Beyond Static Benchmarks: An Active Testing Paradigm for Large Language Models

Anonymous authors
Paper under double-blind review

Abstract

Static benchmarks remain the dominant way to evaluate large language models, yet fixed questions are vulnerable to saturation, contamination, and overfitting. More fundamentally, their test distributions do not adapt to the model under evaluation, potentially spending evaluation budget on regions that are already well understood or largely solved. We propose **Active Testing**, an evaluation paradigm in which observed model behavior dynamically guides where to test next, with emphasis on regions that are both failure-prone and uncertain. To demonstrate its effectiveness, we instantiate this paradigm for web-search agents by turning Wikidata into a large, structured, and verifiable task space with 12 controllable dimensions. We realize the active tester with a random-forest surrogate, enabling sample-efficient search and targeted discovery of weak regions in a large discrete task space. Across six agents, accuracy drops by an average of 28.0 percentage points from the initial random baseline to the final attack round. Rankings by *Active-Testing AUC* also diverge from initial-accuracy rankings, revealing performance differences that the shared starting distribution obscures. The tester converges to different regions for different models, exposing model-specific weaknesses while also uncovering shared failure patterns across agents. These results motivate a shift from static benchmarks toward active testing protocols, offering a more proactive and diagnostically informative approach to evaluating frontier models.

1 Introduction

Static benchmarks with fixed questions remain the standard tool for evaluating large language models (LLMs) (Wei et al., 2025; Mialon et al., 2024; Shi et al., 2023). However, as models rapidly evolve (OpenAI, 2025; DeepSeek-AI et al., 2024; Kimi Team et al., 2025; GLM-4.5 Team, 2025; Anthropic, 2025), this static evaluation paradigm is reaching a fundamental bottleneck. Static test sets are prone to rapid saturation, benchmark contamination, and subtle overfitting (Deng et al., 2024a; Jain et al., 2025; Akhtar et al., 2026a). More critically, static evaluation is inherently *passive*: it evaluates all models against an identical, unyielding distribution of questions, failing to adapt to the model-specific capability boundaries and long-tail vulnerabilities of individual target models. Consequently, static scores often yield an incomplete and potentially deceptive picture of true model capability.

To go beyond static datasets, recent evaluation efforts have shifted toward live benchmark refreshing (White et al., 2025; Jain et al., 2025; Zhou et al., 2025). However, these approaches remain static in nature: simply releasing new fixed questions merely delays benchmark saturation without fundamental changes. Another emerging direction explores dynamic task generation, which endlessly synthesizes diverse tasks (Zhu et al., 2024; Zhang et al., 2024; Zheng et al., 2025; Kiela et al., 2021; Zheng et al., 2026). Yet, it only expands *what* can be tested, failing to intelligently determine *where* to focus testing within that vast space. Because model capability is non-uniform across a task space, uniform sampling risks repeatedly probing already well-solved regions while missing narrow or unknown capability boundaries. Instead, observed model behavior provides a highly informative, natural feedback signal to direct testing toward uncertain or failure-prone regions, motivating a closed-loop, active evaluation framework that dynamically guides subsequent tests.

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

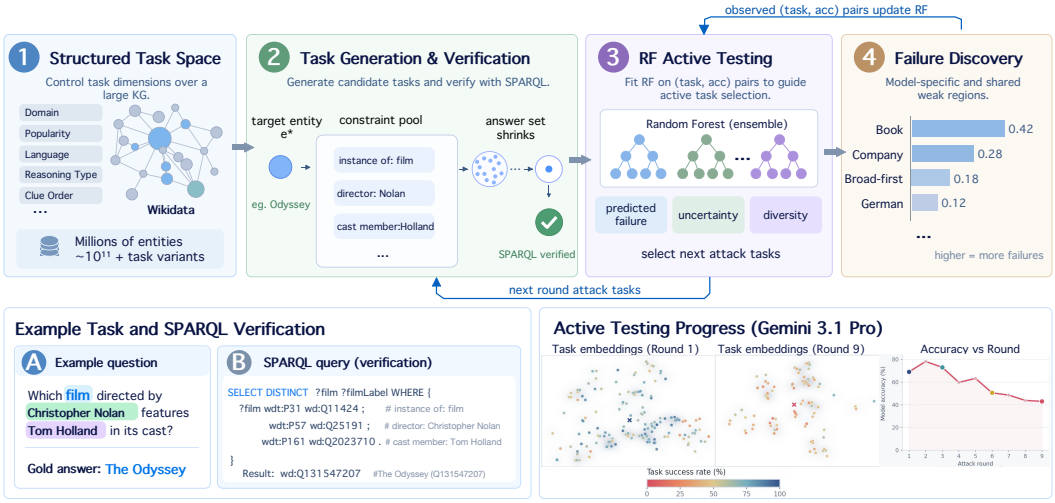


Figure 1: **Overview of our active-testing framework. Top:** (1) Factorize Wikidata into a structured, controllable task space; (2) instantiate candidate tasks via inverse construction and verify them with SPARQL queries; (3) fit a Random Forest (RF) surrogate on observed (TaskConfig, accuracy) pairs to select adversarial tasks balancing predicted failure, uncertainty, and diversity; and (4) discover model-specific and shared failure regions. **Bottom left:** An example generated task with its verified SPARQL query. **Bottom right:** t-SNE visualization of task embeddings (Rounds 1 vs. 9 on Gemini 3.1 Pro), illustrating diverse attack selection and declining model accuracy across testing rounds.

In this paper, we propose a shift from fixed benchmark evaluation to **Active Testing**: instead of sampling all questions in advance, an active tester observes model performance, estimates where weak regions may lie, and selects subsequent tasks accordingly. This requires a task environment that is 1) *large* enough to prevent exhaustive sampling, 2) *structured* so observations can guide task selection, and 3) *verifiable* to ensure task quality. We instantiate these properties on Wikidata (Vrandečić & Krötzsch, 2014), an evolving graph of over 1.2×10^8 items (Wikidata contributors, 2026). As shown in Figure 1, we factor tasks into a 12-dimensional controllable space spanning knowledge scope, constraint structure, reasoning topology, and presentation. Through inverse construction, constraints uniquely identify a target entity and are verified with SPARQL, preserving the open-web search challenge of benchmarks like BrowseComp (Wei et al., 2025) while making the space directly navigable. Our tester fits a random-forest surrogate on observed *(TaskConfig, accuracy)* pairs to select subsequent adversarial tasks.

Our experiments reveal three benefits of active testing. **1) Exposing capability differences beyond fixed evaluation.** Across six web-search agents, accuracy drops by an average of 28.0% over nine rounds, while rankings by *Active-Testing AUC* differ markedly from those based on initial accuracy. Notably, GPT-5.6 Sol suffers the steepest accuracy drop among all agents. These results show that adapting the test distribution can expose capability differences that a fixed initial evaluation obscures. **2) Efficiently navigating a large adaptive test space.** Because only a small fraction of the space can be evaluated, deciding where to test next is critical. Our random-forest proposer reaches 23.2% accuracy on GPT-5.4 Mini, outperforming RL and LLM red-teaming alternatives while retaining task-space diversity. This suggests that explicit surrogate modeling provides an effective mechanism for concentrating evaluation on empirically weak regions. **3) Revealing model-dependent evaluation trajectories.** As testing adapts to observed behavior, the selected regions diverge across models—for example, increasingly selecting *Book* tasks for DeepSeek V4 Pro, while *Company* tasks are enriched by up to 6.9× for Gemini 3.1 Pro. Together with search-trace analysis, these trajectories reveal differences in both where models fail and how they search, providing richer diagnostics than a single shared test distribution.

2 Active Testing

In this section, we first decompose large-scale Wikidata task synthesis into two components: a factorized structured task space and a programmatic inverted-generation pipeline. We then introduce Active Testing, which uses observed model behavior to adaptively explore uncertain and failure-prone regions of this space under a limited evaluation budget.

2.1 Structured Task Space

Rather than predefining a small set of task families based on human priors, we factor the underlying knowledge-graph space into 12 low-level configuration parameters. For interpretability, we organize them along four primary axes. `Knowledge scope` controls the target domain and entity familiarity through *domain* and *popularity_range*. `Constraint structure` controls how evidence narrows the candidate set through *min_c*, *max_c*, *neg_c*, *min_filter*, and *selection_ordering*. `Reasoning topology` controls direct identification, property projection, and linked entity isolation through *reasoning_type*, *hops*, and *multi_layer_isolation*. `Surface realization` controls *language* and *presentation_ordering* without changing the underlying graph semantics. For readability, Figure 3 uses human-readable aliases for selected coordinates, such as *Clue Priority*, *Negative Clues*, *Candidate Floor*, and *Positive Clues*. Full definitions and valid ranges are provided in Table A1.

Together, these parameters define a large but explicitly addressable task space. Grounding these controls over Wikidata entities, relation paths, and valid constraint subsets yields an estimated combinatorial capacity of $\gtrsim 10^{11}$ task variants. This number describes the construction space rather than the number of successfully generated questions; each instantiated task must still pass graph-realizability and verification checks. The resulting space is therefore large enough to support broad exploration while retaining explicit coordinates for targeted diagnosis (see Appendix A.1).

2.2 Programmatic Task Generation

Given a task configuration $\mathbf{c} \in \mathcal{C}$, we instantiate a concrete task through an *inverted construction* process: first select a ground-truth answer entity e^*, iteratively construct a constraint set that uniquely isolates e^*, and finally render the verified graph structure into natural language.

1. Ground the Answer. We sample a target entity e^* based on the requested domain $\mathbf{c}.\textit{domain}$ and popularity window $\mathbf{c}.\textit{popularity_range}$. Let $\mathcal{S}_0 = \{e \in \mathcal{E} \mid \text{type}(e) = \mathbf{c}.\textit{domain}\}$ denote all candidate entities within the domain. From the graph neighborhood of e^* in Wikidata $\mathcal{G}$, we retrieve candidate relational clues $\mathcal{H}(e^*) = \{(r, n) \mid (e^*, r, n) \in \mathcal{G}\}$. We filter out relations that mainly serve ontological, navigational, or editorial functions (e.g., type/hierarchy links, maintenance metadata, and semantically ambiguous relations), and retain relations that can be expressed as clear and verifiable natural-language constraints.

2. Isolate the Answer. Starting from the domain candidate universe $\mathcal{S}_0$, each appended clue $h_k = (r_k, n_k)$ imposes a relational constraint filtering out irrelevant entities. Accumulating clues $\mathcal{P}_k = (h_1, \ldots, h_k)$ contracts the candidate set $\mathcal{S}(\mathcal{P}_k)$, inducing a strictly shrinking sequence:

$$\mathcal{S}_0 \supsetneq \mathcal{S}(\mathcal{P}_1) \supseteq \mathcal{S}(\mathcal{P}_2) \supseteq \cdots \supseteq \mathcal{S}(\mathcal{P}_K) = \{e^*\}. \tag{1}$$

To prevent multi-constraint tasks from collapsing into single-clue lookups, we enforce $|\mathcal{S}(\mathcal{P}_k)| > 1$ for all $k < K$ until $|\mathcal{S}(\mathcal{P}_K)| = 1$. Entity isolation serves as the atomic operator across reasoning topologies. A *single-layer* task directly queries the isolated entity (Algorithm 1); a *sub-property* task further queries one of its properties; and a *multi-layer* task composes the same isolation operator over relation-connected entities for the requested number of hops.

3. Verify and Render. We execute the constructed constraints as SPARQL queries and retain only tasks whose unique result is e^* (example in Figure 1). The verified structure is then rendered according to the specified task-presentation configuration. A final automated semantic validator rejects candidate questions that leak hidden entities, omit required constraints, reverse relations, or alter the intended reasoning structure. Natural-language generation therefore changes only its surface form, while task semantics and ground truth remain strictly anchored to the underlying executable graph structure.

2.3 Random Forest for Active Testing

Active testing treats test selection as exploration over the task space. Observed failures indicate potential capability limitations, uncertainty marks regions that remain insufficiently characterized, and diversity prevents the search from repeatedly concentrating on the same region. Together, these signals define where evaluation should proceed next. We therefore formulate test selection as a sequential model-based optimization (SMBO) problem (Hutter et al., 2011) over the discrete coordinate space $\mathcal{C}$, providing a principled and controllable way to balance exploration of uncertain regions with exploitation of empirically observed weaknesses.

We employ a Random Forest (RF) as the non-parametric surrogate model $\hat{f}_t : \mathcal{C} \to [0, 1]$. RF natively accommodates heterogeneous, discrete task coordinates while maintaining high sample efficiency. Crucially, the ensemble variance across trees provides a practical estimate of epistemic uncertainty $\sigma_t(\mathbf{c})$, enabling uncertainty-aware search over the finite space $\mathcal{C}$.

At round t, the candidate configuration is selected by maximizing a structured acquisition function $a_t(\mathbf{c})$:

$$a_t(\mathbf{c}) = \alpha_t \mu_t(\mathbf{c}) + \beta_t \sigma_t(\mathbf{c}) + \gamma_t \mathcal{D}_t(\mathbf{c}), \tag{2}$$

where $\mu_t(\mathbf{c})$, $\sigma_t(\mathbf{c})$, and $\mathcal{D}_t(\mathbf{c})$ quantify predicted failure rate, model uncertainty, and spatial diversity, respectively. We govern the exploration–exploitation transition using a cosine schedule that gradually increases α_t while decaying β_t and γ_t. Early rounds therefore emphasize uncertainty and diversity to encourage broad coverage, while later rounds increasingly concentrate the acquisition policy on verified high-failure regions. As shown in the t-SNE task embeddings across different rounds on Gemini 3.1 Pro (Figure 1), this search strategy achieves broad exploration across task dimensions while progressively concentrating attacks on agent-specific failure regions.

3 Experiments

3.1 Experimental Setup

We evaluate six LLM agents spanning four model families—GPT, Gemini, DeepSeek, and GLM. All agents use the same task-generation protocol and required answer schema, while search is executed through each model's native harness interface. We begin with a shared 300-task random warm start, consisting of 150 configurations with two independently generated questions each, followed by eight adaptive attack rounds. Each round targets 100 configurations, again with two questions per configuration. We cap reasoning depth at three hops so that the finite evaluation budget can probe a broader range of task dimensions rather than being dominated by increasingly deep relation chains. The RF surrogate uses 300 trees with a minimum leaf size of four and is refit after every round. Full evaluation details and implementation settings are provided in Table A3.

3.2 Active Testing Reveals Static Evaluation Blind Spots

3.2.1 Adaptive Testing Changes Model Comparisons

Figure 2 shows how agent performance evolves as testing adapts toward observed failure regions. Across six agents, accuracy drops by an average of 28.0% from the initial random evaluation to the final attack round, with a maximum drop of 39.9 percentage points. The sustained degradation shows that Active Testing progressively redirects evaluation toward

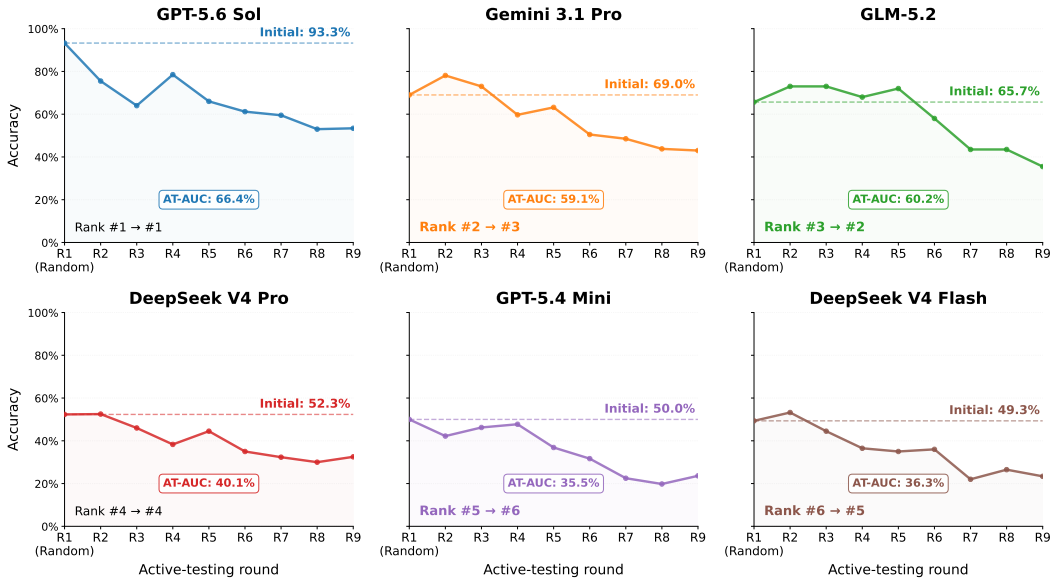


Figure 2: **Performance trajectories diverge under multi-round Active Testing.** R1 contains 300 randomly sampled tasks shared across agents, followed by eight rounds (R2–R9) of model-specific Active Testing. Dashed lines indicate initial random-round accuracy. Each panel reports **Active-Testing AUC (AT-AUC)**, the normalized area under the multi-round accuracy trajectory, together with the change from initial-accuracy rank to AT-AUC rank. Sustained degradation and rank changes show that model comparisons can differ once the test distribution adapts to observed behavior.

empirically weak regions rather than generating a single harder batch. The trajectories also differ substantially across agents, indicating that evaluation outcomes depend not only on the shared initial distribution but also on how testing subsequently adapts.

R1 provides a common starting point across all agents, after which Active Testing selects subsequent tasks according to each model's observed behavior. The realized test sequences therefore differ, while the evaluation protocol remains fixed: all agents share the same initial sample, task space, acquisition rule, generation pipeline, and number of rounds. Active Testing thus standardizes the *evaluation protocol*, rather than the realized test items. To summarize performance over this process, we introduce *Active-Testing AUC* (AT-AUC), the normalized area under the multi-round accuracy trajectory:

$$\text{AT-AUC} = \frac{1}{T-1} \sum_{t=1}^{T-1} \frac{\text{Acc}(t) + \text{Acc}(t+1)}{2}, \tag{3}$$

where $\text{Acc}(t)$ denotes accuracy at round t. AT-AUC captures both initial performance and how well it persists as testing moves through the task space. Because later rounds are selected separately for each model, it should be interpreted as performance persistence under a common active-testing protocol rather than accuracy on a single shared distribution.

This adaptive view changes model comparisons in Figure 2: GLM-5.2 moves from #3 to #2, while GPT-5.4 Mini moves from #5 to #6. GPT-5.6 Sol remains #1 under both rankings, yet falls from 93.3% initially to 53.4% in the final round, showing that a strong relative rank can still conceal substantial within-model degradation.

3.2.2 Active Testing Targets Model-Specific Failure Regions

The model-dependent trajectories above suggest that Active Testing does not simply converge to a universally difficult subset of the task space, but instead adapts toward different failure regions for different target models. To test this more directly, we conduct a cross-model replay between GPT-5.4 Mini and DeepSeek V4 Flash, whose initial performance

and AT-AUC are closely matched. We collect verified tasks from the final two DeepSeek rounds containing five of its weak marginals and replay the identical questions on GPT-5.4 Mini (Table 1). GPT-5.4 Mini achieves only **26.0%** macro-averaged accuracy on the DeepSeek-selected regions, confirming that some difficult configurations transfer across models. However, transfer alone does not explain where Active Testing converges. Among the five DeepSeek-selected weak marginals, only two—*French* and *Human*—also appear among GPT-5.4 Mini's own top-5 weak marginals (see Figure A1).).

More importantly, even when the marginal label overlaps, GPT-5.4 Mini can be more vulnerable in the region discovered by its own testing trajectory. On *Human* tasks, GPT-5.4 Mini obtains **28.1%** accuracy on DeepSeek-selected examples but only **20.5%** on examples reached by its own Active Testing process. The same pattern is weaker but consistent for *French*. This shows that Active Testing is not merely replaying broadly difficult task types; it further concentrates evaluation on regions that are particularly weak for the model under test.

Table 1: **Active Testing localizes model-specific failure regions.** DeepSeek-selected late-round tasks remain difficult for GPT-5.4 Mini, indicating shared difficulty, yet GPT-5.4 Mini performs worse in overlapping regions reached by its own Active Testing trajectory.

Region	DeepSeek	GPT replay	Top-5 overlap	GPT targeted
Ship	10.3%	20.5%	No	–
French	21.3%	24.0%	**Yes**	**23.6%**
Niche	24.4%	31.0%	No	–
Chinese	26.2%	26.4%	No	–
Human	34.4%	28.1%	**Yes**	**20.5%**

These results support the model-specific nature of the adaptive testing process. Human-interpretable marginals provide useful summaries, but Active Testing operates over complete TaskConfigs. Consequently, two models may share a difficult marginal while the joint configurations emphasized by their respective testing trajectories remain different. We next examine the complementary question of which failure patterns are shared across agents in Sec. 3.3.

3.3 Shared Failure Patterns Across Agents

To determine whether different models exhibit common vulnerabilities under combined task dimensions, we analyze agent performance across joint configuration cells (a, b). For each model m, relative performance is defined as $\Delta_m(a, b) = \mathrm{Acc}_m(a, b) - \mathrm{Acc}_m(\text{overall})$, where negative values indicate degradation relative to the model's overall baseline. To isolate robust failure patterns, we retain only joint configurations that satisfy two criteria: (1) each model evaluates at least 30 tasks per cell, and (2) all six models exhibit consistent degradation, i.e., $\Delta_m(a, b) < 0$. Figure 3 presents the six interaction pairs with the largest macro-averaged performance drops.

Agents struggle with weakly discriminative clues. Across the shared performance degradation regions highlighted in Figure 3, `Broad`-first selection is consistently associated with lower relative accuracy across evaluated architectures. Although the complete clue set uniquely identifies the target entity in principle, individual broad clues possess low discriminative power on their own and consequently leave a large number of plausible candidate entities. Under Broad-first selection, the accuracy shift is −9.2 pp when no negative clues are provided, but recovers to +2.6 pp with the inclusion of one negative clue (Figure 3b), suggesting that explicitly exclusionary evidence can effectively assist agents in narrowing down otherwise ambiguous candidate sets. Furthermore, multi-layer cells also demonstrate higher relative accuracy under Broad-first selection regimes. One plausible explanation is that additional structural layers supply independent clues for linked entities, thereby creating valuable extra anchors for the agent (see Section A.3).

More search does not always help. A common assumption in agentic evaluation is that agents fail primarily because they terminate their search process too early. However, the empirical *Q (C/I)* statistics detailed in Table 2 reveal the exact opposite trend: failed task trajectories often trigger substantially more queries than successful ones (e.g., 17.8

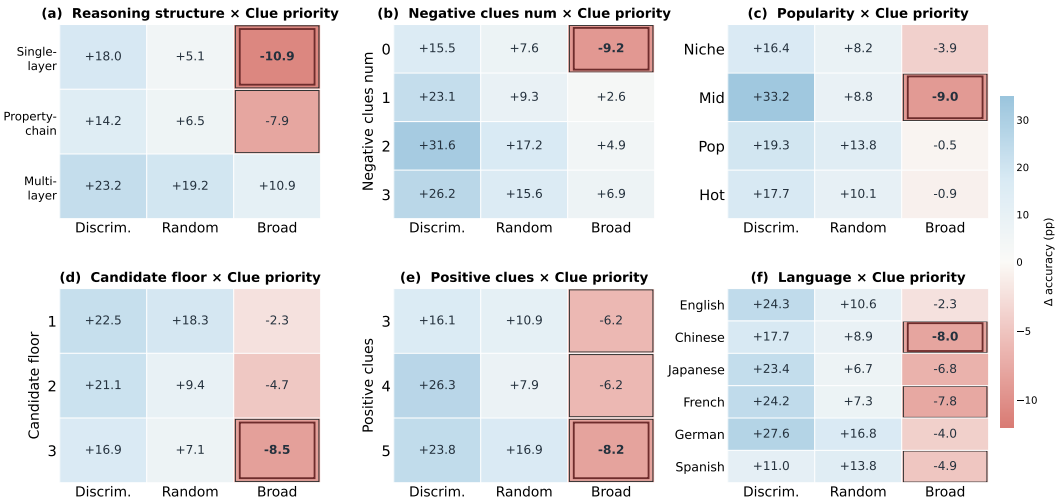


Figure 3: **Shared failure patterns across different models.** Heatmaps report macro-averaged accuracy shifts $\Delta(a, b) = \mathrm{Acc}(a, b) - \mathrm{Acc}_{\mathrm{overall}}$ across six agents. Red cells ($\Delta < 0$) indicate shared degradation. Across multiple interactions, *Broad*-first clue selection repeatedly produces the largest performance drops. Parameters include: (1) **`Clue Priority`** (*Discrim.*: more discriminative; *Broad*: less discriminative); (2) **`Reasoning structure`** (*Single-layer*, *Property-chain*, *Multi-layer*); (3) **`Negative clues`** (number of exclusions); (4) **`Popularity`** (Wikipedia sitelinks: *Niche* 1–3, *Mid* 4–15, *Pop* 16–50, *Hot* 51–350); (5) **`Candidate floor`** (minimum candidate set before final isolation); (6) **`Positive clues`** (number of positive constraints); and (7) **`Language`** (six languages).

vs. 8.9 queries for Gemini 3.1 Pro). Yet, this additional search effort frequently fails to resolve the underlying task ambiguity, as agents continuously reformulate overlapping clues or repeatedly revisit already plausible candidates without decisively narrowing the overall hypothesis space. Thus, the primary bottleneck appears to lie less in the raw search budget itself than in how effectively and strategically agents can eliminate weak candidates and accumulate truly discriminative evidence.

Agents differ in how they search. In practice, web agents exhibit remarkably distinct search behaviors and query patterns rather than uniform tool usage. As quantitatively illustrated in Table 2, we observe three recurring operational strategies across models: (1) *Targeted Search*, which systematically narrows plausible candidates early in the process and then issues specific or domain-restricted queries for final verification; (2) *Relation-chain Search*, which incrementally follows complex entity relationships using diverse query formulations; and (3) *Constraint Aggregation*, which repeatedly packs numerous constraints simultaneously into structurally similar search queries. For instance, GPT-5.6 Sol requires an average of only 3.6 queries per task, whereas DeepSeek V4 Pro utilizes 17.7 queries. A complete search trace analysis on an independently generated task is in Appendix D.

4 Efficacy of Surrogate-Based Active Testing

Baselines. We compare our RF acquisition with two adaptive tester families: *Reinforcement Learning (RL)* and *LLM Red-Teamers*. For RL, *RL (History-Unconditioned)* autoregressively proposes *TaskConfigs* without explicit failure memory, whereas *RL (History-Conditioned)* incorporates failure summaries and per-dimension error statistics. For red-teaming, *Red-Team (Direct Config)* directly proposes complete discrete configurations, while *Red-Team (Subspace Guidance)* first identifies vulnerable subspaces before sampling valid configurations. Full baseline prompts, hyperparameters, and implementation details are provided in Appendix C.2 and Appendix E.

Table 2: **Case Study: Search Strategies Across Web Agents.** We report mean query count (**Avg. Q**), average queries on correct vs. incorrect tasks (**Q (C/I)**), and character 3–5 gram TF–IDF query diversity (D_Q). Representative queries are drawn from the same case question: *"There is a river located in Barry County and Allegan County, belonging to the Kalamazoo River watershed, originating from a lake named 'Gun', and eventually flowing into the Kalamazoo River. What country does this river belong to?"*

Model[a]	Avg. Q	Q (C/I)	D_Q	Strategy	Representative Query
GPT-5.6	3.6	3.1 / 4.7	0.67	Targeted	*(Candidate → Specific)*
GPT-5.4	4.3	3.6 / 4.7	0.58		`site:michigan.gov "Gun River" Barry Allegan`
Gemini	12.5	8.9 / 17.8	**0.71**	Relation-chain	*(Entity → Relation → Entity)*
GLM	11.2	8.3 / 15.4	0.69		1. `river outflow from "Gun Lake" "Kalamazoo"` 2. `"Gun River" "Kalamazoo River" Barry County`
DS-Pro	17.7	12.2 / 21.5	0.61	Constraint agg.	*(Pack Constraints → Refine)*
DS-Flash	3.7	3.0 / 4.1	**0.49**		`Gun River Kalamazoo watershed Barry Allegan Gun Lake tributary Michigan`

[a] Model abbreviations: GPT-5.6 = GPT-5.6 Sol; GPT-5.4 = GPT-5.4 Mini; Gemini = Gemini 3.1 Pro; GLM = GLM-5.2; DS-Pro = DeepSeek V4 Pro; DS-Flash = DeepSeek V4 Flash.

Results. Table 3 highlights the necessity of explicit failure modeling and structured search. Without historical failure feedback, RL performance remains near the random baseline; conditioning on past failure patterns reduces accuracy to 28.9%, albeit with a drop in diversity. Similarly, directly generating complete task configurations via an LLM yields only a modest attack accuracy (48.5%), whereas restricting it with subspace guidance reduces both accuracy and diversity. These findings demonstrate that adaptive testers critically rely on explicit structural guidance to navigate large discrete task spaces efficiently.

Table 3: **Baseline comparison on GPT-5.4 Mini.** Lower accuracy (↓) indicates a stronger attack; higher diversity (↑) indicates broader coverage.

Method	Acc. ↓	Diversity ↑
Random warm-up	50.0%	**0.186**
RL (History-Unconditioned)	51.3%	0.183
RL (History-Conditioned)	28.9%	0.100
Red-Team (Direct Config)	48.5%	0.165
Red-Team (Subspace Guidance)	35.0%	0.133
RF (Surrogate Acquisition)	**23.2%**	0.129

Our RF acquisition achieves the strongest attack, reaching 23.2% accuracy while retaining non-trivial diversity. The surrogate models non-linear interactions among task coordinates, estimates uncertainty from tree disagreement, and cheaply scores large candidate pools before execution. This allows Active Testing to progressively concentrate evaluation on empirically verified failure regions while maintaining broader spatial coverage than purely mode-seeking alternatives.

5 Related Work

5.1 From Static Benchmarks to Dynamic Task Generation

Static benchmarks have long served as the primary paradigm for evaluating Large Language Models (LLMs) Hendrycks et al. (2020); Srivastava et al. (2022); Liang et al. (2022). However, they increasingly suffer from severe benchmark contamination and rapid performance saturation (Sainz et al., 2023; Deng et al., 2024b; Akhtar et al., 2026b). To mitigate contamination, recent efforts have introduced live benchmarks with periodic dataset updates (White et al., 2025; Jain et al., 2025); yet these benchmarks remain largely passive, as newly introduced questions are still selected independently of the behavior of the model under evaluation. A complementary line of work moves beyond dataset refreshment toward dynamic task generation, automatically synthesizing new evaluation instances from programmatic structures, reasoning graphs, or controllable transformations (Zhu et al., 2024; Zhang et al., 2024; Zheng et al., 2026; Mirzadeh et al., 2025; Kiela et al., 2021). While such approaches substantially improve benchmark renewability, their generation spaces are typically defined in advance by a limited set of task operators or difficulty dimensions. We

instead parameterize the large and continuously evolving Wikidata graph along 12 fine-grained dimensions, yielding an addressable task space that supports not only dynamic generation, but also subsequent model-driven exploration.

5.2 Open-Web Search and Verifiable Agent Evaluation

The need for such a task space is particularly pronounced for autonomous web-search agents, whose evaluation increasingly involves multi-source synthesis, complex retrieval, and multi-constraint reasoning (Wei et al., 2025; Krishna et al., 2025). Current agent evaluation, however, faces a tension between environment realism and scalable verification. On one hand, evaluation in unrestricted open-web environments is difficult to verify reliably because web content changes over time, relevant evidence can be distributed across sources, and ground-truth answers may be ambiguous (Wei et al., 2025; Mialon et al., 2024). On the other hand, knowledge-graph-based benchmarks provide executable structure and reliable answers (Dammu et al., 2025; Zhao et al., 2026), but often characterize task difficulty through relatively coarse structural factors such as path length or graph topology. We bridge these directions by formulating Wikidata-based entity identification as a fine-grained, controllable task space. Tasks are grounded in executable SPARQL semantics, while agents must solve the resulting natural-language questions through open-web search. This provides both reliable verification and interpretable coordinates for failure analysis, making the task space suitable for adaptive test selection.

5.3 Weakness-Driven Adaptive Evaluation

Given a large and verifiable task space, the remaining question is where to test under a limited evaluation budget (Hutter et al., 2011). Adaptive testing methods, such as Item Response Theory (IRT), select subsequent evaluation items according to observed responses; however, they typically assume a pre-calibrated item bank—a fixed set of questions whose individual difficulty levels and discrimination parameters have already been accurately estimated through prior large-scale human or model testing (Lalor et al., 2016; Rodriguez et al., 2021; Polo et al., 2024). In safety and alignment research, adversarial evaluation and red-teaming search for failure-inducing inputs to discover vulnerabilities, jailbreaks, and unsafe behaviors in large language models (Perez et al., 2022; Hong et al., 2024; Samvelyan et al., 2024), though these methods often operate in unstructured input spaces or rely heavily on specific attacker priors. Prior active model evaluation similarly selects informative examples from a fixed test pool to efficiently estimate performance on the original distribution (Nguyen et al., 2018; Kossen et al., 2021; Lee et al., 2024; Li et al., 2025). Our objective differs: rather than correcting adaptive sampling back to a fixed target distribution, which may itself become contaminated, our goal is to intentionally evolve the test distribution toward empirically observed failure regions.

6 Conclusion

In this work, we introduce **Active Testing**, an adaptive evaluation paradigm in which the test distribution evolves with the model under evaluation. We instantiate it on a large, structured, and verifiable Wikidata task space, using a random-forest tester to sequentially select adversarial tasks from observed model performance. Across six web-search agents, Active Testing substantially lowers measured accuracy and changes model rankings relative to the initial test distribution, showing that fixed benchmarks can obscure meaningful capability differences. The tester also converges to distinct model-specific failure regions while revealing shared failure patterns, and search traces expose differences in how agents allocate search effort and prioritize evidence. Compared with RL and LLM-based proposers, the surrogate-based tester explores the task space more sample-efficiently under limited budgets. Extending Active Testing beyond verifiable knowledge-intensive reasoning to valuable but harder-to-verify domains remains an important direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools for natural-language task rendering, question validation, answer judging, and manuscript text editing and polishing. We have not used generative AI tools for theoretical formulation, core algorithm design (e.g., the Random Forest surrogate acquisition policy), or empirical analysis, and other tasks with required disclosure are not applicable to this work. Additionally, we used generative AI tools for LLM-based red-teaming baseline proposals. We have reviewed all AI-assisted work. Specifically, all LLM-generated task prose, validation outputs, and prompt templates were manually inspected and verified by the authors to ensure factual correctness and strict adherence to Wikidata graph semantics. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Ethics statement

This study adheres to the ICLR Code of Ethics. Our evaluation framework is constructed entirely using open, publicly available knowledge graph entities from Wikidata and open-web sources. No human subjects, private personal data, or crowdsourced human annotators were involved in our experiments or dataset construction. All generated tasks are grounded in objective factual knowledge (e.g., films, literature, and public entities). Although our active testing framework systematically discovers adversarial task configurations to uncover model weak spots, these failure analyses are conducted strictly to expose static evaluation blind spots, diagnose capability limits, and foster more reliable, persistent agent benchmark design. We foresee no direct negative ethical impacts, safety risks, or privacy concerns from this work.

Reproducibility statement

To ensure the full reproducibility of our methodology and empirical findings, we provide complete technical specifications and implementation details throughout the main text and appendix. The 12-dimensional task space, parameter bounds, and conditional topology rules are specified in Section 2.1 and Appendix A.1 (Table A1), while Section 2.2, Algorithm 1, and Appendix A.4 detail the programmatic inverted-construction pipeline and executable SPARQL verification templates. Complete settings for our active testing framework—including the structured acquisition function, cosine schedule, and Random Forest surrogate hyperparameters—are described in Section 2.3 and Appendix C.3 (Table A5). Complete source code, task space specifications, and trajectory logs are included in the supplementary materials and will be made publicly available upon publication.

References

Mubashara Akhtar, Anka Reuel, Prajna Soni, Sanchit Ahuja, Pawan Sasanka Ammanamanchi, Ruchit Rawal, Vilém Zouhar, Srishti Yadav, Chenxi Whitehouse, Dayeon Ki, Jennifer Mickel, Leshem Choshen, Marek Šuppa, Jan Batzner, Jenny Chim, Jeba Sania, Yanan Long, Hossein A. Rahmani, Christina Knight, Yiyang Nan, Jyoutir Raj, Yu Fan, Shubham Singh, Subramanyam Sahoo, Eliya Habba, Usman Gohar, Siddhesh Pawar, Robert Scholz, Arjun Subramonian, Jingwei Ni, Mykel Kochenderfer, Sanmi Koyejo, Mrinmaya Sachan, Stella Biderman, Zeerak Talat, Avijit Ghosh, and Irene Solaiman. When AI benchmarks plateau: A systematic study of benchmark saturation. *arXiv preprint arXiv:2602.16763*, 2026a. URL `https://arxiv.org/abs/2602.16763`.

Mubashara Akhtar, Anka Reuel, Prajna Soni, Sanchit Ahuja, Pawan Sasanka Ammanamanchi, Ruchit Rawal, Vilém Zouhar, Srishti Yadav, Chenxi Whitehouse, Dayeon Ki, et al. When ai benchmarks plateau: A systematic study of benchmark saturation. *arXiv preprint arXiv:2602.16763*, 2026b.

Anthropic. Claude 4 system card. Technical Report, may 2025. URL `https://www.anthropic.com/`.

Preetam Prabhu Srikar Dammu, Himanshu Naidu, and Chirag Shah. Dynamic-kgqa: A scalable framework for generating adaptive question answering datasets. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3498–3508, 2025.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, et al. DeepSeek-V3 Technical Report. *arXiv preprint arXiv:2412.19437*, 2024. URL https://arxiv.org/abs/2412.19437.

Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. Investigating data contamination in modern benchmarks for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8706–8719. Association for Computational Linguistics, 2024a. doi: 10.18653/v1/2024.naacl-long.482. URL https://aclanthology.org/2024.naacl-long.482/.

Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. Investigating data contamination in modern benchmarks for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8706–8719, 2024b.

GLM-4.5 Team. GLM-4.5: Agentic, reasoning, and coding (ARC) foundation models. *arXiv preprint arXiv:2508.06471*, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Zhang-Wei Hong, Idan Shenfeld, Tsun-Hsuan Wang, Yung-Sung Chuang, Aldo Pareja, James R. Glass, Akash Srivastava, and Pulkit Agrawal. Curiosity-driven red-teaming for large language models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4KqkizXgXU.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pp. 507–523. Springer, 2011. doi: 10.1007/978-3-642-25566-3_40.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/94074dd5a072d28ff75a76dabed43767-Abstract-Conference.html.

Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. Dynabench: Rethinking benchmarking in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4110–4124. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.324. URL https://aclanthology.org/2021.naacl-main.324/.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, et al. Kimi K2: Open Agentic Intelligence. *arXiv preprint arXiv:2507.20534*, 2025. URL https://arxiv.org/abs/2507.20534.

Jannik Kossen, Sebastian Farquhar, Yarin Gal, and Tom Rainforth. Active testing: Sample-efficient model evaluation. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5753–5763. PMLR, 2021.

Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqui. Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4745–4759, 2025.

John P Lalor, Hao Wu, and Hong Yu. Building an evaluation scale using item response theory. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 648–657, 2016.

JooChul Lee, Likhitha Kolla, and Jinbo Chen. Towards optimal model evaluation: Enhancing active testing with actively improved estimators. *Scientific Reports*, 14:10690, 2024. doi: 10.1038/s41598-024-58633-3.

Yang Li, Jie Ma, Miguel Ballesteros, Yassine Benajiba, and Graham Horwood. Active evaluation acquisition for efficient LLM benchmarking. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 35581–35602. PMLR, 2025.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *International Conference on Learning Representations*, volume 2024, pp. 9025–9049, 2024.

Seyed Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-Symbolic: Understanding the limitations of mathematical reasoning in large language models. In *International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=AjXkRZIvjB`.

Phuc Nguyen, Deva Ramanan, and Charless Fowlkes. Active testing: An efficient and robust framework for estimating accuracy. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3759–3768. PMLR, 2018.

OpenAI. GPT-5 System Card, August 2025. URL `https://openai.com/index/gpt-5-system-card/`.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pp. 3419–3448, 2022.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. tinybenchmarks: evaluating llms with fewer examples. *arXiv preprint arXiv:2402.14992*, 2024.

Pedro Rodriguez, Joe Barrow, Alexander Miserlis Hoyle, John P Lalor, Robin Jia, and Jordan Lee Boyd-Graber. Evaluation examples are not equally informative: How should that change nlp leaderboards? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4486–4503, 2021.

Oscar Sainz, Jon Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. Nlp evaluation in trouble: On the need to measure llm data contamination for each benchmark. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10776–10787, 2023.

Mikayel Samvelyan, Sharath C Raparthy, Andrei Lupu, Eric Hambro, Aram H Markosyan, Manish Bhatt, Yuning Mao, Minqi Jiang, Jack Parker-Holder, Jakob Foerster, et al. Rainbow teaming: Open-ended generation of diverse adversarial prompts. *Advances in Neural Information Processing Systems*, 37:69747–69786, 2024.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31210–31227. PMLR, 2023. URL https://proceedings.mlr.press/v202/shi23a.html.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014. doi: 10.1145/2629489.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. BrowseComp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025. URL https://arxiv.org/abs/2504.12516.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh Agrawal, Sandeep Singh Sandha, Siddartha Venkat Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. LiveBench: A challenging, contamination-limited llm benchmark. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/e4a46394ba5378b3f9a186a5b4c650d1-Abstract-Conference.html.

Wikidata contributors. Wikidata: Statistics. https://www.wikidata.org/wiki/Wikidata:Statistics, 2026. Accessed September 2026.

Zhehao Zhang, Jiaao Chen, and Diyi Yang. DARG: Dynamic evaluation of large language models via adaptive reasoning graph. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-4317. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/f5198bc255e1d5f959edd6d1d1a86fab-Abstract-Conference.html.

Jiarui Zhao, Rongzhi Zhang, Lingchuan Liu, Hao Yang, Xunliang Cai, and Xi Su. Lohosearch: Benchmarking long-horizon search agents beyond the human difficulty ceiling. *arXiv preprint arXiv:2606.12837*, 2026.

Xinyue Zheng, Haowei Lin, Kaichen He, Zihao Wang, Qiang Fu, Haobo Fu, Zilong Zheng, and Yitao Liang. MCU: An evaluation framework for open-ended game agents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 78221–78259. PMLR, 2025. URL https://proceedings.mlr.press/v267/zheng25j.html.

Xinyue Zheng, Haowei Lin, Shaofei Cai, Yaodong Yang, Zilong Zheng, and Yitao Liang. UniCode: Augmenting evaluation for code reasoning. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2510.17868.

Heng Zhou, Ao Yu, Yuchen Fan, Jianing Shi, Li Kang, Hejia Geng, Yongting Zhang, Yutao Fan, Yuhao Wu, Tiancheng He, et al. Livesearchbench: An automatically constructed benchmark for retrieval and reasoning over dynamic knowledge. *arXiv preprint arXiv:2511.01409*, 2025.

Kaijie Zhu, Jiaao Chen, Jindong Wang, Neil Zhenqiang Gong, Diyi Yang, and Xing Xie. DyVal: Dynamic evaluation of large language models for reasoning tasks. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=gjfOL9z5Xr.

Appendix

A Task Space Specification

A.1 Complete Task-Space Specification

Table A1 specifies the search coordinates used by the adaptive testers. Selection order controls *which clues are constructed*; presentation order controls *how the selected clues are stated*.

Table A1: **Complete task-space specification.**

Parameter	Values	Generation role
Knowledge scope		
`DOMAIN`	34 entity classes	Target class; e.g., Human, Film, Software.
`POP_RANGE`	$(1,3),(4,15)$; $(16,50),(51,350)$	Wikipedia sitelink interval; entity familiarity.
Constraint structure		
`MIN_C`	$\{3,4,5\}$	Minimum positive clues per isolation stage.
`MAX_C`	$\{5,6,7\}$	Maximum positive clues per isolation stage.
`NEG_C`	$\{0,1,2,3\}$	Grounded exclusionary constraints.
`MIN_FILTER`	$\{1,2,3\}$	Candidate-size control before final isolation.
`SELECTION_ORDERING`	`desc`, `asc`, `random`	Broad-first, specific-first, or random clue selection.
Reasoning topology		
`REASONING_TYPE`	`single_layer`, `sub_property`, `multi_layer`	Direct identification, property projection, or connected entity-isolation stages.
`HOPS`	$\{1,2,3\}$	Canonical task-depth setting; conditional on reasoning type.
`MULTI_LAYER_ISOLATION`	`global`, `prev_link`	Isolate in a global domain or among candidates linked from the preceding entity.
Surface realization		
`LANGUAGE`	`zh`, `en`, `ja`, `fr`, `de`, `es`	Question language.
`PRESENTATION_ORDERING`	`desc`, `asc`, `random`	Broad-first, specific-first, or random presentation of selected clues.
Admissible schema cells	**4,758,912**	Before graph-realizability filtering.

Conditional schema. Single-layer uses $(\texttt{HOPS}, \texttt{ISOLATION}) = (1, \texttt{global})$; sub-property uses $(2, \texttt{global})$; multi-layer permits $\{2,3\} \times \{\texttt{global}, \texttt{prev_link}\}$. These yield six admissible topology tuples. The total schema count is $34 \times 4 \times 3 \times 3 \times 4 \times 3 \times 3 \times 6 \times 6 \times 3$. This is not the number of successfully generated questions. The original draft's grounded-capacity estimate is $\gtrsim 10^{11}$ variants, based on approximately 7.1×10^6 entities/paths and 2.2×10^4 constraint subsets; it remains a combinatorial estimate, not a measured generation yield.

A.2 Task Generation Algorithm

Algorithm 1 Entity isolation for verified task generation

Require: Configuration $\mathbf{c}$, Wikidata graph $\mathcal{G}$, admissible clue budget K
Ensure: Verified task $(Q, e^\star)$ or $\emptyset$
Sample $e^\star$ satisfying the requested domain and popularity interval.
Retrieve its admissible type/relation clue pool $\mathcal{H}$; $\mathcal{P} \leftarrow \emptyset$.
for $k = 1, \ldots, K$ **do**
    Select $h_k \in \mathcal{H}$ according to $\mathbf{c}$; $\mathcal{P} \leftarrow \mathcal{P} \cup \{h_k\}$.
    $\mathcal{S} \leftarrow \{e : e \text{ satisfies the selected clues}\}$.
    if $k < K$ and $|\mathcal{S}| = 1$ **then return** $\emptyset$.
end for
Complete $\mathcal{P}$ with requested grounded exclusions and execute its SPARQL query.
if its answer set is not exactly $\{e^\star\}$ **then return** $\emptyset$.
$Q \leftarrow \textsc{Render}(\mathcal{P}, \mathbf{c}.\texttt{LANGUAGE}, \mathbf{c}.\texttt{PRESENTATION_ORDERING})$.
return $(Q, e^\star)$ if $\textsc{SemanticValidator}(Q, e^\star)$, otherwise $\emptyset$.

Single-layer returns the isolated entity; sub-property projects a requested property; multi-layer composes isolation stages and checks the final endpoint. K and intermediate filtering respect the configured count and filter bounds.

A.3 Matched Examples: Same Answer, Different Evidence

The three new illustrations share the intended answer **Christopher Nolan** (`Q25191`). Single-layer identifies a person directly; sub-property identifies the hidden film *The Odyssey* (`Q131547207`) and asks for its director; multi-layer supplies clues for both entities. Human and Film are both in the canonical domain set. These examples illustrate construction, not measured model difficulty.

Table A2: **Construction-level comparison of the three examples.** Positive-clue counts exclude the domain type and the inter-entity link. All use English and zero negative clues; multi-layer uses global isolation.

	Single-layer	**Sub-property**	**Multi-layer**
Start domain	Human	Film	Film
Positive clues per stage	3	3	$3 + 3$
Inter-entity relation	—	Director	Director
Film / person anchors	0/3	3/0	3/3
Canonical depth setting	1	2	2

A.3.1 Questions and Layer-Specific Clues

The separators below distinguish clue groups for exposition. They are removed from the ordinary-prose questions sent to agents.

Example 1: Single-layer — identify the person

Question. Which English-speaking film director is married to Emma Thomas?

Clues. Human | speaks English | film director | spouse: Emma Thomas.

Intended answer. Christopher Nolan (`Q25191`). Only the person-level clues are available; no film or cast is supplied.

Example 2: Sub-property — identify the film, then its director

Question. Who directed the English-language film whose cast includes Tom Holland and Matt Damon?

Clues. Film | English | Tom Holland | Matt Damon $\xrightarrow{director}$ requested person.

Hidden base. *The Odyssey*, the film (`Q131547207`). **Intended answer.** Christopher Nolan (`Q25191`). No biographical clue about the director is supplied.

Example 3: Multi-layer — identify both linked entities

Question. An English-language film features Tom Holland and Matt Damon. ‖ Its director speaks English, works as a film director, and is married to Emma Thomas. ‖ Who directed the film?

Layer 1. Film | English | Tom Holland | Matt Damon.

Link. film $\xrightarrow{\texttt{P57: director}}$ person.

Layer 2. Human | speaks English | film director | spouse: Emma Thomas.

Intended answer. Christopher Nolan (`Q25191`).

More layers can supply more anchors. The single-layer example supplies only person-level evidence; sub-property supplies only film-level evidence and a requested attribute. Multi-layer combines both: the cast can localize the film, while the spouse clue can localize its director. The link then supports cross-checking. Thus, extra layers may add useful retrieval anchors instead of adding difficulty.

A.4 Executable Queries and Verification Scope

The queries return the final answer without fixing its QID or truncating the result to one row. The verifier separately checks each full isolation stage and requires non-singleton prefixes before the final selected clue.

Single-layer: direct person isolation.

```
PREFIX wd:  <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
SELECT DISTINCT ?answer WHERE {
  ?answer wdt:P31   wd:Q5 ;        # human
          wdt:P1412 wd:Q1860 ;    # speaks English
          wdt:P106  wd:Q2526255 ; # film director
          wdt:P26   wd:Q242951 .  # spouse: Emma Thomas
}
```

Sub-property: isolate the film, then project its director.

```
PREFIX wd:  <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
SELECT DISTINCT ?answer WHERE {
  ?film wdt:P31  wd:Q11424 ;       # film
        wdt:P364 wd:Q1860 ;        # original language: English
        wdt:P161 wd:Q2023710 ;     # cast: Tom Holland
        wdt:P161 wd:Q175535 ;      # cast: Matt Damon
        wdt:P57  ?answer .        # director
}
```

Multi-layer: film and person clues with global isolation.

```
PREFIX wd:  <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
SELECT DISTINCT ?answer WHERE {
  # Layer 1: identify the hidden film.
  ?film wdt:P31  wd:Q11424 ;
        wdt:P364 wd:Q1860 ;
        wdt:P161 wd:Q2023710 ;
        wdt:P161 wd:Q175535 ;
        wdt:P57  ?answer .
  # Layer 2: independently identify the director (global).
  ?answer wdt:P31   wd:Q5 ;
          wdt:P1412 wd:Q1860 ;
          wdt:P106  wd:Q2526255 ;
          wdt:P26   wd:Q242951 .
}
```

Supporting statements. Wikidata records the relevant person facts at `Q25191`, the film facts at `Q131547207`, and Emma Thomas at `Q242951` (checked 22 September 2026). A small offline fixture additionally includes Steven Spielberg (`Q8877`) and *Spider-Man: Homecoming* (`Q23010088`) as non-target controls. Its counts are not full-graph counts.

Verification boundary. Singleton SPARQL results certify uniqueness only in the queried graph. Semantic validation checks clue preservation, answer type, polarity, direction, and entity leakage. A single-layer open-web gate can seek concrete counterexamples, but does not prove uniqueness over all real-world facts.

B Model-Specific Failure Profiles and Weakness Analysis

B.1 Marginal Degradation and Sampling Enrichment

To characterize model vulnerability, we analyze the accuracy differences between the late two rounds and the early two rounds for each model. Specifically, we identify the top-5 worst interpretable marginal dimensions—spanning domain, language, and popularity—and evaluate the intensified attack sampling strength applied by Active Testing on these vulnerable regions, as shown in Figure A1.

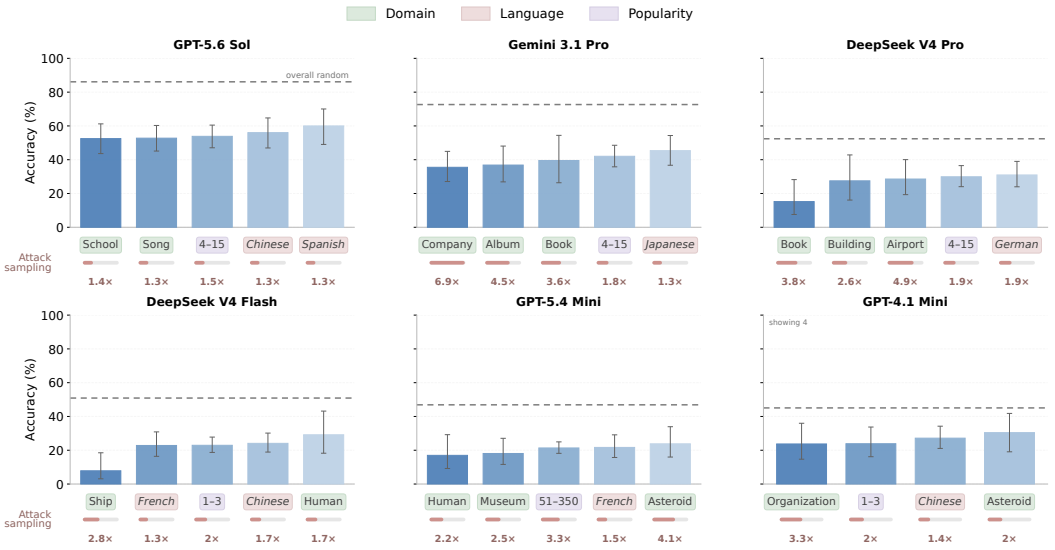


Figure A1: **Top-5 weak region marginal accuracy visual of different models.** Main bars show weak-region accuracy, dashed reference lines show initial random accuracy, and lower red bars show late-round sampling enrichment. We observe that model-specific weak regions emerge across agents, which Active Tests target with intensified attack sampling.

B.2 Model-Specific Semantic Weaknesses

Active testing identifies School and Song weaknesses for GPT-5.6 Sol, Book enrichment for DeepSeek V4 Pro, and Company enrichment for Gemini 3.1 Pro. The targeted domains differ across models. Non-English queries also recur among the weak regions, while popularity effects are model-dependent rather than uniformly concentrated on the least popular entities.

Regional accuracy measures performance; enrichment measures budget allocation. Neither alone makes a feature intrinsically difficult: a heavily sampled region can co-occur with other weak coordinates, and a low-accuracy region can remain sparsely observed.

B.3 Marginals Are Diagnostic, Not Causal

The reported marginals summarize complete, adaptively sampled TaskConfigs. They do not hold the other coordinates fixed: domain can co-vary with reasoning structure, language, or clue selection. In addition, initial overall accuracy and late regional accuracy refer to different sampling distributions, it is not an isolated causal effect of changing one coordinate. The main paper's shared weakness analysis complements these profiles with cross-dimensional associations. It remains observational and does not remove selection effects induced by adaptive sampling.

C Baseline Settings

C.1 Shared Experimental Protocol

The methods start from the same 300-task random warm start: 150 configurations with two independently generated questions each. Eight adaptive rounds then target 100 configurations per round, split into 30 historical repeats and 70 new configurations. Budgets refer to target evaluated configurations; generation failures and rejected proposals are not successful evaluations.

Table A3: **Shared experimental protocol.**

Parameter	Setting
Warm-start configurations	150
Questions per warm-start configuration	2
Total warm-start tasks	300
Adaptive attack rounds	8
Total evaluation rounds, including warm start	9
Target configurations per adaptive round	100
Questions per configuration	2
Maximum generation attempts per configuration	30
Question-generation model	`gpt-5.5`
Question-validation model	`gpt-5.4-mini`
Answer-judging model	`gpt-5.4-mini`

C.2 Baseline Design and Implementation in One Table

Table A4: RL baseline implementation settings.

Setting / Parameter	RL No-history	RL History
Policy hidden dimension	64	128
History-set embedding	—	64
Action embedding	16	16
Learning rate	3×10^{-4}	3×10^{-4}
Weight decay	10^{-4}	10^{-4}
Warm-training epochs	15	80
Training batch size	64	64

We observed that the history-conditioned RL policy fails to converge under low-data regimes with small warm-start samples. Consequently, in the main text, the history-conditioned RL baseline utilizes a larger warm-start dataset (1,400 tasks vs. 300 tasks for RF) alongside an extended warm-training schedule to ensure stable convergence.

C.3 RF Acquisition and Cosine Schedule

Table A5: **RF implementation settings.** $B = 100$ is the per-round configuration target; t indexes the adaptive rounds from zero.

Parameter	Setting
Number of trees	300
Minimum samples per leaf	4
Random seed	42
Maximum tasks per proposal cell	6
Diversity-calibration pool	8000

D Case Studies

This section presents a complete case, including the SPARQL query, the original question, the gold answer, and the search queries, search summaries, final answers, confidence scores, and evaluation results of five agents. The task is specified in Table A6, and the overall agent results are summarized in Table A7.

D.1 Task

Table A6 reports the original question, structured constraints, and SPARQL query for this case.

Table A6: Case task, structured constraints, and SPARQL query.

Item	Content
Original question	所有関係の記録に所有者としてイェール大学美術館およびマルマデ・ギャラリーが挙げられており、樽を描いている絵画のジャンルは何ですか。
English translation	What is the genre of the painting that depicts a barrel and whose ownership records list both the Yale University Art Gallery and the Malmadé Gallery?
Target entity	*An Allegory of Intemperance* (Q2091114)
Target artwork	
Gold answer	寓意 (allegory; Q2839016)
Structured constraints	`instance of` (P31) = painting (Q3305213); `owned by` (P127) = Yale University Art Gallery (Q1568434); `owned by` (P127) = Malmadé Gallery (Q134377234); `depicts` (P180) = barrel (Q10289); target property `genre` (P136) = allegory (Q2839016).

SPARQL	`SELECT DISTINCT ?painting ?answer ?answerLabel`
	`WHERE {`
	`  ?painting`
	`  wdt:P31  wd:Q3305213 ;`
	`  wdt:P127 wd:Q1568434 ;`
	`  wdt:P127 wd:Q134377234 ;`
	`  wdt:P180 wd:Q10289 ;`
	`  wdt:P136 ?answer .`
	`  SERVICE wikibase:label {`
	`  bd:serviceParam wikibase:language "ja,en" .`
	`  }`
	`}`

D.2 Agent Results

Table A7 summarizes the number of search calls, final answer, and evaluation result for each agent. A single search call may contain multiple semicolon-separated queries. The complete recorded outputs are reported in Tables A8–A12. The evaluation logs do not contain a separate internal-reasoning field for GLM-5.2, GPT-5.6 Sol, Gemini 3.1 Pro, or DeepSeek V4 Pro; we therefore report only the recorded queries, search summaries, and answers. The explicit explanation emitted by DeepSeek V4 Flash before its JSON output is retained in Table A12.

Table A7: Overall performance of the five agents on the case.

Agent	Search calls	Final answer	Evaluation
GLM-5.2	13	アレゴリー（寓意画 / allegory）	Correct
GPT-5.6 Sol	4	寓意画（allegory）	Correct
Gemini 3.1 Pro	7	寓意画（または宗教画）	Correct
DeepSeek V4 Pro	10	寓意画（アレゴリー、allegory）	Correct
DeepSeek V4 Flash	3	静物画（Still Life）	Invalid output

D.2.1 GLM-5.2

The complete query sequence and answer from GLM-5.2 are listed in Table A8.

Table A8: Search trajectory and answer from GLM-5.2.

Field	Content
Search queries	**1.** “Allegory of Intemperance” Bosch Wikidata genre **2.** “Galerie Malmedè” Hieronymus Bosch provenance **3.** “Marmaduke Gallery” art gallery **4.** “Marmaduke Gallery” provenance painting barrel Yale University Art Gallery **5.** Hieronymus Bosch “Allegory of Intemperance” Yale University Art Gallery genre classification barrels **6.** Wikidata Q2091114 “An Allegory of Intemperance” genre instance of **7.** Wikipedia painting barrel provenance “Yale University Art Gallery” “Marmaduke Gallery” **8.** Yale University Art Gallery collection painting barrel provenance “Marmaduke” **9.** Yale University Art Gallery painting depicting barrels provenance genre **10.** openartbrowser.org Q2091114 “An Allegory of Intemperance” genre allegory **11.** painting depicting barrels provenance Yale University Art Gallery Marmaduke Gallery owner **12.** site:wikidata.org Q2091114 genre allegory P136 **13.** wikidata.org Q2091114 genre P136 allegory
Search summary	The painting in question is ’An Allegory of Intemperance’ by Hieronymus Bosch (ca. 1495–1500) at the Yale University Art Gallery (accession no. 1959.15.22). Its provenance records list both ’Galerie Malmedè, Cologne’ (マルマデ・ギャラリー in Japanese) and ’Yale University Art Gallery’ (イェール大学美術館) as owners. The painting depicts a fat man riding a barrel. According to Wikidata (Q2091114), the genre (P136) of this painting is ’allegory’ (アレゴリー/寓意画).
Final answer	アレゴリー（寓意画 / allegory）
Confidence	0.95
Evaluation	Correct

D.2.2 GPT-5.6 Sol

The complete query sequence and answer from GPT-5.6 Sol are listed in Table A9.

Table A9: Search trajectory and answer from GPT-5.6 Sol.

Field	Content
Search queries	**1.** “Yale University Art Gallery” “Marmaduke Gallery” barrel painting **2.** “Yale University Art Gallery” “Marmaduke” owner painting **3.** artwork “Marmaduke Gallery” “Yale” **4.** site:artgallery.yale.edu/collections/objects “Allegory of Intemperance”
Search summary	「Yale University Art Gallery」「Malmadé Gallery」「barrel」を組み合わせて検索し、イェール大学美術館の作品記録と Wikidata の関連記録を照合しました。イェールの公式記録は作品名、作者、樽の主題、所蔵・来歴を確認でき、Wikidata はジャンルを「allegory（寓意画）」と明示しています。(artgallery.yale.edu)
Final answer	その絵画のジャンルは「寓意画（allegory）」です。該当作品はヒエロニムス・ボスの《放蕩の寓意（An Allegory of Intemperance）》で、樽を描いており、所有関係にはイェール大学美術館とマルマデ・ギャラリーが含まれます。(artgallery.yale.edu)
Confidence	0.99
Evaluation	Correct

D.2.3 Gemini 3.1 Pro

The complete query sequence and answer from Gemini 3.1 Pro are listed in Table A10.

Table A10: Search trajectory and answer from Gemini 3.1 Pro.

Field	Content
Search queries	**1.** “Allegory of Gluttony and Lust” genre **2.** “Allegory of Intemperance” genre **3.** “An Allegory of Intemperance” Bosch barrel **4.** “Yale University Art Gallery” “Malmede Gallery” **5.** “Yale University Art Gallery” “Malmede” painting **6.** “Yale University Art Gallery” “Malmadé” **7.** “Yale University Art Gallery” provenance Malmede barrel
Search summary	検索結果より、所有履歴（プロヴィナンス）に「マルマデ・ギャラリー（Kunsthaus Malmedé)」や「イェール大学美術館（Yale University Art Gallery)」が記録されており、樽（に乗った人物など）が描かれている絵画は、ヒエロニムス・ボッシュの『大食と淫蕩の寓意（または放蕩の寓意 / An Allegory of Intemperance)』であることが確認できました。美術作品としてのジャンルは、Wikidata や各種美術データベースにおいて「寓意画（Allegory)」および「宗教画（Religious painting)」と分類されています。
Final answer	寓意画（または宗教画）
Confidence	0.95
Evaluation	Correct. The primary answer matches the gold answer, although the model additionally mentions “religious painting.”

D.2.4 DeepSeek V4 Pro

The complete query sequence and answer from DeepSeek V4 Pro are listed in Table A11.

Table A11: Search trajectory and answer from DeepSeek V4 Pro.

Field	Content
Search queries	**1.** “Allegory of Intemperance” genre painting Bosch; Galerie Malmedè Yale Bosch; Bosch Allegory of Intemperance Malmedè **2.** “An Allegory of Intemperance” Yale University Art Gallery provenance; “Allegory of Intemperance” Jheronimus Bosch Yale; Yale 1959.15.22 provenance **3.** “マルマデ” ギャラリー; “Marumade” gallery; マルマデ画廊 **4.** Yale University Art Gallery Marmade Gallery barrel painting; イェール大学美術館マルマデ・ギャラリー樽絵画; Marmade Gallery Yale University Art Gallery painting barrel **5.** Yale University Art Gallery Marmade provenance; “Marmade” gallery art painting; Marmade Yale provenance painting **6.** Yale University Art Gallery barrel painting provenance; “Yale University Art Gallery” barrel painting; 樽絵画イェール大学美術館 **7.** Yale University Art Gallery still life barrel; “Yale University Art Gallery” “barrel” painting; Yale Art Gallery “cask” or “barrel” painting **8.** artgallery.yale.edu 43515 Allegory of Intemperance; Yale University Art Gallery Bosch 1959.15.22 classification **9.** マルマデギャラリー美術; Marmade Gallery 日本; Marmade gallery Yale **10.** マルマデ・ギャラリー絵画; Marmade Gallery; Marmade gallery art
Search summary	题目中「マルマデ・ギャラリー」对应的是德文「Galerie Malmedè」(科隆的马尔梅德画廊)。通过检索该画廊与耶鲁大学美术馆（Yale University Art Gallery）共同出现在所有权记录（provenance）中的画作，可确定该画作为希罗尼穆斯·博斯（Hieronymus Bosch）创作的《An Allegory of Intemperance》(《不節制の寓意》/《暴食と欲望の寓意》)。画中描绘了一个胖子骑在桶上（樽を描いている)。根据 WikiArt 和 Wikidata 的权威分类，该画作的「ジャンル」(genre）为「寓意画(allegory)」，同时也可归入「宗教画（religious painting)」。本题以「寓意画」为核心答案。
Final answer	寓意画（アレゴリー、allegory）
Confidence	0.95
Evaluation	Correct

D.2.5 DeepSeek V4 Flash

The complete query sequence, explicit answer explanation, and output evaluation from DeepSeek V4 Flash are listed in Table A12.

Table A12: Search trajectory and answer from DeepSeek V4 Flash.

Field	Content
Search queries	**1.** Marmaduke Gallery painting Yale University Art Gallery barrel; イェール大学美術館マルマデギャラリー樽絵画; Yale University Art Gallery painting cask barrel owner Marmaduke Gallery **2.** マルマデギャラリー絵画樽; Marmaduke Gallery painting barrel still life; 樽を描いた絵静物画イェール **3.** 樽を描いた絵画イェール大学美術館マルマデ・ギャラリー; painting barrel Yale University Art Gallery Marmaduke Gallery genre; Yale University Art Gallery still life barrel painting
Search summary	Empty
Answer explanation	基于检索结果，我现在回答您的问题。题目描述：在所有关系（所藏/所有权）记录中，所有者同时列有“耶鲁大学美术馆（Yale University Art Gallery）”和“马尔马杜克画廊（Marmaduke Gallery）”，并描绘了樽（木桶/大桶）的绘画，其画种（ジャンル）是**静物画（still life）**。依据：耶鲁大学美术馆的藏品记录中，此类以桶、酒桶、器皿等日常静止物体为主体的绘画，其归属类别（Classification）均为“Paintings”，而画种归类即“Still Life（静物画）”。综合检索结果与绘画题材（以樽为描绘对象）判断，该画作属于静物画这一画种。
Final answer	静物画（Still Life）
Confidence	0.78
Evaluation	`INVALID_OUTPUT`. The JSON object was preceded by additional text, so semantic evaluation was not performed; the answer “still life” also differs from the gold answer.

E Prompts

This section presents instantiated task-generation examples together with the general evaluated-agent instructions and the prompts used by the two LLM red-team baselines. Text enclosed in braces denotes content filled dynamically at runtime.

E.1 Single-Layer Generation

Prompt 1: Single-layer generation (instantiated excerpt).

```
SYSTEM
You are a helpful puzzle engine. The question prose must be entirely in English.
Output only the requested format.

USER
You are a Puzzle Generation Expert. Craft a natural language puzzle based on the
following tracking constraints.

[CRITICAL]
Output strictly in English. Do NOT mention the Target Name 'Christopher Nolan'
directly.

[TARGET CONSTRAINTS]
Positive:
- instance of: human
- languages spoken: English
- occupation: film director
- spouse: Emma Thomas
Negative: none

[CORE TASK]
Blend these clues smoothly. The question text MUST end by asking to identify the
target entity itself.

[QUALITY RULES]
Ask for exactly one answer: the target entity.
The hidden target must be a direct grammatical answer to the final question.
Keep every clue attached to the entity it describes. Render negative clues only
as exclusions; never turn them into positive facts.
Do not invent context, history, aliases, successors, predecessors, or other
facts.

[OUTPUT FORMAT]
Question: [The puzzle text]
Do not output the answer.
```

Corresponding question. Which English-speaking film director is married to Emma Thomas?

Intended answer. Christopher Nolan (`Q25191`).

E.2 Sub-Property Generation

Prompt 2: Sub-property generation (instantiated excerpt).

```
SYSTEM
You are a helpful puzzle engine. The question prose must be entirely in English.
Output only the requested format.

USER
You are a Puzzle Generation Expert. Craft a 2-step single-target attribute
sub-property question.

[CRITICAL]
Output strictly in English. Do NOT mention the base name 'The Odyssey' or the
answer 'Christopher Nolan'.

[STEP 1 CONSTRAINTS TO IDENTIFY BASE ENTITY]
Positive:
- instance of: film
- original language of film: English
- cast member: Tom Holland
- cast member: Matt Damon
Negative: none

[STEP 2 SUB-PROPERTY EXTENSION]
After identifying the base entity via Step 1 clues, the question MUST ask for its
specific attribute/property: "director".

[QUALITY RULES]
Ask for exactly one answer: the value of "director". The base entity is only an
implicit clue and must not also be requested.
The private answer 'Christopher Nolan' must directly and grammatically answer the
final question, but must not appear in the question.
Write the base entity as a definite descriptive noun phrase, never as an
interrogative.
Do not reverse the property direction or attach a base clue to the answer entity.
Do not add unsupported facts, narrative history, aliases, successors, or
predecessors.

[OUTPUT FORMAT]
Question: [The puzzle text]
Do not output the answer.
```

Corresponding question. Who directed the English-language film whose cast includes Tom Holland and Matt Damon?

Intended answer. Christopher Nolan (`Q25191`).

E.3 Multi-Layer Generation

Prompt 3: Multi-layer generation (instantiated excerpt).

```
SYSTEM
You are a helpful puzzle engine. The question prose must be entirely in English.
Output only the requested format.

USER
You are a knowledge-graph multi-hop question generator. Rewrite the given
structured chain into one natural, clear, and searchable question.

[CURRENT STRUCTURED CHAIN]
Start entity: The Odyssey [film; private, do not reveal]
- instance of: film
- original language of film: English
- cast member: Tom Holland
```

```
- cast member: Matt Damon

Relation to next entity:
film --P57: director--> person

Final entity: Christopher Nolan [private, do not reveal]
- instance of: human
- languages spoken: English
- occupation: film director
- spouse: Emma Thomas

[GLOBAL ISOLATION]
Each layer has its own identifying clues, and the relation connects it to the
previous layer. Preserve the chain order, but express each connection as a
natural attachment rather than as a traversal step.

[HARD RULES]
Do not directly mention any target entity label from the chain.
Do not add facts, relations, entities, or reasoning steps that are not provided.
Do not use explicit structural labels such as "Layer 1" or "Step 2" in the final
question.
The final question must ask for the final entity only.
Preserve every relation direction exactly according to its P-ID metadata.
Keep each clue on its own entity. No clue may be omitted.

[OUTPUT FORMAT]
Question: [The puzzle text]
Do not output the answer.
```

Corresponding question. An English-language film features Tom Holland and Matt Damon. Its director speaks English, works as a film director, and is married to Emma Thomas. Who directed the film?

Intended answer. Christopher Nolan (`Q25191`).

E.4 Evaluated-Agent Prompt

Prompt 4: Evaluated agents—general rules.

```
You are a real problem-solving agent with web-search access.
Use the built-in web_search tool to research step by step and answer the
question.

[HARD REQUIREMENTS]
1. You must use the built-in web_search tool. Do not guess blindly.
2. The final response must be valid JSON that json.loads can parse, with no
surrounding Markdown.
3. The JSON must contain exactly these fields:
- final_answer: your exact final answer
- search_summary: a concise summary of your search strategy and conclusion
- confidence: a number from 0 to 1

[QUESTION]
{question}
```

E.5 LLM Red-Team Prompts

E.5.1 LLM Red-Team v1: Subspace Guidance

Prompt 5: LLM Red-team v1—subspace-guidance prompt.

```
USER
You are an expert in advanced multi-agent evaluation. Your goal is to map the
tested agent's capabilities as quickly as possible over eight interactive rounds
and discover as many capability blind spots as possible.

This is {round_label}. Use the historical information to determine whether the
exploration directions in earlier rounds were correct, identify plateaus, change
points, misattributions, and overlooked key parameters, and provide the subspace
and must_test_points most worth testing in the next round.

[SOLE CANONICAL TASKCONFIG SAMPLING SPACE]
{canonical_space_json}

[CONDITIONAL CONSTRAINTS — MUST BE OBEYED]
- single_layer fixes HOPS=1 and MULTI_LAYER_ISOLATION="global".
- sub_property is a genuine two-step task and fixes HOPS=2 and
MULTI_LAYER_ISOLATION="global".
- multi_layer permits only HOPS in [2,3]; MULTI_LAYER_ISOLATION is an active
sampling dimension.
- Multi-hop tasks always prohibit reusing an entity that already appears in a
positive clue at the current layer; this is not an experimental parameter.
- ORDERING is only a legacy alias for SELECTION_ORDERING. Do not treat ORDERING
as an independent cell dimension.
- subspace and must_test_points may use only the fields and values listed above.

[CURRENT GLOBAL WEAKEST POINTS]
Prioritize the top 20 TaskConfigs with accuracy below 20%, and extract the
weakness patterns most worth attacking further.
{weakest_configs_text}

[Global Trajectory Matrix]
{global_trajectory_matrix}

[Marginal Attribute Distributions]
{marginal_attribute_distributions}

[Historical Hypotheses and Alignment Checks]
{historical_hypotheses_and_alignment_checks}

[HISTORY LOG]
Read the history round by round. Keep the information compact and non-redundant;
retain only the key conclusions that support attribution and the next-round
decision.
{history_log}

[CURRENT-ROUND LLM ANALYSIS]
The current batch summary is:
{current_batch_json}
{rf_shap_prior_text}

[TASKS]
1. Determine whether the exploration directions in previous rounds were correct
and whether any plateau or misattribution occurred.
2. Summarize the model's stable capabilities, vulnerable capabilities, and
boundary capabilities.
3. Provide the subspace to explore in the next round and the must_test_points
that must be tested.
4. Report historical hypotheses and alignment checks, explicitly stating which
hypotheses were validated and which may be misattributions.
5. Output strict JSON with no additional explanation.
6. Emphasize exploration in early rounds across different configurations,
domains, and languages. In later rounds, exploit more when appropriate and
generate tasks the agent is certain to be unable to solve.
```

```
[OUTPUT REQUIREMENTS]
Output strict JSON containing at least these top-level fields:
- overall_summary
- error_analysis
- dimension_analysis
- hypothesis_alignment
- next_round_guidance

next_round_guidance must contain:
- strategy: "EXPLOIT" / "EXPLORE" / "BOUNDARY_PROBE" / "MIXED"
- subspace: a strict JSON object or array representing the parameter subspace to
explore next round
- must_test_points: a strict JSON array representing the key points that must be
validated next round
- rationale: a brief explanation of why these tests should be run
- recommended_search_space: an expression compatible with subspace for the legacy
sampling logic
```

E.5.2 LLM Red-Team v2: Direct Configuration Proposal

Prompt 6a: LLM Red-team v2—system template.

```
SYSTEM
You are an automated red-team attacker for a web-search agent.

Your only action is to choose TaskConfig configurations from a provided discrete
task space.
You do NOT write questions, prompts, jailbreaks, explanations, scores, or
subspaces.
The downstream program will instantiate each TaskConfig into a verified
Wikipedia/Wikidata task and evaluate the target agent.

Your goal is to discover failure regions of the target agent. Propose TaskConfigs
that jointly satisfy two objectives:
1. LOW ACCURACY: configurations should be as likely as possible to make the
target agent answer incorrectly.
2. DIVERSITY: the proposed batch should cover substantially different TaskConfig
regions and avoid collapsing onto one narrow pattern.

Use the observed historical realized-TaskConfig/accuracy table to exploit
reliable weaknesses, but also test distinct combinations and interactions that
may reveal new weak regions. The history coordinate is the TaskConfig actually
instantiated and evaluated, not a request that the generator silently changed.
Diversity means diversity in the explicit TaskConfig dimensions, not superficial
wording diversity.

Important ordering rule: the downstream generator may stop after it obtains
enough valid tasks. Therefore diversity must hold throughout the ordered list,
including early prefixes. Interleave different domains, languages, reasoning
types, and structural settings rather than grouping similar configurations
together.

Hard constraints:
- Every returned item must be a compact JSON array representing one COMPLETE
TaskConfig.
- Every config array must contain exactly {len(TASK_CONFIG_CELL_FIELDS)} values
in this fixed field order:
  {compact_fields}
- Do not include field names inside individual config arrays. POP_RANGE remains a
nested two-integer array value.
- Every value must come from the allowed values in the schema.
- Prefer distinct TaskConfigs. The runner processes candidates in order and skips
a cell once its execution budget is exhausted.
```

```
- The runner separately allocates the historical-repeat slice. Avoid exact
TaskConfigs shown in the weakness summary; the runner filters any other
historical exact repeats locally.

Return STRICT JSON only, with exactly one top-level key:
{"configs":[[value1,value2,...,value{len(TASK_CONFIG_CELL_FIELDS)}], ...]}

Target {n} compact TaskConfig arrays in configs, but returning fewer is allowed.
No text before or after the JSON.
```