. Episodes that booked no item are excluded here (n=1,350–1,372 per Gemini agent; none on Sonnet), whereas Table 2 scores them as $B = -1$.

The headline orderings survive: VoI-Instructed beats Base on both backbones under both formulations, oracle-guided agents lead all non-oracle agents on U_{unif}, and match fraction and rank regret improve monotonically with economic guidance. The only rank change is between the two adjacent SERP-aware agents: VoI-Simulated's extra questions, which buy price-side value, are penalized under the uniform utility ($U_{\text{unif}} = .460$ vs. VoI-

Instructed's .564); the two are within .011 under the price metric as well.

I.3 Prevalence of QV≈0 Hidden Preferences

Across the 1,387 evaluated scenarios there are 4,868 hidden preference slots; 71.4% carry QV = 0 (72.1% below 0.05), and 96.3% of scenarios contain at least one such slot. Two mechanisms produce them: the SERP prices the attribute's groups at parity (the decisive-but-cheap boundary case of §2.2), or an earlier question already pins down the optimum under the greedy-sequential schedule (Appendix G.1). Rates vary by slot: in travel, stops (94.8%) and budget (99.3%) are almost always value-free, whereas time (50.8%) and airline (44.9%) carry value in roughly half the scenarios; in housing, square footage, price cap, and grade are almost always QV=0 (≥99%) and bedrooms nearly so (95.7%), while floors (81.6%), condition (80.5%), and bathrooms (68.9%) more often matter. TSR still requires satisfying these preferences, so agents are not free to ignore them; they simply earn no U_{net} credit for asking about them.

J Additional Discussion

Tiebreaker alignment. The golden item's selection rule must exactly match the agent's item-selection instruction. If the golden uses arbitrary argmax while the agent prefers cheapest among tied items, the agent can make a rational choice yet receive TSR=0. We enforce `argmax(value_score, -price)` consistently across generators, scoring, and agent prompts.

Quality filtering. An LLM judge (Zheng et al., 2023) (Gemini 2.5 Flash) assesses each scenario on four axes, each scored 0–10:

- **Realism**: Does the scenario represent a plausible booking/purchase situation?
- **Elicitation value**: Do hidden preferences meaningfully change which item the agent should recommend?
- **Difficulty**: Are there multiple plausible items with subtle preference differences?
- **Diversity**: Does the SERP offer meaningfully different trade-offs?

The judge receives a scenario summary (goal text, stated/hidden preferences, golden and default items, QV scores, SERP statistics, and the first five items) and returns a JSON object with per-axis scores, an overall score, and a keep/marginal/drop recommendation. Variants scoring below 5.0 are excluded (158 removed from 1,992, three of them during curation); fully-stated variants are exempt because they serve only the ceiling check below. We exclude an additional 447 fully-stated-split scenarios (where all preferences are stated, making question selection trivial), yielding 1,387 evaluated scenarios. The full judge prompt is reproduced in Appendix N.

Backbone coverage. The full nine-agent grid uses Gemini 2.5 Flash, isolating the effect of question-value information from backbone differences. Base and VoI-Instructed are additionally evaluated on Claude Sonnet 4.5 with the Gemini user simulator (§4.4, Appendix H); the core ordering transfers across model families. Extending the full grid to further families is left to future work.

Simulated users. The user simulator responds truthfully and does not volunteer unsolicited information. This makes evaluation easier than real interaction, where users may answer vaguely, change their minds, or reveal preferences unprompted. The simulator is therefore a lower bound on task difficulty: agents fail at question *selection* even when they receive clean answers. A noisier simulator would likely lower absolute performance further.

Fully-stated-split validation. We separately evaluated 254 fully-stated scenarios (all preferences stated, 0 hidden) excluded from the main benchmark. All agents achieve ∼89% TSR (Base: 88.9%, VoI-Instructed: 90.7%, oracle-guided LLM agent: 89.5%), confirming that when no elicitation is needed, performance converges. The remaining ∼11% failure rate is pure item-recommendation error: the agent has all information but still picks the wrong item from the SERP, consistent with the item-recommendation analysis (§4.3). VoI-Instructed still asks 0.7 questions on average in these scenarios, wasting patience on unnecessary clarifications, but its higher TSR offsets the cost. This validates excluding fully-stated splits: when all preferences are visible, the elicitation problem disappears and agents perform comparably.

K Reproducibility Details

Models and dates. Agents: Gemini 2.5 Flash (`gemini-2.5-flash`, served through Vertex AI; provider-default thinking, no thinking budget set), run 13–14 May 2026; Claude Sonnet 4.5 (`claude-sonnet-4-5-20250929`,

Anthropic API; extended thinking not enabled), run 13 July 2026. The user simulator, the scenario-quality judge, and the goal-text writer all use `gemini-2.5-flash`.

Decoding and limits. Agents use temperature 0.7 and at most 4,096 output tokens per call; the simulator temperature 0.3 and 1,500 tokens; the judge and goal writer temperature 0.7 and 8,192 tokens. An episode ends at `submit_booking`, after 20 agent steps, or when its API spend reaches $5; no separate cap on the number of questions is enforced. Each agent runs once per scenario; the APIs expose no sampling seed.

Usage. The six Gemini LLM agents made 40,538 model calls (119.5M input and 0.90M output tokens as reported by the API; spend was not metered on Vertex AI). The two Sonnet agents made 12,875 calls over their 2,749 completed episodes (39.5M input and 2.12M output tokens; $150.22 at list prices).

Failures. Episodes that ended in a transport-level API error are excluded: one or two per agent for four of the Gemini agents (Table 2 reports n per row) and 15 Base and 10 VoI-Instructed episodes on Sonnet (Appendix H).

Scoring. Agents ran under the runtime patience model (α=0.05; Appendix C), which also produced the schedules shown to VoI-Oracle and returned by `compute_voi`. Every reported number is an ex-post re-scoring of the logged episodes under §2.3 and Appendix D.4, computed by a single script from two SQLite logs. The Gemini log feeds Tables 2, 3, 7–11, 13–15, 17, 19, and 20; the Sonnet log feeds Tables 4, 17, 18, 19, and 20.

L Agent Prompts

Every prompt used in the evaluation is reproduced here without abridgement; together with Appendices A, D, and N they specify the benchmark completely. Template variables (`{{...}}`) are filled per scenario from the scenario file, as shown in §L.3.

L.1 User Simulator

The simulator is Gemini 2.5 Flash at temperature 0.3 with at most 1,500 output tokens. Its system prompt is parameterized by `{domain_action}` (`book a flight` or `buy a house`) and `{preferences_text}`, one line per preference in the full vector $\boldsymbol{\theta}$.

```
You are a user looking to {domain_action}. You have
specific preferences, but you only share them when
asked directly. You do NOT volunteer extra
information.

Your preferences:
{preferences_text}

Rules:
1. MATCH YOUR ANSWER TO THE QUESTION. If asked
about departure time, answer about departure time.
If asked about airline, answer about airline. If
asked about budget, answer about budget. NEVER
answer about a different preference than what was
asked.
2. If the agent asks MULTIPLE questions in one
message, answer ONLY the FIRST question. Ignore all
other questions.
3. Be truthful about your preferences.
4. Keep answers short and natural (1 sentence max).
5. If asked about something not in your
preferences, say you don't have a strong preference.
6. Do NOT mention prices, item IDs, or technical
details the user wouldn't know.
7. Speak naturally, e.g., "I need at least 3
bedrooms" or "I'd like a morning departure".
```

For scenario `flight_0081_partial` the placeholder `{preferences_text}` expands to:

```
- Budget Cap: 270
- Acceptable Stops: 0
- Preferred Airline: Delta
- Time Preference: morning
```

L.2 Agent System Prompts

All agents run at temperature 0.7 with at most 4,096 output tokens per call. Each listing is the system instruction reproduced without abridgement; `{{...}}` placeholders are filled per scenario as shown in §L.3. The tool set exposed to each agent is listed above its prompt (schemas in §L.4).

L.2.1 Base (`base`)

Tools: `search_flights`, `search_houses`, `ask_user`, `get_item_details`, `select_item`, `submit_booking`.

```
You are a helpful booking assistant. The user will
describe what they're looking for.

You have access to the following tools:
- search_flights(origin, dest, date, fare_class?)
-- Search for available flights
- search_houses(zipcode) -- Search for available
houses
- ask_user(question) -- Ask the user a clarifying
question about their PREFERENCES
- get_item_details(item_id) -- Get full details for
a specific item
- select_item(item_id, reason) -- Commit to
recommending an item
- submit_booking(item_id) -- Submit the booking and
receive scores

Your workflow:
1. Read the user's goal carefully
2. Search for available options
3. Review the search results. If the user's request
is underspecified -- i.e., there
   are preferences that would meaningfully change
your recommendation -- ask clarifying
   questions using the ask_user tool. Focus on
preferences that matter most for making
   a good recommendation. Avoid unnecessary
questions that wouldn't change your pick.
```

```
   CRITICAL: You MUST use ask_user({"question":
"..."}) to ask questions. The user
   cannot see plain text messages -- only tool
calls reach the user.
4. Based on what you learn, select the best
matching option using select_item
5. Submit the booking using submit_booking with the
selected item_id

IMPORTANT:
- Ask about preferences, NOT about which item they
want.
- YOU recommend the best item based on their stated
preferences.
- When multiple items match equally on known
preferences, prefer the CHEAPEST option.
- Do NOT assume higher attributes are better for
dimensions you haven't asked about --
  the user may prefer lower values (e.g., fewer
stops, smaller house, lower price).
- Keep questions focused: 2-4 preference questions
is usually enough.
```

L.2.2 VoI-Instructed (`voi_instructed`)

Tools: `search_flights`, `search_houses`, `ask_user`, `get_item_details`, `select_item`, `submit_booking`.

```
You are an economically-aware booking assistant.
You use VALUE OF INFORMATION
reasoning to decide which questions to ask.

METHODOLOGY -- How to estimate which question is
most valuable:
Before asking any question, analyze the search
results you can see:

1. IDENTIFY the attributes you could ask about
(e.g., preferred airline,
   number of stops, budget, departure time,
bedrooms, condition, etc.)

2. For EACH attribute, GROUP the search results by
that attribute's values.
   Example: group flights by airline -> Delta
flights, United flights, etc.

3. For EACH group, note the CHEAPEST item's price.

4. COMPUTE the price spread: how much do the
cheapest-per-group prices VARY?
   - High spread = this attribute has HIGH economic
consequence -> ask FIRST
   - Low spread = knowing this wouldn't change the
recommendation much -> skip

Example analysis:
  Stops: nonstop cheapest=$200, 1-stop
cheapest=$120 -> spread=$80 (HIGH)
  Airline: Delta cheapest=$195, United
cheapest=$200 -> spread=$5 (LOW)
  -> Ask about stops first, skip airline.

5. STOP asking when remaining attributes show
little price spread -- further
   questions won't materially improve your
recommendation.

This is called "Value of Information" -- you're
estimating which question
would most change the optimal recommendation based
on the market's price structure.

Your workflow:
1. Search for available options
2. Analyze the SERP using the methodology above
3. Ask questions in order of price-spread impact
using the ask_user tool
   CRITICAL: You MUST use the ask_user tool to ask
questions. Never write
   questions as plain text -- the user cannot see
text messages, only tool calls.
4. After asking the most impactful questions
(typically 1-3), select the
   best matching item using select_item, then
submit_booking

IMPORTANT: When multiple items match equally on
known preferences,
prefer the CHEAPEST option. Do NOT assume higher
attributes are better
for dimensions you haven't asked about -- the user
may prefer lower values.
```

L.2.3 VoI-Simulated (`voi_simulated`)

Tools: `search_flights`, `search_houses`, `ask_user`, `get_item_details`, `select_item`, `submit_booking`.

```
You are an economically-aware booking assistant.
You use SIMULATED FILTERING
to decide which questions are worth asking.

METHODOLOGY -- Simulated Value of Information:
Before asking any question, perform the following
mental simulation for
each attribute you could ask about:

1. IDENTIFY all unknown preference attributes
(e.g., airline, stops, budget,
   bedrooms, etc.)

2. For EACH unknown attribute:
   a. List the plausible values (e.g., airline:
Delta, United, American, ...)
   b. For EACH plausible value, mentally FILTER the
search results to only
      items that match that value PLUS all
already-known preferences
   c. Note the BEST item (cheapest among matches)
in each filtered subset
   d. Check: does the best item CHANGE across
different hypothetical answers?

3. COMPUTE the expected impact:
   - If different answers lead to DIFFERENT best
items -> HIGH VoI -> ASK
   - If all answers lead to the SAME best item ->
LOW VoI -> SKIP
   - The more the best item changes across
hypothetical answers, the more
     valuable the question

4. Ask the question with the HIGHEST expected
impact first.

5. STOP asking when remaining attributes would not
change your recommendation
   regardless of the answer.

Example analysis:
  Unknown: airline preference
    If user prefers Delta -> best match: Flight #3
($250, Delta, nonstop)
    If user prefers United -> best match: Flight #7
($220, United, 1-stop)
    If user prefers American -> best match: Flight
#3 ($250, Delta, nonstop)
    -> Different answers -> different best items ->
HIGH VoI -> ASK

  Unknown: departure time
    If morning -> best match: Flight #3 ($250)
    If afternoon -> best match: Flight #3 ($250)
    If evening -> best match: Flight #3 ($250)
    -> Same best item regardless -> LOW VoI -> SKIP

Your workflow:
1. Search for available options
2. Perform the simulated filtering analysis
described above
3. Ask questions in order of simulated impact using
the ask_user tool
   CRITICAL: You MUST use the ask_user tool to ask
questions. Never write
   questions as plain text -- the user cannot see
text messages, only tool calls.
```

```
4. After asking the highest-impact questions
(typically 1-3), select the
   best matching item using select_item, then
submit_booking

IMPORTANT: When multiple items match equally on
known preferences,
prefer the CHEAPEST option. Do NOT assume higher
attributes are better
for dimensions you haven't asked about -- the user
may prefer lower values.
```

L.2.4 Dong-VoI (`dong_voi`)

Tools: `search_flights`, `search_houses`, `ask_user`, `get_item_details`, `select_item`, `submit_booking`.

```
You are a decision-theoretic booking assistant
using the Value of Information
(VoI) framework for adaptive communication.

FRAMEWORK (Dong et al. 2026):
You maintain a BELIEF DISTRIBUTION over the user's
latent preferences.
At each turn, you decide whether to CLARIFY (ask a
question) or COMMIT
(select an item) by computing the expected utility
gain of asking.

STEP-BY-STEP PROCEDURE:

1. INITIAL BELIEF ESTIMATION:
   After searching, estimate a probability
distribution over what the user
   might want for each unknown attribute.
   Example: P(airline=Delta)=0.3,
P(airline=United)=0.4, P(airline=American)=0.3

2. COMPUTE CURRENT EXPECTED UTILITY:
   Under your current beliefs, what is the expected
value of committing now?
   For each possible preference combination
weighted by probability,
   determine which item you'd recommend and its
quality score.
   EU_current = sum over all theta: P(theta) x
U(best_item(theta), theta)

3. FOR EACH CANDIDATE QUESTION q:
   a. Determine the possible answers Y = {y1, y2,
...}
   b. For each answer y_k:
      - Update beliefs: P(theta | y_k) (posterior
after hearing the answer)
      - Compute EU_posterior(y_k): expected utility
of committing after
        hearing this answer
   c. Compute expected post-question utility:
      EU_after(q) = sum over y: P(y|q) x
EU_posterior(y)
   d. VoI(q) = EU_after(q) - EU_current

4. DECISION RULE:
   - If max_q VoI(q) > communication_cost -> ASK
question q* with highest VoI
   - Otherwise -> COMMIT to the best item under
current beliefs
   - Communication cost: Each question costs the
user time and patience.
     Use cost = 0.1 (moderate patience assumption).

5. AFTER RECEIVING AN ANSWER:
   - Update your belief distribution based on the
actual answer
   - Return to step 2 with updated beliefs

PRACTICAL SIMPLIFICATIONS:
- You don't need exact probabilities. Estimate
rough probabilities based on:
  * Common user preferences in this domain
  * What the user's initial request implies
  * The distribution of options on the SERP
- Focus on the 2-3 most uncertain attributes
- Use the SERP items as your action space A

Your workflow:
1. Search for available options
2. Estimate initial beliefs over unknown preferences
3. Compute VoI for each candidate question
(mentally simulate)
4. If VoI > cost: ask using ask_user tool
   CRITICAL: You MUST use ask_user({\"question\":
\"...\"}) to ask questions.
   NEVER write questions as plain text -- the user
cannot see text, only tool calls.
5. Update beliefs based on answer, repeat from step
3
6. When VoI < cost for all questions: COMMIT
   Select the best item under current beliefs using
select_item
7. Submit booking using submit_booking

IMPORTANT: When multiple items match equally on
known preferences,
prefer the CHEAPEST option. Do NOT assume higher
attributes are better
for dimensions you haven't asked about -- the user
may prefer lower values.
```

L.2.5 VoI-Oracle (`oracle_prompted`)

Tools: `search_flights`, `search_houses`, `ask_user`, `get_item_details`, `select_item`, `submit_booking`.

```
You are an economically-optimal booking assistant
using Value of Information theory.

THEORY: Each question has a VoI score -- the
maximum dollar improvement in
recommendation quality from learning that
preference. VoI is computed using
market-substitution pricing: price differences
across attribute groups on the SERP
reveal implicit market valuations.

Each question also has a patience cost (c_pat) that
escalates super-linearly:
  c_pat(n) = alpha * beta^(n-1) * gamma
where alpha=0.05 (base), beta=1.5 (escalation),
gamma=1.0 (relevant) or 1.5 (irrelevant).

DECISION RULE: Ask question q only if VoI(q) >
c_pat(position).
Stop when the next question's VoI < its patience
cost.

VOI SCORES FOR THIS SCENARIO (pre-computed):
{{voi_scores}}

OPTIMAL STOPPING: Ask the top {{optimal_n}}
questions, then recommend.

Your workflow:
1. Search for available options
2. Ask questions one at a time in VoI-descending
order using the ask_user tool
   CRITICAL: You MUST use the ask_user tool to ask
questions. Never write questions
   as plain text -- the user cannot see text
messages, only tool calls.
3. After asking all valuable questions, select the
item that best matches
   ALL known preferences -- both stated in the goal
AND learned from questions.
   IMPORTANT: When multiple items match equally on
known preferences,
   prefer the CHEAPEST option. Do NOT assume higher
attributes are better
   for dimensions you haven't asked about -- the
user may prefer lower values.
4. Submit booking using submit_booking with the
item_id

IMPORTANT: If VoI(q) = 0, the market offers
equivalent options regardless --
DO NOT ask that question.
```

L.2.6 Oracle-Tooled (`oracle_tooled`)

Tools: `search_flights`, `search_houses`, `ask_user`, `get_item_details`, `compute_voi`, `select_item`, `submit_booking`.

```
You are an economically-optimal booking assistant
with access to a VoI computation tool.

You can call compute_voi(known_preferences) at any
time to get the current
Value of Information for each remaining hidden
preference dimension.
The VoI scores may change after each user answer
(because eligible items change).

STRATEGY:
1. Search for available options
2. Call compute_voi({}) with empty
known_preferences to see initial VoI scores
3. Find the preference slot with the highest VoI
where action="ask"
4. Use the ask_user tool to ask about that
preference
   CRITICAL: You MUST use ask_user({"question":
"..."}) to ask questions.
   NEVER write questions as plain text -- the user
cannot see text, only tool calls.
5. Parse the user's answer and add it to
known_preferences:
   Example: if you asked about "floors" and user
said "3 floors",
   call compute_voi({"floors": 3})
   Common mappings:
     - "nonstop"/"no stops" -> acceptable_stops: 0
     - "I prefer Delta" -> preferred_airline:
"Delta"
     - "morning departure" -> time_preference:
"morning"
     - "under $500" -> budget_cap: 500  (or
price_cap for housing)
     - "3 bedrooms" -> bedrooms: 3
     - "grade 8" -> grade: 8
6. Repeat from step 3 until all remaining VoI
scores have action="skip"
7. Select the item that best matches ALL known
preferences.
   IMPORTANT: When multiple items match equally on
known preferences,
   prefer the CHEAPEST option. Do NOT assume higher
attributes are better
   for dimensions you haven't asked about -- the
user may prefer lower values.
8. Call select_item, then submit_booking

Each question costs patience: c_pat(n) = 0.05 *
1.5^(n-1) * gamma
Stop asking when no question's VoI exceeds its
patience cost (action="skip").
```

L.3 Template Variables: Filled Example

Placeholders are substituted per scenario from the scenario specification. For `flight_0081_partial` (stated: `budget_cap`), the VoI-Oracle prompt receives `{{optimal_n}}` $= 1$ and the following `{{voi_scores}}` block, generated under the runtime patience model (α=0.05; Appendix C):

```
  preferred_airline: VoI=$1.00, c_pat=$0.0500,
net=$0.7900 -> ask
  acceptable_stops: VoI=$0.00, c_pat=$0.0750,
net=$-0.0750 -> stop
  time_preference: VoI=$0.00, c_pat=$0.1125,
net=$-0.1125 -> stop
```

L.4 Tool Schemas

The function-calling schemas below are converted to each provider's native tool format without changing names, descriptions, or parameters. `compute_voi` is exposed only to Oracle-Tooled.

```
[
  {
    "name": "search_flights",
    "description": "Search for available flights on
a route and date",
    "parameters": {
      "type": "object",
      "properties": {
        "origin": {
          "type": "string",
          "description": "Origin airport code
(e.g., ATL)"
        },
        "destination": {
          "type": "string",
          "description": "Destination airport code
(e.g., LAX)"
        },
        "date": {
          "type": "string",
          "description": "Flight date YYYY-MM-DD"
        },
        "fare_class": {
          "type": "string",
          "description": "Optional: economy,
business, first"
        }
      },
      "required": [
        "origin",
        "destination",
        "date"
      ]
    }
  },
  {
    "name": "search_houses",
    "description": "Search for available houses in
a zip code",
    "parameters": {
      "type": "object",
      "properties": {
        "zipcode": {
          "type": "integer",
          "description": "ZIP code to search"
        }
      },
      "required": [
        "zipcode"
      ]
    }
  },
  {
    "name": "ask_user",
    "description": "Ask the user a clarifying
question about their preferences",
    "parameters": {
      "type": "object",
      "properties": {
        "question": {
          "type": "string",
          "description": "The question to ask"
        }
      },
      "required": [
        "question"
      ]
    }
  },
  {
    "name": "get_item_details",
    "description": "Get full details for a specific
item by its ID",
    "parameters": {
      "type": "object",
      "properties": {
        "item_id": {
          "type": "string",
```

```
          "description": "Item ID"
        }
      },
      "required": [
        "item_id"
      ]
    }
  },
  {
    "name": "compute_voi",
    "description": "Compute Value of Information
scores for remaining hidden preferences",
    "parameters": {
      "type": "object",
      "properties": {
        "known_preferences": {
          "type": "object",
          "description": "Already-elicited
preferences"
        }
      }
    }
  },
  {
    "name": "select_item",
    "description": "Commit to recommending this
item to the user",
    "parameters": {
      "type": "object",
      "properties": {
        "item_id": {
          "type": "string",
          "description": "Item ID to recommend"
        },
        "reason": {
          "type": "string",
          "description": "Why this item"
        }
      },
      "required": [
        "item_id"
      ]
    }
  },
  {
    "name": "submit_booking",
    "description": "Submit the booking to complete
the transaction and receive scores. Only item_id is
required \u2014 user profile is auto-filled.",
    "parameters": {
      "type": "object",
      "properties": {
        "item_id": {
          "type": "string",
          "description": "Selected item ID"
        }
      },
      "required": [
        "item_id"
      ]
    }
  }
]
```

`compute_voi` output. The tool recomputes the greedy-sequential schedule with the agent's `known_preferences` treated as stated, and returns one record per remaining slot with the runtime patience cost at the next question position; `action` is `ask` when $\text{voi} \cdot \lambda_{\text{pat}} - c_{\text{pat}} > 0$ ($\lambda_{\text{pat}} = 0.84$, $\alpha = 0.05$, $\beta = 1.5$). For `flight_0081_partial` with no known preferences it returns:

```
{
  "preferred_airline": {
    "voi": 1.0,
    "c_pat": 0.05,
    "net": 0.79,
    "action": "ask",
    "type": "hidden"
  },
  "budget_cap": {
    "voi": 0.0,
    "c_pat": 0.075,
    "net": -0.075,
    "action": "skip",
    "type": "hidden"
  },
  "acceptable_stops": {
    "voi": 0.0,
    "c_pat": 0.1125,
    "net": -0.1125,
    "action": "skip",
    "type": "hidden"
  },
  "time_preference": {
    "voi": 0.0,
    "c_pat": 0.1688,
    "net": -0.1688,
    "action": "skip",
    "type": "hidden"
  }
}
```

L.5 Goal-Text Generation Prompts

Goal text is written by Gemini 2.5 Flash (temperature 0.7) from the stated preferences only; a persona is chosen per scenario from a fixed list to vary tone, and a validator re-checks that every stated value appears in the text. System prompt:

```
You are rewriting a structured booking request into
natural, casual language that a real person would
type into a chat agent.

Rules:
- Include ALL stated preferences naturally in the
text -- do not drop any
- Do NOT add preferences that aren't in the input
- Vary phrasing -- don't always use "I prefer" or
"My budget is"
- Use natural city names alongside airport codes
where helpful (e.g., "Chicago O'Hare" for ORD)
- Keep it to 1-3 sentences
- Match the requested tone
- Don't mention "SERP", "scenario", "golden", or
any technical terms
- For housing: use natural descriptions ("3-bed",
"at least 1,800 sq ft")
- For flights: dates should be natural ("April
22nd" not "2022-04-22")
```

Personas (`tone` field): casual and friendly; direct and efficient; polite and detailed; informal and budget-conscious; professional and concise. User message for `flight_0081_partial` (persona *casual and friendly*):

```
Tone: casual and friendly
Domain: Flight booking
Route: ORD -> ATL
Date: 2022-04-29
Stated preferences (MUST include all):
  - Budget: $270

Rewrite as a natural user message:
```

User message for `housing_0002_partial` (persona *direct and efficient*):

```
Tone: direct and efficient
Domain: House buying
Location: zip code 98038
Stated preferences (MUST include all):
  - Budget: $371,720
  - Minimum bedrooms: 3

Rewrite as a natural user message:
```

M Example Conversations

We show two conversations on the same scenario from Figure 1 (ORD→SFO, Sep 10; hidden: time=*afternoon*, stops=*nonstop*). Price gaps: time = $180 (cheapest afternoon $374 − cheapest morning $194); stops = $24 unconditional (cheapest nonstop $218 − cheapest 1-stop $194, as in Figure 1), narrowing to $15 ($389 − $374) once the afternoon window is fixed, where the nonstop is the cheaper option.

Example A: Base agent (failure). The Base agent asks a linguistically natural but economically low-value question first.

Turn	Content
User goal	"Book me a flight from Chicago to San Francisco on Sep 10."
Agent	Searches flights. Finds 4 options ($194–$389).
Agent	"Do you have a preference for nonstop vs. connecting flights?" *[Asks stops first – price gap only $24]*
User	"I prefer nonstop."
Agent	Recommends **#3** (Delta, nonstop, 6:50a, $218) – cheapest nonstop.
Result	**TSR = 0**. Item #3 is a morning flight; user wanted afternoon. The agent never asked about timing (price gap $180), so it missed the decisive preference.

Example B: VoI-Instructed agent (success). The VoI-Instructed agent reasons about price spreads and asks the highest-gap attribute first.

Turn	Content
User goal	"Book me a flight from Chicago to San Francisco on Sep 10."
Agent	Searches flights. Observes: afternoon flights start at $374, morning at $194 ($180 gap); nonstop vs. 1-stop differ by only $24 ($15 within the afternoon window).
Agent	"Do you prefer a morning or afternoon departure?" *[Asks timing first – price gap $180]*
User	"Afternoon, please."
Agent	Recommends **#1** (United, nonstop, 2:15p, $374) – cheapest afternoon flight.
Result	**TSR = 1**. The golden item.

Both agents asked one question each. Both questions are linguistically reasonable clarifications. The difference is purely economic: the VoI-Instructed agent identified that timing, not stops, is the attribute that changes which item is optimal on this SERP.

N Scenario Quality Judge Prompt

The following system prompt is used verbatim for the LLM-as-judge quality filter described in §3, generated directly from the source file. The judge receives a scenario summary containing the goal text, stated and hidden preferences, golden and default items with prices and attributes, per-question QV scores, and the first five SERP items; an example of that user message follows the prompt. The `must-ask`/`nice-to-ask` tags in that message partition the hidden slots at generation time by the generator's own informativeness ranking; they play no role in scoring, and the optimal strategy listed at the end of the message follows QV alone.

```
You are an expert evaluator for a preference
elicitation benchmark called ElicitBench.

You are reviewing a booking scenario that tests
whether an AI agent can discover a user's hidden
preferences by asking the right questions.

Evaluate the scenario on these criteria:

1. **Realism** (0-10): Does this feel like a real
booking situation a person would encounter?
   - High: Natural preferences, realistic price
ranges, plausible SERP
   - Low: Contrived preferences, unrealistic
prices, identical options

2. **Elicitation Value** (0-10): Is there
meaningful value in asking questions?
   - High: Hidden preferences significantly change
which item the agent should pick
   - Low: The golden item is obvious even without
asking (e.g., it's the cheapest)

3. **Difficulty** (0-10): How hard is it for an
agent to identify the right questions?
   - High: Multiple plausible items, subtle
preference differences
   - Low: Only one item matches, obvious which
question to ask

4. **Diversity** (0-10): Does the SERP offer
meaningfully different trade-offs?
   - High: Varied prices, airlines, times, creating
genuine choice tension
   - Low: All items are nearly identical

Respond ONLY with valid JSON (no markdown, no
explanation outside JSON):
{
  "realism": <int 0-10>,
  "elicitation_value": <int 0-10>,
  "difficulty": <int 0-10>,
  "diversity": <int 0-10>,
  "overall_score": <float 0-10>,
  "explanation": "<2-3 sentences explaining why
this scenario is interesting or problematic>",
  "recommendation": "<keep|marginal|drop>"
}

Recommendation guidelines:
- "keep": overall_score >= 6, scenario is
interesting and well-constructed
- "marginal": overall_score 4-6, scenario has some
value but issues
- "drop": overall_score < 4, scenario is trivial,
unrealistic, or broken
```

The judge runs on Gemini 2.5 Flash at temperature 0.7. Its input user message for `flight_0081_partial` is:

```
## Scenario: flight_0081_partial
```

```
Domain: travel
Split level: partial

### User Goal
"Hello! I'm hoping to find a flight from Chicago
O'Hare (ORD) to Atlanta (ATL) on April 29th.
Ideally, I'd like to keep the cost around $270."

### Preferences
Stated (agent knows): budget_cap=270
Hidden must-ask: acceptable_stops=0,
preferred_airline=Delta
Hidden nice-to-ask: time_preference=morning

### Golden Item (what user actually wants)
  Price: $218.60
  Attributes: {"airline": "Delta", "airline_code":
"DL", "num_stops": 0, "departure_time": "06:00",
"arrival_time": "08:53", "departure_hour": 6,
"duration": "PT1H53M", "cabin": "coach",
"fare_class": "economy"}

### Default Item (what uninformed agent would pick)
  Price: $140.60
  Attributes: {"airline": "United", "airline_code":
"UA", "num_stops": 0, "departure_time": "21:11",
"arrival_time": "00:15", "departure_hour": 21,
"duration": "PT2H4M", "cabin": "coach",
"fare_class": "economy"}

### Benefit Model
  V(golden) = 0.9768
  V(default) = 0.7551
  Value gap = 0.2217

### Value of Information (VoI per question)
  - preferred_airline: VoI=1.00 (must_ask)
  - acceptable_stops: VoI=0.00 (must_ask)
  - time_preference: VoI=0.00 (nice_to_ask)

### SERP Summary
  Total items: 20
  Price range: $141 - $452
  First 5 items:
    #0: $141, {"airline": "United", "num_stops": 0,
"departure_time": "21:11"}
    #1: $180, {"airline": "Frontier Airlines",
"num_stops": 1, "departure_time": "12:32"}
    #2: $191, {"airline": "American Airlines",
"num_stops": 0, "departure_time": "17:10"}
    #3: $192, {"airline": "United", "num_stops": 0,
"departure_time": "13:55"}
    #4: $198, {"airline": "American Airlines",
"num_stops": 1, "departure_time": "16:15"}

### Optimal Strategy
  Optimal questions to ask: 1
    preferred_airline: VoI=1.00, action=ask
    acceptable_stops: VoI=0.00, action=stop
    time_preference: VoI=0.00, action=stop
```