BLUE: BENCHMARKS, LET US EVOLVE WITH AGENT

Anonymous authors
Paper under double-blind review

ABSTRACT

As LLM agents acquire new skills, tools, and other actions, their capability space continues to expand. However, existing benchmarks are either static or generate more complex tasks only within a previously defined capability space, creating a growing mismatch between what agents can do and what benchmarks can evaluate. To address this gap, we introduce BLUE, an evolvable benchmark generator for agent evaluation. For each newly introduced capability, BLUE requires only a simple user-supplied atomic task and its corresponding evaluation. It groups atomic tasks into atomic scenarios, constructs a scenario graph to generate complex tasks, and simultaneously composes atomic evaluations into evaluators for these complex tasks. BLUE combines rule-based metrics with LLM judges and supports both single-turn and multi-turn evaluation. We construct BLUE-Bench, which covers diverse multi-capability scenarios, to systematically evaluate five representative coding agents paired with three foundation models. We further conduct a detailed analysis of BLUE key components to validate the effectiveness of the framework. All code and data are available at https://anonymous.4open.science/r/BLUE-ICLR.

1 INTRODUCTION

LLM agents increasingly act as general-purpose systems that interact with external environments through a diverse set of capabilities, such as skills (Wang et al., 2024), tools (Schick et al., 2023), and other actions (Yao et al., 2023). Existing manually constructed benchmarks have advanced agent evaluation across code generation, software engineering, web interaction, and tool use (Chen et al., 2021; Jimenez et al., 2024; Liu et al., 2024; Zhou et al., 2024; Mialon et al., 2024). Furthermore, some works (Li et al., 2025; Shi et al., 2026) have explored automatically generating benchmarks using LLMs. These approaches reduce the cost of manual task construction and can produce larger and more challenging tasks that better evaluate pre-defined capabilities of agents.

However, agent capabilities are evolving, and benchmarks defined over a fixed capability space cannot fully capture the evolving capabilities of real-world agents. **When an agent evolves, an ideal benchmark should evaluate not only the new capabilities, but also the composition with existing capabilities in complex, realistic tasks.** As illustrated in Figure 1, agents may acquire new skills, tools, settings, and other capabilities. To fully evaluate the agent, it is high cost for manual test case construction, and existing LLM-based generation methods only support test case expansion within a fixed capability space, rather than capability composition.

In this work, we focus on capability composition expansion. This compositional view fundamentally changes how a benchmark grows: instead of expanding only by adding independent instances, the benchmark can expand combinatorially as newly introduced capabilities interact with existing ones. As a result, the evaluation space can grow substantially while requiring less human effort.

Realizing such capability composition expansion introduces two key challenges: **(1) Intra-task coherence:** how to effectively organize different capabilities to construct realistic tasks together with corresponding evaluations. **(2) Inter-task diversity:** how to generate diverse and non-redundant task compositions at scale while maintaining task quality.

To address these challenges, we introduce **BLUE**, an evolvable benchmark generator that enables capability composition expansion of both tasks and evaluations. For each newly introduced capability, users only need to provide a simple atomic task and its corresponding evaluation. BLUE first groups atomic tasks into atomic scenarios, then constructs a typed scenario graph to ensure intra-task

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.

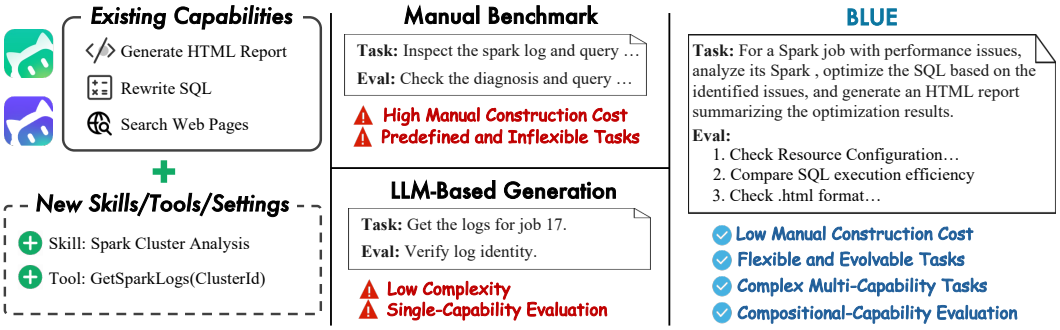


Figure 1: Example of BLUE. Existing manual benchmarks require substantial human effort to construct predefined and inflexible tasks, while LLM-based generation typically focuses on simple, single-capability evaluations. In contrast, BLUE enables low-cost generation of evolvable tasks that evaluate complex compositions of existing and newly introduced capabilities.

coherence, and uses a task scheduler to generate diverse multi-capability compositions, controlling inter-task diversity. The corresponding atomic evaluations are further composed into milestone-level and end-to-end evaluators, supporting both single-turn and multi-turn evaluation.

Our contributions are threefold:

- **Generating Evolvable Benchmarks.** We introduce BLUE, a benchmark generation framework that reformulates agent benchmarking as capability composition expansion, enabling benchmarks to evolve alongside newly introduced agent capabilities.
- **Composing Tasks and Evaluations Hierarchically.** BLUE organizes atomic capabilities into coherent multi-step tasks through hierarchical composition, and correspondingly composes their evaluations into milestone-level and end-to-end evaluators.
- **Building BLUE-Bench and Analysing the BLUE framework.** We use BLUE to construct BLUE-Bench and evaluate five agent harnesses paired with three foundation models. We further conduct a detailed analysis of BLUE from multiple perspectives to systematically validate the effectiveness of the framework in generating high-quality tasks.

2 Related Work

Manual Benchmarks: Existing agent benchmarks (Xi et al., 2025; Ma et al., 2024) cover a broad range of capabilities, including code generation, software engineering, tool use (Li et al., 2023b; Qin et al., 2024; Yao et al., 2025), web interaction (Yao et al., 2022; Deng et al., 2023; Drouin et al., 2024; Koh et al., 2024; Trivedi et al., 2024), operating-system interaction, and general reasoning. HumanEval (Chen et al., 2021), SWE-bench (Jimenez et al., 2024), AgentBench (Liu et al., 2024), WebArena (Zhou et al., 2024), OSWorld (Xie et al., 2024), and GAIA (Mialon et al., 2024) provide standardized tasks and evaluators for these settings. However, they are typically released as fixed collections of tasks and evaluation procedures. As agent tools or skills evolve, extending such static benchmarks to newly introduced capabilities generally requires significant manual effort to restructure.

LLM-Based Generation: A growing line of work uses LLM-based pipelines to automatically generate benchmark instances. AutoBencher (Li et al., 2025) generates tasks from predefined desiderata and privileged sources; TaskCraft (Shi et al., 2026) performs depth- and width-wise expansion over a predefined pool of single-tool atomic tasks; Wang et al. (2025) rewrites existing question-answer pair seeds; and Continuous Benchmark Generation (Saxena et al., 2026) derives tasks from available reference commits. These approaches primarily operate within a fixed capability space, generating more or increasingly complex instances.

3 METHOD

BLUE builds a three-level composition hierarchy. An **atomic task** τ is a single capability-level operation, such as loading a dataset, detecting its language, or querying cluster status. An **atomic scenario** s is the smallest self-contained workflow that achieves a coherent business sub-goal. It contains one or more atomic tasks that operate on the same object and are connected by an explicit execution dependency or data flow. A **complex task** q composes multiple atomic scenarios into the final, user-facing task presented to the evaluated agent.

3.1 OVERALL FRAMEWORK

BLUE is a three-stage generation framework that progressively expands atomic tasks into complex tasks for single-turn and multi-turn agent evaluation. For task generation, BLUE first constructs atomic scenarios from fine-grained atomic tasks, then builds a scenario graph to capture inter-scenario relations, and finally generates complex tasks under the guidance of the task scheduler. For metric construction, user-provided atomic evaluations are organized into rule-based metrics and LLM-judge metrics, enabling both single-turn and multi-turn evaluation.

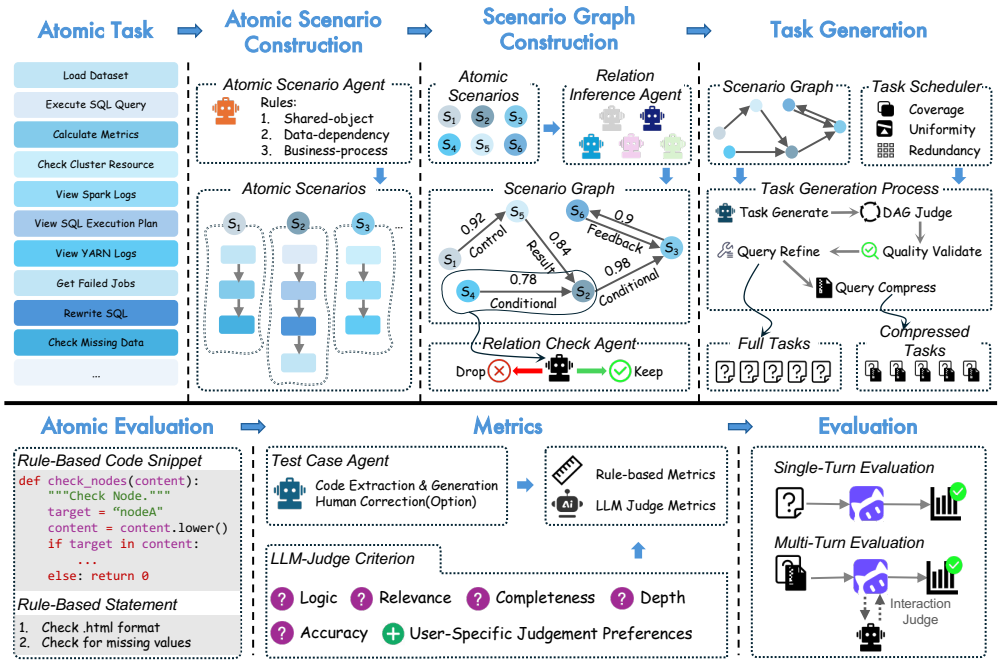


Figure 2: Overall framework of BLUE.

3.2 TASKS

The Tasks section consists of three main modules: Atomic Scenario Construction, Scenario Graph Construction, and Task Generation. In the following, we provide a detailed introduction to each of these three modules.

3.2.1 ATOMIC SCENARIO CONSTRUCTION

The Atomic Scenario Agent transforms fine-grained atomic tasks into atomic scenarios that serve as the nodes of the scenario graph. Tasks are grouped only when they form a plausible local workflow, as determined by three composition cues: **(1) shared-object**: operating on the same object; **(2) data-dependency**: an intermediate result of one task is consumed by another; **(3) business-process**: successive stages of the same operational process. By default, tasks remain separate unless a composition cue is present.

This intermediate abstraction is necessary because individual atomic tasks (e.g., a single API call or SQL execution) carry insufficient semantic context for reliable graph-level relation inference, as they represent only the smallest executable units. Grouping them into atomic scenarios yields semantically enriched units: each scenario now exposes a local business objective, an explicit list of participating objects, and clear boundaries.

3.2.2 SCENARIO GRAPH CONSTRUCTION

To accurately model inter-scenario relations, we construct a typed scenario graph $G = (\mathcal{S}, E)$, which encodes logical dependencies among atomic scenarios. Each node $s_i \in \mathcal{S}$ represents one atomic scenario, and each directed edge $e_{ij} = (s_i, s_j, r_{ij}, c_{ij}, h_{ij})$ contains a typed relation r_{ij}, a confidence score $c_{ij} \in [0, 1]$, and natural-language evidence h_{ij}. BLUE provides five relation types: FEEDBACK, VALIDATION, CONDITIONAL DEPENDENCY, RESULT DEPENDENCY, and CONTROL DEPENDENCY. Full definitions are provided in Appendix B.

Reliable scenario graph construction is the foundation for subsequent complex task generation. It consists of two main components: multi-round voting for inter-scenario relation inference and an adversarial check for filtering low-quality edges.

Relation Inference. To mitigate the unreliability and instability of single-round LLM relation inference, BLUE adopts a majority-voting mechanism. For each remaining scenario pair (s_i, s_j), BLUE performs M independently configured voting rounds. Each round v outputs $(r_{ij}^{(v)}, c_{ij}^{(v)}, h_{ij}^{(v)})$.

Let $n_{ij}(r) = \left|\left\{v : r_{ij}^{(v)} = r\right\}\right|$ denote the number of rounds supporting relation r. BLUE instantiates an edge only if the most-supported relation in the relation set $\mathcal{R}$ receives at least τ_n votes:

$$r_{ij}^* = \arg\max_{r \in \mathcal{R}} n_{ij}(r), \qquad n_{ij}(r_{ij}^*) \geq \tau_n. \tag{1}$$

For the supporting rounds $\mathcal{V}_{ij} = \{v : r_{ij}^{(v)} = r_{ij}^*\}$, we average their confidence scores:

$$c_{ij}^* = \frac{1}{|\mathcal{V}_{ij}|} \sum_{v \in \mathcal{V}_{ij}} c_{ij}^{(v)} \tag{2}$$

and retain the evidence from the highest-confidence round:

$$h_{ij}^* = h_{ij}^{(v^*)} \text{ where } v^* = \arg\max_{v \in \mathcal{V}_{ij}} c_{ij}^{(v)}. \tag{3}$$

Finally, the resulting edge is $e_{ij} = (s_i, s_j, r_{ij}^*, c_{ij}^*, h_{ij}^*)$.

Relation Check. To further improve relation reliability, candidate edges whose aggregated confidence is below τ_c undergo adversarial verification by a reverse verifier. Given only the two scenario descriptions and the predicted relation r_{ij}^*, the verifier attempts to falsify the edge by examining business plausibility, relation-type consistency, directional validity, and competing explanations (Appendix B). The edge is retained only if no substantive counter-evidence is found. To reduce confirmation bias, the verifier has no access to the selected evidence h_{ij}^* or aggregated confidence c_{ij}^*. Rejected edges are removed, whereas verified edges retain $(r_{ij}^*, c_{ij}^*, h_{ij}^*)$.

3.2.3 TASK GENERATION

Based on the constructed scenario graph, BLUE generates complex tasks through two steps: Task-Scheduler-guided workflow sampling and workflow structural validation.

Task Scheduler. To avoid excessively long contexts when generating hundreds of tasks at once, BLUE adopts a batch-based generation strategy, producing n_g tasks per batch. To ensure broad scenario coverage and uniform usage while maintaining diversity across generated tasks, we introduce a task scheduler that guides workflow sampling and task generation for each new batch.

Let $u_t(s)$ denote the cumulative usage of scenario s before batch t, with mean $\bar{u}_t$ and standard deviation σ_t, and let $N = |\mathcal{S}|$ be the number of available atomic scenarios. Scenario coverage is

measured by

$$\rho_t = \frac{|\{s : u_t(s) > 0\}|}{N} \tag{4}$$

Usage uniformity is measured by normalized entropy:

$$U_t = -\frac{1}{\log N} \sum_{s:u_t(s)>0} \frac{u_t(s)}{\sum_{s'} u_t(s')} \log \frac{u_t(s)}{\sum_{s'} u_t(s')}, \tag{5}$$

where larger U_t indicates more uniform scenario usage.

To measure structural redundancy, each workflow π_q is mapped to a canonical signature $\mathrm{sig}(\pi_q)$ that captures its ordered scenarios and typed edges. Let Σ_t be the set of unique signatures observed before batch t, and let n_t be the number of workflows accepted before that batch. For $n_t > 0$, we define redundancy as

$$R_t = 1 - \frac{|\Sigma_t|}{n_t} \tag{6}$$

where a smaller R_t indicates higher workflow diversity.

The scheduler prioritizes uncovered and underused scenarios ($u_t(s) < \bar{u}_t$), while down-weighting overused ones satisfying $u_t(s) > \max(2\bar{u}_t, \bar{u}_t + \sigma_t)$, unless required for workflow validity. Workflows with previously seen signatures are also down-weighted to encourage structural diversity. All statistics and priorities are updated after each batch.

Workflow Structural Validity. To ensure that sampled complex-task workflows can be executed correctly, BLUE introduces a workflow validation mechanism. A sampled workflow $\pi_q = (s_{i_1}, \ldots, s_{i_L})$ is valid only if each consecutive pair $(s_{i_k}, s_{i_{k+1}})$ is connected by a directed relation in G and the resulting dependency graph is acyclic after excluding FEEDBACK edges. FEEDBACK relations are exempt because they may represent intentional iteration, whereas cycles formed by other relations imply invalid circular dependencies. BLUE detects such cycles using a three-color DFS (Appendix B) and discards invalid workflows.

To further improve the quality of generated tasks and support multi-turn evaluation, BLUE applies three post-processing steps to generated tasks: Quality Validate, Query Refine and Query Compress.

Quality Validate and Refine. For Quality Validate, BLUE checks whether each task is consistent with the sampled workflow and data context. It rejects tasks only when they contain clear errors, such as invalid data references, missing deliverables, inconsistent dependencies, or violated domain constraints. During refinement, valid tasks are polished to improve clarity and conciseness while preserving all evaluation-relevant requirements.

Query Compress. To support multi-turn evaluation, BLUE transforms a complete task q_{f} into a shorter, underspecified request q_{b}.

The brief version keeps the main objective but omits details that the agent must recover through clarification, such as data paths, parameter values, output filenames, and deliverable requirements.

BLUE checks that q_{b} still reflects the original objective while leaving out the intended details. Each task can therefore be used for both single-turn evaluation with q_{f} and multi-turn evaluation starting from q_{b}. Figure 3 shows an example.

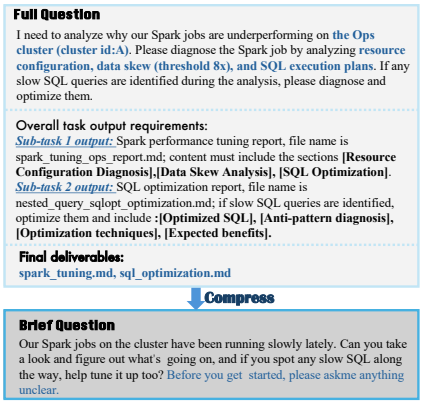


Figure 3: Example of compression

3.3 METRICS

Evaluation mirrors the task hierarchy, providing both end-to-end task scores and scenario-level milestone scores. For a complex task q with scenarios $\mathcal{S}_q = \{s_1, \ldots, s_m\}$, BLUE evaluates deterministic metrics using a rule-based judge and open-ended properties using an LLM judge.

3.3.1 RULE-BASED JUDGE AND LLM JUDGE

Rule-based Judge We introduce a customized Test Case Agent that parses each atomic evaluation into structured test rules, including expected outputs, acceptance conditions, required artifacts, and failure criteria. It then composes these rules into scenario-level test cases. Let $\mathcal{J}_i$ denote the rules for scenario s_i, with obtained score g_{ij} and maximum score $g_{ij}^{\max}$ for each rule $j \in \mathcal{J}_i$. The scenario- and task-level scores are $S_{\text{rule}}(s_i) = \frac{\sum_{j\in\mathcal{J}_i} g_{ij}}{\sum_{j\in\mathcal{J}_i} g_{ij}^{\max}}$, $S_{\text{rule}}(q) = \sum_{i=1}^{m} w_i S_{\text{rule}}(s_i)$, where $w_i \geq 0$ and $\sum_{s_i\in\mathcal{S}_q} w_i = 1$. Scenario weights are uniform by default and can be configured according to scenario importance.

LLM Judge For evaluation criteria that are difficult to formalize with deterministic rules, such as analytical logic, report completeness, and reasoning depth, we introduce an LLM judge (Zheng et al., 2023; Liu et al., 2023). Each evaluation dimension $d \in \mathcal{D}_i$ receives an integer score $z_{i,d} \in \{0, \ldots, 5\}$, normalized as $\hat{z}_{i,d} = z_{i,d}/5$. The scenario- and task-level scores are $S_{\text{llm}}(s_i) = \sum_{d\in\mathcal{D}_i} \lambda_{i,d}\hat{z}_{i,d}$, $S_{\text{llm}}(q) = \sum_{i=1}^{m} w_i S_{\text{llm}}(s_i)$, where $\lambda_{i,d} \geq 0$, $\sum_{d\in\mathcal{D}_i} \lambda_{i,d} = 1$, and $\sum_{s_i\in\mathcal{S}_q} w_i = 1$. Dimension and scenario weights can be configured by domain. The full dimension list and anchored scoring protocol are provided in Appendix D.

3.3.2 EVALUATION

We evaluate not only single-turn interactions with fully specified tasks, but also multi-turn interactions that start from underspecified user intent.

Single-turn evaluation provides q_{f} directly, whereas multi-turn evaluation starts from q_{b} and allows the agent to recover omitted requirements by querying an answer proxy agent. The proxy agent simulates user interaction and is restricted to answering with information contained in q_f.

For multi-turn evaluation, we additionally report an interaction quality score, which includes dialogue efficiency, clarification quality and requirement understanding, to assess how effectively the agent clarifies and understands missing requirements. The details are shown in Appendix D.

3.4 BLUE-BENCH

To demonstrate the generator, we use BLUE to construct a new benchmark, **BLUE-Bench**, which covers representative capabilities of coding agents in data science, SQL intelligence, cluster management & operations, multimodal processing, and visualization. We define three difficulty levels by sub-task type: Easy tasks mainly involve retrieval and simple processing operations, such as generating a simple HTML file. Medium tasks require end-to-end execution and verification, such as executing SQL. Hard tasks involve multiple execution-and-verification steps together with analytical reasoning. Details are provided in Appendix A.

Running BLUE on these inputs yields the BLUE-Bench instance used throughout our experiments, comprising **200 complex tasks** across three difficulty levels (**54 easy / 84 medium / 62 hard**). The underlying data resources are described in Appendix E, and the generator configuration is provided in Appendix F.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

In BLUE-Bench, we evaluate five agents harnesses: Claude Code, Codex, CodeBuddy, OpenCode, and Hermes, each paired with three foundation models: DeepSeek-v4-pro, GLM-5, and Kimi-k2.6. Detailed configurations are provided in Appendix F. We analyze the benchmark from five aspects: overall results, analysis of the BLUE framework, human and LLM judge agreement, benchmark evolution, and case studies comparing BLUE with related work.

Table 1: Overall results. **Bold** and underlined values indicate the best and second-best performance for each evaluation setting, respectively. The Overall score is computed using per-question aggregation, combining available rule-based and LLM judge scores.

Agent	Model	Single-Turn Evaluation							Multi-Turn Evaluation						
		Easy		Medium		Hard		Overall	Easy		Medium		Hard		Overall
		Rule	LLM	Rule	LLM	Rule	LLM		Rule	LLM	Rule	LLM	Rule	LLM	
Claude Code	DeepSeek-v4-pro	71.36	**65.99**	57.10	**50.50**	57.38	**46.14**	56.68	63.24	**64.73**	46.66	**50.16**	40.75	**45.52**	45.40
	GLM-5	77.30	59.48	67.04	46.69	60.98	43.30	60.30	**74.77**	57.08	56.70	46.41	49.03	43.01	**52.36**
	Kimi-k2.6	70.99	62.38	61.93	47.14	60.79	44.85	57.38	53.89	60.48	46.70	47.50	43.02	43.75	43.04
Codex	DeepSeek-v4-pro	62.08	62.14	61.01	47.56	60.10	43.79	54.96	53.62	58.89	46.06	46.93	**50.01**	43.97	46.10
	GLM-5	76.34	59.20	69.06	45.60	61.22	42.60	60.37	65.36	52.54	50.38	45.12	42.57	41.48	47.21
	Kimi-k2.6	68.01	61.04	63.22	46.68	59.23	42.71	56.61	57.07	61.10	49.37	46.72	40.75	42.81	45.08
CodeBuddy	DeepSeek-v4-pro	73.83	63.44	68.01	48.52	61.29	45.05	60.38	57.65	60.54	52.81	47.59	40.59	43.63	45.87
	GLM-5	73.32	57.08	65.90	45.07	**62.47**	43.81	59.15	64.58	57.77	51.57	45.58	40.08	41.74	45.76
	Kimi-k2.6	61.99	59.72	62.77	46.96	60.85	43.40	55.43	69.11	59.03	51.05	47.42	49.75	42.54	50.07
OpenCode	DeepSeek-v4-pro	**78.92**	63.57	63.96	47.57	61.82	43.99	**60.51**	69.01	62.21	**59.66**	49.09	42.92	43.69	51.08
	GLM-5	77.56	58.36	68.72	45.34	61.30	42.94	60.46	67.33	60.88	51.16	46.08	46.41	42.75	47.95
	Kimi-k2.6	70.70	59.82	68.03	46.99	61.81	43.49	58.87	63.34	60.96	53.23	46.08	46.21	43.52	48.11
Hermes	DeepSeek-v4-pro	68.78	63.18	49.43	49.25	50.36	45.38	51.22	53.69	64.48	44.21	50.06	32.24	43.29	41.32
	GLM-5	68.86	60.12	64.91	46.71	60.56	43.18	58.27	50.54	60.84	40.92	44.95	41.14	42.97	40.25
	Kimi-k2.6	73.79	61.91	**69.72**	46.95	59.44	44.06	59.85	57.53	58.72	52.47	47.98	42.77	43.85	45.45

4.2 Main Results

We evaluate different Agent and LLM combinations across single-turn and multi-turn settings and varying task difficulty levels. Table 1 reports the overall results. Overall, Claude Code + GLM-5 achieves the best combined performance across single-turn and multi-turn evaluation, followed by OpenCode + DeepSeek-v4-pro. We observe four main findings:

(1) Agent harnesses matter more in multi-turn settings. The performance gap across agent harness grows from 3.5 points in single-turn evaluation to 6.71 in multi-turn evaluation. In contrast, the gap across foundation models shrinks from 2.96 to 0.75 points. This suggests that for complex and underspecified tasks, the agent harnesses plays a larger role than foundation model.

(2) Rule-based evaluation better reflects actual task completion. From single-turn to multi-turn evaluation, rule-based scores drop much more than LLM-judge scores, especially on hard tasks. This suggests that LLM judges may reward well-written outputs even when the task is not fully completed, while rule-based evaluation directly checks whether specific requirements are satisfied. Therefore, rule-based evaluation should receive more weight for tasks with clear execution requirements.

(3) Good interaction does not guarantee task success. We find Codex + Kimi-k2.6 achieves the highest interaction score 97.31 as shown in Figure 4, but achieves only 45.08 overall in the multi-turn setting. This shows that good interaction alone does not guarantee task success; agents must also reliably execute the task after good interaction.

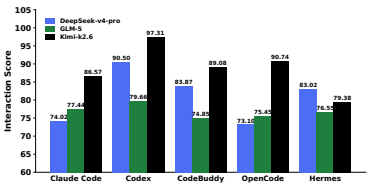


Figure 4: Interaction quality.

(4) Different configs have very different multi-turn robustness. When moving from clear single-turn tasks to underspecified multi-turn tasks, all systems drop, but by very different amounts. The drop ranges from 5.36 points for CodeBuddy + Kimi-k2.6 to 18.02 points for Hermes + GLM-5. This shows that strong single-turn performance does not always transfer to multi-turn settings, where agents also need to ask for missing information and keep track of it across turns.

4.3 BLUE Framework Analysis

We analyze how the key components of BLUE affect the quality of generated tasks from two aspects: (1) the realism and coherence of complex tasks, which depend on the reliability of relations in the

Table 2: Overall multi-turn conversation cost. **Avg / P95 / Max** are mean, 95th-percentile, and worst-case turns per task.

Agent	Model	Avg	P95	Max
Claude Code	DeepSeek-v4-pro	2.31	5.05	9
	GLM-5	1.84	3.00	4
	Kimi-k2.6	2.20	5.15	7
Codex	DeepSeek-v4-pro	**1.15**	**2.00**	3
	GLM-5	1.66	3.00	4
	Kimi-k2.6	1.34	**2.00**	3
CodeBuddy	DeepSeek-v4-pro	1.66	**2.00**	6
	GLM-5	2.02	3.00	5
	Kimi-k2.6	1.75	3.00	5
OpenCode	DeepSeek-v4-pro	1.50	2.05	4
	GLM-5	1.78	3.00	4
	Kimi-k2.6	1.54	**2.00**	3
Hermes	DeepSeek-v4-pro	1.39	**2.00**	3
	GLM-5	1.58	**2.00**	**2**
	Kimi-k2.6	1.85	3.00	8

Table 3: Ablation of Scenario Graph Construction.

Method	Precision ↑	Recall ↑	F1 ↑
w/o Check	28.6	57.1	38.1
w/o Voting	25.0	42.9	31.6
BLUE (full)	**50.0**	**57.1**	**53.3**

Table 4: Ablation of the task scheduler.

Method	Coverage ↑	Uniformity ↑	Redundancy ↓
w/o Scheduler	0.491	0.755	0.380
BLUE (full)	**0.965**	**0.970**	**0.100**

Table 5: Human–LLM Judge Agreement.

Dimension	Avg. LLM	Avg. Human	$\Delta \downarrow$	$r \uparrow$
AnswerQuality	2.467	2.569	0.102	0.934
InteractionQuality	4.248	4.000	0.248	0.938

Table 6: Human Preference Study.

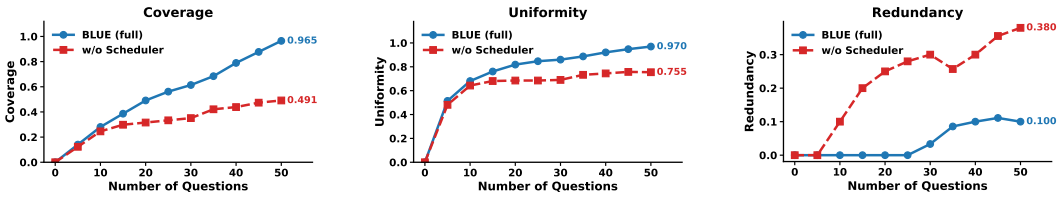
Method	Directed	AutoBencher	TaskCraft	BLUE
Avg. Score ↑	65.50	56.00	35.67	**92.83**
Avg. Rank ↓	2.38	2.76	3.57	**1.29**

scenario graph; and (2) the diversity and coverage of large-scale task generation, which are mainly controlled by the task scheduler.

Accuracy of inter-scenario relations. We evaluate scenario graph construction using Precision, Recall, and F1 of the inferred edge relations. We manually annotate 10 directed scenario relations as ground truth. As shown in Table 3, removing relation check reduces F1 from 53.3 to 38.1, while removing multi-round voting reduces it to 31.6. Relation check mainly improves Precision by filtering spurious edges, while multi-round voting makes relation inference more stable.

Quality of large-scale task generation. We evaluate scale generation from three aspects: scenario coverage, usage uniformity, and workflow redundancy. The metric definitions are provided in Section 3.2.3.

To evaluate the effect of the task scheduler, we generate 50 tasks with and without scheduler guidance. Compared with greedy LLM sampling, the task scheduler increases scenario coverage from 49.1% to 96.5%, improves usage uniformity from 0.755 to 0.970, and reduces workflow redundancy from 0.380 to 0.100. Figure 5 shows how these metrics change during generation.



(a) Coverage. (b) Uniformity. (c) Redundancy.

Figure 5: Task Scheduler Analysis.

4.4 Human–LLM Judge Agreement Analysis

BLUE uses LLM judges to evaluate Answer Quality and Interaction Quality. To validate their reliability, we randomly sample 50 task executions and have three expert evaluators independently score them using the same rubrics. We compare the averaged human scores with the corresponding LLM-judge scores.

As shown in Table 5, the average absolute human–LLM score gap is only $\Delta = 0.102$ for Answer Quality and $\Delta = 0.248$ for Interaction Quality. Moreover, the high Pearson correlation ($r > 0.9$) indicates strong agreement at the individual-task level. These results support the use of LLM judges as a reliable proxy for human evaluation at benchmark scale.

4.5 BENCHMARK EVOLVING

We study how the size of the atomic-task pool affects the semantic diversity of generated complex questions to show benchmark evolution. We fix the number of generated questions at 200 and vary the number of atomic tasks as $k \in \{10, 40, 70, 100\}$. All generated questions are embedded using `text-embedding-3-small` and projected with PCA.

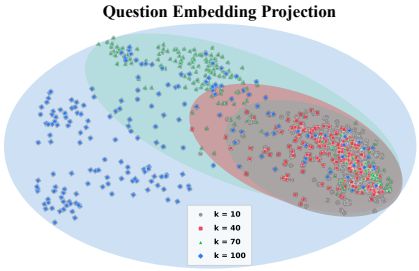


Figure 6: Semantic coverage.

As shown in Figure 6, the semantic distribution expands consistently as k increases, indicating that a larger atomic-task pool leads to more diverse generated questions. Interestingly, the expansion from $k = 70$ to $k = 100$ is substantially larger than that from $k = 10$ to $k = 40$, suggesting that the semantic space grows increasingly rapidly as more atomic tasks are introduced. This trend reflects the compositional nature of BLUE: newly introduced capabilities can interact not only with one another, but also with previously available capabilities, giving rise to a rapidly expanding space of task compositions. Consequently, BLUE can generate increasingly diverse and complex tasks with lower additional human effort, enabling the benchmark to scale more efficiently as the capability pool grows.

4.6 COMPARISON WITH RELATED METHODS

Human preference study. We compare BLUE with three representative task-generation baselines: AutoBencher, TaskCraft, and direct LLM generation. We construct 50 comparison cases and ask three expert evaluators to independently rank the four generated tasks from 1 (best) to 4 (worst). Rankings are further mapped to scores of 100, 75, 50, and 25, respectively.

As shown in Table 6, BLUE achieves the highest average score of 92.83 and the best average rank of 1.29, outperforming direct LLM generation, AutoBencher, and TaskCraft. This indicates a clear human preference for tasks generated by BLUE.

Case Study. Figure 7 compares a Spark performance diagnosis task generated by BLUE with three baselines. BLUE combines resource configuration, data-skew analysis, SQL-plan diagnosis, optimization, and explicit deliverables into a complete workflow, together with rule-based and LLM-based evaluation. Direct LLM generation covers a narrower diagnosis scenario with coarse report-level evaluation, while TaskCraft and AutoBencher generate more isolated analysis or knowledge questions with simpler evaluation targets in question-answer format. This case demonstrates the ability of BLUE to compose multiple capabilities into coherent tasks with concrete outputs and verifiable evaluation criteria.

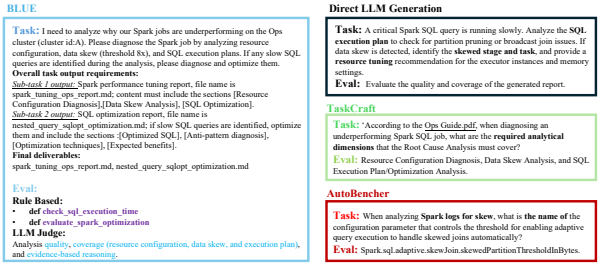


Figure 7: Tasks and evaluation generated by BLUE and three baselines.

5 CONCLUSION

We introduce **BLUE**, an evolvable benchmark generator that expands the evaluation space through capability composition. Starting from user-provided atomic tasks and evaluations, BLUE composes them into coherent and diverse multi-capability tasks together with milestone-level and end-to-end evaluations, supporting both single-turn and multi-turn evaluation. Using BLUE, we construct **BLUE-Bench** with 200 complex tasks across three difficulty levels and evaluate five agents paired with three foundation models. Furthermore, we conduct detailed analyses of BLUE, evaluation reliability, benchmark evolution, and generated task quality, demonstrating the effectiveness of BLUE for automated construction of reliable benchmarks.

AI USE STATEMENT

In this work, we used generative AI tools to assist in the construction of the benchmark and conducted evaluations of generative AI systems using the resulting benchmark. Additionally, we used generative AI tools to aid in polishing the language and improving the clarity of the manuscript. We reviewed all AI-assisted content and take full responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work involves human evaluation of generated benchmark tasks and agent outputs. Specifically, expert evaluators are asked to assess task quality, rank generated tasks, and score agent outputs using predefined evaluation rubrics. The evaluation does not involve personally identifiable or sensitive data, and the study does not require evaluators to disclose personal information. The benchmark and experiments primarily concern the evaluation of large language model agents in coding- and data-related tasks. We do not identify significant ethical risks specific to the proposed benchmark-generation framework beyond those generally associated with the use and evaluation of large language models, including potential errors or biases introduced by LLM-based task generation and judging. We mitigate these risks through rule-based evaluation, human evaluation, and analysis of human–LLM judge agreement. We take responsibility for the design, analysis, and reporting of all experiments.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of BLUE, the task-generation pipeline, evaluation protocol, baselines, and experimental settings in the main paper and appendix. You can find it in the URL https://anonymous.4open.science/r/BLUE-ICLR. And appendix further documents the benchmark construction process, task specifications, evaluator design, and implementation details needed to reproduce our experiments. We will release the code, task definitions, evaluation scripts, and configuration files used in our experiments to facilitate reproducibility.

REFERENCES

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. doi: 10.48550/arXiv.2107.03374. URL https://arxiv.org/abs/2107.03374.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a Generalist Agent for the Web. In *Advances in Neural Information Processing Systems*, volume 36, pp. 28091–28114. Curran Associates, Inc., 2023. doi: 10.52202/075280-1220. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/5950bf290a1570ea401bf98882128160-Paper-Datasets_and_Benchmarks.pdf.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. 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*, pp. 11642–11662. PMLR, 2024. URL https://proceedings.mlr.press/v235/drouin24a.html.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visually grounded web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.50. URL https://aclanthology.org/2024.acl-long.50/.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-Tau Yih, Daniel Fried, Sida Wang, and Tao Yu. DS-1000: A natural and reliable benchmark for data science code generation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 18319–18345. PMLR, 2023. URL https://proceedings.mlr.press/v202/lai23b.html.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can LLM already serve as a database interface? a BIg bench for large-scale database grounded text-to-SQLs. In *Advances in Neural Information Processing Systems*, volume 36, 2023a. doi: 10.52202/075280-1835. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/83fc8fab1710363050bbd1d4b8cc0021-Abstract-Datasets_and_Benchmarks.html.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A Comprehensive Benchmark for Tool-Augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3102–3116, Singapore, 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.187. URL https://aclanthology.org/2023.emnlp-main.187/.

Xiang Lisa Li, Farzaan Kaiyom, Evan Zheran Liu, Yifan Mai, Percy Liang, and Tatsunori Hashimoto. AutoBencher: Towards declarative benchmark construction. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ymt4crbbXh.

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, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2511–2522, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153/.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An Analytical Evaluation Board of Multi-turn LLM Agents. In *Advances in Neural Information Processing Systems*, volume 37, pp. 74325–74362. Curran Associates, Inc., 2024. doi: 10.52202/079017-2365. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/877b40688e330a0e2a3fc24084208dfa-Paper-Datasets_and_Benchmarks_Track.pdf.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. In *The Twelfth International Conference on Learning Representations*, pp. 9695–9717, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/28e50ee5b72e90b50e7196fde8ea260e-Paper-Conference.pdf.

Divyanshu Saxena, Rishikesh Maurya, Xiaoxuan Ou, Gagan Somashekar, Shachee Mishra Gupta, Arun Iyer, Yu Kang, Chetan Bansal, Aditya Akella, and Saravan Rajmohan. Continuous benchmark generation for evaluating enterprise-scale LLM agents. In *Proceedings of the 3rd International Workshop on Large Language Models for Code*, pp. 1–5. Association for Computing Machinery, 2026. doi: 10.1145/3786181.3788708. URL https://doi.org/10.1145/3786181.3788708.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551. Curran Associates, Inc., 2023. doi: 10.52202/075280-2997. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/d842425e4bf79ba039352da0f658a906-Paper-Conference.pdf.

Dingfeng Shi, Jingyi Cao, Qianben Chen, Weichen Sun, Weizhen Li, Hongxuan Lu, Fangchen Dong, Tianrui Qin, King Zhu, Minghao Liu, Yuchen Eleanor Jiang, Jian Yang, Ge Zhang, Jiaheng Liu, Changwang Zhang, Jun Wang, and Wangchunshu Zhou. TaskCraft: Automated generation of agentic tasks. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=UJFCyrYM1V.

Transaction Processing Performance Council. TPC-DS, n.d. URL https://www.tpc.org/tpcds/. Accessed September 20, 2026.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A Controllable World of Apps and People for Benchmarking Interactive Coding Agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

Siyuan Wang, Zhuohan Long, Zhihao Fan, Xuanjing Huang, and Zhongyu Wei. Benchmark self-evolving: A multi-agent framework for dynamic LLM evaluation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 3310–3328, Abu Dhabi, UAE, 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.223/.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Xin Guo, Dingwen Yang, Chenyang Liao, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. AgentGym: Evaluating and Training Large Language Model-based Agents across Diverse Environments. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27914–27961, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.1355. URL https://aclanthology.org/2025.acl-long.1355/.

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. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-1650. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5d413e48f84dc61244b6be550f1cd8f5-Abstract-Datasets_and_Benchmarks_Track.html.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757. Curran Associates, Inc., 2022. doi: 10.52202/068431-1508. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/82ad13ec01f9fe44c01cb91814fd7b8c-Paper-Conference.pdf.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b126cc38b8638e07bef37e7b2bb72bf-Abstract-Conference.html.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3911–3921, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1425. URL https://aclanthology.org/D18-1425/.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-2020. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

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. WebArena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

APPENDIX OVERVIEW

The appendix follows the same dependency order as the BLUE pipeline: atomic tasks are composed into atomic scenarios; atomic scenarios become nodes in the scenario graph; the task scheduler samples valid workflows from that graph; and the resulting complex tasks are paired with hierarchical evaluators. Data, implementation, and supplementary results are documented afterward. Specifically:

- **Appendix A** defines atomic-task inputs, the three cues for composing atomic scenarios, the difficulty taxonomy, and the atomic-task inventory (Table 7).
- **Appendix B** defines the five typed relations (Table 8), adversarial edge verification, and workflow-validity checks (Algorithm 1).
- **Appendix C** gives the coverage-aware batched generation algorithm (Algorithm 2) and its scenario- and domain-level balancing rules.
- **Appendix D** details the anchored protocols for the LLM judge and the multi-turn interaction judge (Table 9).
- **Appendix E** documents the BLUE-Bench data resources, including the extended TPC-DS schema and the DS-1000/BIRD/Spider task adaptation.
- **Appendix F** lists implementation details, cluster environment, and evaluated agent backends (Tables 10–12).
- **Appendix G** reports supplementary experimental results, including per-difficulty dialogue-cost statistics (Table 14).
- **Appendix H** provides a consolidated notation reference (Table 15).

A ATOMIC TASKS AND SCENARIO COMPOSITION

This section formalizes the first transition in the BLUE hierarchy, from user-provided atomic tasks to the atomic scenarios that serve as graph nodes. It then defines the task-difficulty taxonomy and documents the atomic-task inventory used to instantiate BLUE-Bench.

From atomic tasks to atomic scenarios. An atomic task τ_i is the smallest executable and evaluable unit supplied to BLUE. The Atomic Scenario Agent organizes the atomic-task set $\mathcal{T}$ into ordered groups $\mathcal{T}_s$, one for each atomic scenario $s \in \mathcal{S}$, while ensuring that every atomic task is covered. Each resulting scenario must be independently executable and independently verifiable, so that it can serve both as a semantic node in the scenario graph and as a milestone in the composed evaluator.

Composition rule. The Atomic Scenario Agent uses exactly the three composition cues introduced in the main paper:

- **Shared object:** the tasks read, update, or analyze the same business or data object.
- **Data dependency:** an output of one task is required as an input to another task.
- **Business process:** the tasks are successive stages of the same operational process.

A group is formed only when at least one cue supports a plausible local workflow. Data dependency also determines execution order whenever one task consumes another's output; otherwise, the business process supplies the order. Tasks supported only by superficial lexical overlap remain separate. This default-to-separate policy prevents noisy low-level associations from becoming graph nodes.

Illustrative example. Atomic tasks such as `ViewYarnLog`, `ViewSparkLog`, and `ViewResourceMonitor` may be composed into a *Job Failure Diagnosis* scenario because they inspect the same failed job and jointly support a diagnosis. By contrast, `RerunJob` is kept in a separate recovery scenario when rerunning is contingent on that diagnosis. The scenario graph can then express the cross-scenario dependency explicitly instead of hiding it inside one oversized node. Thus, atomic scenarios capture local coherence, whereas graph edges capture dependencies between local workflows.

Task-difficulty definition. Task difficulty in BLUE is not determined directly by the number of atomic scenarios, workflow length, or number of domains. It is determined by the types of constituent atomic tasks and their required verification depth. We define three levels:

- **Easy (Level 1) — Information Retrieval.** Tasks that require only querying, retrieval, or API invocation, followed by simple aggregation or formatting.
- **Medium (Level 2) — Execution and Verification.** Tasks that contain at least one sub-task requiring genuine end-to-end execution and result verification, e.g., NL2Code, NL2SQL, or SQL optimization.
- **Hard (Level 3) — Composite Analysis.** Tasks that combine information-retrieval subtasks, multiple execution-verification sub-tasks, and analytical reasoning within a single workflow.

The difficulty label of a complex task is therefore distinct from the structural-validity conditions applied to its workflow. Appendix B defines those conditions, and Appendix C explains how valid workflows are sampled.

Atomic-task inventory. Table 7 summarizes the 121 atomic tasks organized by domain and subdomain that are currently covered by the BLUE generator. Each row reports the count of atomic tasks in a sub-domain together with representative capabilities. A new capability is introduced by adding an atomic task to this inventory and supplying its paired atomic evaluation.

B Scenario Graph Construction

This section formalizes the second transition in BLUE, from atomic scenarios to a typed scenario graph. We first define edge semantics, then describe how inferred edges are verified, and finally state the conditions under which a sampled workflow is structurally valid.

Relation types. Let $\mathcal{R}$ denote the five directed relation types in Table 8. For an ordered pair (s_i, s_j), the relation describes how s_i constrains the execution of s_j. If relation inference supports more than one type for the same ordered pair, BLUE retains the highest-priority type in the table; it does not add parallel edges.

Adversarial edge verification. Multi-round voting produces a candidate edge $e^*_{ij} = (s_i, s_j, r^*_{ij}, c^*_{ij}, h^*_{ij})$. As described in the main paper, a candidate with $c^*_{ij} < \tau_c$ is passed to a reverse verifier that sees the two scenario descriptions and r^*_{ij}, but not c^*_{ij} or h^*_{ij}. It attempts to falsify the candidate along four dimensions:

- **Business plausibility:** whether a meaningful business-logic relation connects the scenarios.
- **Relation-type consistency:** whether r^*_{ij} matches the relation expressed by their descriptions.
- **Directional validity:** whether the dependency runs from s_i to s_j, rather than in the reverse direction.
- **Competing explanation:** whether another type, direction, or a common-factor explanation is more plausible.

An edge enters E only if it meets the voting threshold τ_n and, when triggered, passes this verification. Neither confidence nor evidence is recomputed by the verifier.

Workflow structural validity. For a sampled workflow $\pi_q = (s_{i_1}, \ldots, s_{i_L})$, let $V_q = \mathcal{S}_q$ be its scenario set and let E_q be the edges among those scenarios used by the sampled dependency structure. The workflow is accepted only if (i) every consecutive pair $(s_{i_k}, s_{i_{k+1}})$ is supported by a directed edge in E_q, (ii) its entry scenario is executable without an unmet predecessor, and (iii) the non-feedback graph $G^-_q = (V_q, E^-_q)$ is acyclic, where

$$E^-_q = \{e \in E_q : r(e) \neq \textsc{Feedback}\}.$$

Table 7: Atomic tasks covered by the benchmark generator, grouped by domain and sub-domain.

Domain	Sub-domain	# Tasks	Representative Capabilities
Multimodal Processing	Text Processing	20	data loading, length/regex filtering, keyword screening, language detection, translation, fuzzy deduplication, classification, PII masking, structured extraction, summarization, perplexity filtering, unsafe-content detection, sentiment analysis, copy generation, grammar correction, similarity, embedding
	Image Processing	3	aesthetic scoring, NSFW compliance check, image–text consistency matching
Data Science	Data Science Analysis	20	data-science analysis tasks, including DS-1000-style code generation as well as broader analytical operations
Cluster Management & Operations	Cluster Management	48	instance/node queries, resource configuration, scaling price estimation, monitoring overview, HDFS storage analysis, YARN job/queue management, multi-engine query (Spark / Hive / Impala / Kyuubi / StarRocks / Trino), HBase table management, auto-scaling policies, bootstrap scripts, workflow scheduling, user permissions, configuration groups
	Intelligent Ops	14	cluster insight analysis, HDFS / StarRocks / YARN fault diagnosis, Spark data-skew and resource tuning, SQL execution-plan optimization, error-code classification, service/node time-series monitoring, TSDB low-level queries, slow-query retrieval, EMR log retrieval, config/log file browsing
SQL Intelligence	NL2SQL	5	natural-language to complex analytical SQL
	SQL Optimization	4	SQL rewriting and optimization
Visualization	Report Generation	2	structured annual-report generation, marketing-campaign analytics
	Chart & Dashboard	3	HTML dashboards, responsive front-end, AntV charts (line / bar / pie / radar / heatmap, etc.)
	Diagram	2	drawio flowchart generation, general architecture diagrams
Total		**121**	

Table 8: Scenario relation types used in graph construction.

Priority	Relation Type	Formal Semantics
1	FEEDBACK	s_i produces an evaluation or corrective signal that causes s_j to be modified or re-executed in a subsequent iteration.
2	VALIDATION	The primary objective of s_j is to verify or evaluate the quality or correctness of the result produced by s_i.
3	CONDITIONAL DEPENDENCY	The state or result of s_i determines whether s_j is activated and executed.
4	RESULT DEPENDENCY	s_j is executed and directly consumes or reuses a concrete result produced by s_i.
5	CONTROL DEPENDENCY	s_j is executed without directly consuming the result of s_i, but must wait until s_i has completed.

Condition (iii) permits intentional refinement loops represented by FEEDBACK, while rejecting circular execution dependencies of every other type. Relations are not transitively closed: an indirect path never creates a new direct edge. Algorithm 1 implements the acyclicity check with a three-color depth-first search.

Algorithm 1 Acyclicity verification for a sampled workflow.

```
function IsValidDAG(V_q, E_q)
    E_q^- ← {e ∈ E_q : r(e) ≠ Feedback}
    color[s] ← White for every s ∈ V_q
    for all s ∈ V_q do
        if color[s] = White and HasCycle(s, E_q^-) then
            return False
        end if
    end for
    return True
end function

function HasCycle(s, E_q^-)
    color[s] ← Gray
    for all (s, s', ·, ·, ·) ∈ E_q^- do
        if color[s'] = Gray then return True                ▷ back edge
        else if color[s'] = White and HasCycle(s', E_q^-) then
            return True
        end if
    end for
    color[s] ← Black
    return False
end function
```

C Task Scheduler and Batched Generation

This section formalizes the third transition in BLUE, from the scenario graph to complex tasks. The scheduler samples workflows in batches, prioritizes scenario coverage and balanced usage, discourages repeated structures, and passes only structurally valid workflows to task generation.

Batched generation with coverage control. Algorithm 2 uses the notation of the main paper throughout. The target benchmark size is N_q, and n_g is the maximum number of tasks generated per batch. Before batch t, $\mathcal{Q}_{<t}$ is the accepted complex-task set, $u_t(s)$ counts how often scenario s occurs in that set, and Σ_t contains the distinct workflow signatures. Duplicate signatures are down-weighted during ranking rather than prohibited outright; this matches the redundancy metric R_t and avoids stalling when the valid workflow space is small.

Domain-level balancing. Scenario-level balancing alone can still yield a skewed domain distribution: the generator may favor domains for which task construction is easy (e.g., data processing or visualization) while under-representing harder domains (e.g., operational diagnosis or SQL optimization). Let $\mathcal{S}_k^{\mathrm{dom}} \subseteq \mathcal{S}$ be the scenarios assigned to domain k. We use the inventory-proportional target and the observed usage share

$$p_k^{\mathrm{target}} = \frac{|\mathcal{S}_k^{\mathrm{dom}}|}{N}, \qquad p_{t,k}^{\mathrm{obs}} = \frac{\sum_{s \in \mathcal{S}_k^{\mathrm{dom}}} u_t(s)}{\sum_{s \in \mathcal{S}} u_t(s)},$$

and define domain deviation as

$$\Delta_{t,k} = p_{t,k}^{\mathrm{obs}} - p_k^{\mathrm{target}}.$$

The observed share is evaluated after at least one workflow has been accepted; all domains receive neutral priority in the first batch. When $\Delta_{t,k} < -0.10$, the domain is under-covered and its scenarios receive higher priority in the next batch. When $\Delta_{t,k} > 0.15$, the domain is over-covered and its scenarios are down-weighted. These thresholds are configurable; they complement, rather than replace, the scenario-level statistics ρ_t, U_t, and R_t reported in the main paper.

Algorithm 2 Coverage-aware batched task generation.

Require: Scenario graph $G = (\mathcal{S}, E)$, target size N_q, batch size n_g
Ensure: Complex-task set $\mathcal{Q}$

$\mathcal{Q}_{<0} \leftarrow \emptyset$; $u_0(s) \leftarrow 0$ for every $s \in \mathcal{S}$; $\Sigma_0 \leftarrow \emptyset$; $t \leftarrow 0$
while $|\mathcal{Q}_{<t}| < N_q$ **do**
  $b_t \leftarrow \min(n_g, N_q - |\mathcal{Q}_{<t}|)$
  $\bar{u}_t \leftarrow \mathrm{mean}_{s\in\mathcal{S}} u_t(s)$; $\sigma_t \leftarrow \mathrm{std}_{s\in\mathcal{S}} u_t(s)$
  $\mathcal{U}_t^0 \leftarrow \{s : u_t(s) = 0\}$; $\mathcal{U}_t^- \leftarrow \{s : 0 < u_t(s) < \bar{u}_t\}$
  $\mathcal{U}_t^+ \leftarrow \{s : u_t(s) > \max(2\bar{u}_t, \bar{u}_t + \sigma_t)\}$
  $\{\Delta_{t,k}\}_k \leftarrow$ DOMAINDEVIATION$(u_t, \mathcal{S})$
  $\mathcal{C}_t \leftarrow$ BUILDGUIDANCE$(\mathcal{U}_t^0, \mathcal{U}_t^-, \mathcal{U}_t^+, \{\Delta_{t,k}\}_k, \Sigma_t)$
  $\mathcal{P}_t \leftarrow$ SAMPLEWORKFLOWS$(G, \mathcal{C}_t)$
  rank $\mathcal{P}_t$ by coverage gain, usage balance, and signature novelty
  $\mathcal{B}_t \leftarrow \emptyset$
  for all $\pi \in \mathcal{P}_t$ in ranked order **do**
    if π satisfies difficulty constraints **and** ISVALIDWORKFLOW(π, G) **then**
      $q \leftarrow$ GENERATEANDVALIDATETASK(π)
      if q is accepted **then** $\mathcal{B}_t \leftarrow \mathcal{B}_t \cup \{q\}$
      end if
    end if
    if $|\mathcal{B}_t| = b_t$ **then break**
    end if
  end for
  $\mathcal{Q}_{<t+1} \leftarrow \mathcal{Q}_{<t} \cup \mathcal{B}_t$
  $u_{t+1}(s) \leftarrow u_t(s) + \sum_{q\in\mathcal{B}_t} \mathbf{1}[s \in \mathcal{S}_q]$ for every $s \in \mathcal{S}$
  $\Sigma_{t+1} \leftarrow \Sigma_t \cup \{\mathrm{sig}(\pi_q) : q \in \mathcal{B}_t\}$; $t \leftarrow t + 1$
end while
return $\mathcal{Q}_{<t}$

D HIERARCHICAL EVALUATION AND ANCHORED SCORING

This section expands the evaluation stage described in the main paper. The rule-based judge executes composed test rules and therefore requires no subjective anchors. The LLM judge scores open-ended answer-quality dimensions, while the interaction judge scores requirement elicitation in the multi-turn setting; both model-based judges use the anchored protocol below.

Alignment with the task hierarchy. For each $s_i \in \mathcal{S}_q$, the Test Case Agent composes the atomic evaluations attached to $\mathcal{T}_{s_i}$ into the rule set $\mathcal{J}_i$; open-ended criteria for the same milestone form the dimension set $\mathcal{D}_i$. The scenario scores are then aggregated into $S_{\mathrm{rule}}(q)$ and $S_{\mathrm{llm}}(q)$ using the same scenario weights w_i defined in the main paper. The full request q_{f} and brief request q_{b} share these task-performance evaluators because they represent the same complex task. Multi-turn evaluation adds S_{int} as a separate interaction score; it does not replace or alter the task-performance scores.

Global calibration principle. To reduce scoring arbitrariness, each dimension is scored on the integer scale $\{0, \ldots, 5\}$. A score of 0 denotes a missing, unusable, or entirely incorrect response. A score of 3 is the calibration point: the response is broadly correct, covers the main content, and is adequately structured. A score of 4 requires explicit positive evidence beyond that baseline, and 5 is reserved for expert-level treatment such as correct edge-case analysis or quantitative trade-off reasoning. Before assigning a score, the judge enumerates negative evidence, then positive evidence, and finally returns the score with a concise rationale.

Interaction-quality anchors. For multi-turn evaluation, BLUE separately assesses how effectively an agent elicits and understands information omitted from the brief request q_{b}. Let $\mathcal{C}_K$ be the transcript after K turns and let $\mathcal{I}_K = (q_{\mathrm{b}}, q_{\mathrm{f}}, \mathcal{C}_K)$ denote the corresponding interaction, where q_{f} is used only as the reference specification. The interaction judge scores three complementary dimensions:

Table 9: Anchors for multi-turn interaction-quality scoring.

Dimension	Weight	Scoring Anchors
Dialogue Efficiency	α_I	0: no useful clarification; 1: heavy redundancy; 3: partly mergeable questions; 5: near-optimal number of turns with a clear intent per question.
Clarification Quality	β_I	0: no relevant question; 1: misses most requirement gaps; 3: covers some key information; 5: precisely covers all critical gaps with logical progression.
Requirement Understanding	γ_I	0: no evidence of understanding; 1: substantial deviation from q_{f}; 3: main goal understood but key constraints missed; 5: full recovery and preservation of all key details.

- *Dialogue Efficiency* d_{eff}: whether necessary information is obtained without redundant interaction—penalizing both excessive turns and mergeable/repeated questions.
- *Clarification Quality* d_{clar}: whether the agent asks relevant and targeted questions that cover the requirement gaps in q_{b} (data locations, parameters, deliverables, etc.), as opposed to generic or off-topic prompts.
- *Requirement Understanding* d_{under}: whether recovered requirements are correctly interpreted and incorporated into subsequent planning and execution, judged by consistency with q_{f}.

Writing $\hat{d}_x = d_x/5$ for normalized dimension scores, the interaction score is

$$S_{\mathrm{int}}(\mathcal{I}_K) = \alpha_I \hat{d}_{\mathrm{eff}} + \beta_I \hat{d}_{\mathrm{clar}} + \gamma_I \hat{d}_{\mathrm{under}}, \qquad \alpha_I + \beta_I + \gamma_I = 1.$$

The weights are configurable; the experimental values are listed in Table 10. A score of 4 or 5 must cite concrete evidence from $\mathcal{C}_K$. Remaining silent when q_{b} is under-specified is not considered efficient: it lowers clarification quality and requirement understanding. We report S_{int} separately from task performance so that successful clarification is not conflated with successful execution. Table 9 summarizes the dimension-specific anchors.

E BLUE-BENCH DATA RESOURCES

This section documents the data sources underlying the BLUE-Bench instance used in the paper, so that the schema extensions, synthetic content, and task adaptations are fully reproducible.

Extended TPC-DS schema. BLUE-Bench uses TPC-DS (Transaction Processing Performance Council, n.d.) at scale factor 100 as its relational backbone and augments its original schema along two axes:

- *Multimodal extension.* We add one new column to each of two textual/descriptive tables—`promotion` and `reason`—that stores a URI pointing to an image asset associated with the corresponding row (e.g., a promotional banner for a `promotion` entry, or an illustrative image for a `reason` entry). All image assets are hosted in Tencent Cloud Object Storage (COS) and referenced by their COS paths, so that multimodal tasks can be evaluated end-to-end alongside standard relational queries without altering the original TPC-DS join structure.
- *Synthetic privacy-related information.* We inject randomly generated personally identifiable information (PII) into the customer-facing tables, including email addresses, phone numbers, postal addresses, and national ID numbers. All values are synthetic—no real user data is used—and are generated with locale-consistent formats so that PII-masking, regex-based extraction, and privacy-compliance tasks have realistic targets to operate on. The synthetic PII is inserted as new columns rather than modifying the original TPC-DS attributes, preserving the semantics of the base schema for standard SQL tasks.

Together, these two extensions turn TPC-DS into a heterogeneous data source that supports (i) standard SQL workloads on the unmodified subset of the schema, (ii) multimodal workloads that must

Table 10: Main configurations for benchmark generation and evaluation.

Configuration	Setting
Task-generation LLM API	Gemini-3.1-Flash-Lite
Answer-proxy agent LLM API	Gemini-3.1-Flash-Lite
LLM-judge API	Gemini-3.5-Flash
Interaction-judge API	Gemini-3.5-Flash
Graph sampling temperatures	$\{0.3, 0.5, 0.7, 0.9, 1.0\}$
Relation-inference runs	5
Consensus threshold τ_n	3
Reverse-verification threshold τ_c	0.9
Refinement length ratio	$[0.4, 1.1]$
Rule / LLM score weights	$0.5 / 0.5$
Interaction score weights $(\alpha_I, \beta_I, \gamma_I)$	$0.25 / 0.40 / 0.35$
TPC-DS scale factor	100
n_g	7

resolve image URIs and process the referenced assets, and (iii) privacy-oriented workloads that must locate, mask, or transform synthetic PII.

Task-source adaptation. Complex tasks in BLUE-Bench are grounded in the extended TPC-DS through two public task sources:

- *Data-science tasks* are sampled and adapted from DS-1000 (Lai et al., 2023). Each selected problem is re-grounded so that its input data, intermediate artifacts, and expected outputs are expressed against the extended TPC-DS schema (or the multimodal/PII columns introduced above), and its deliverables are re-annotated to match the atomic-evaluation format expected by BLUE.
- *SQL tasks* follow the formulations of BIRD (Li et al., 2023a) and Spider (Yu et al., 2018) but are re-issued against the extended TPC-DS. Question templates, difficulty levels, and evaluation criteria from the original benchmarks are preserved; only the referenced schema and gold queries are rewritten so that they execute on our TPC-DS instance.

This adaptation lets BLUE-Bench inherit the difficulty distributions and question styles of well-established benchmarks while ensuring that every task is executable on a single, shared data backbone.

F Implementation Details and Environment

This section fixes the configuration of the generator and the evaluation harness so that our experiments are fully reproducible: the LLM-API and hyper-parameter settings driving generation and scoring, the cluster environment used for data-intensive and cluster-operation tasks, and the exact versions of the five coding agents under test.

Benchmark generation and evaluation configuration. Table 10 lists the main hyper-parameters used for benchmark generation and evaluation, including the LLM APIs used by the task-generation agent, the answer proxy agent, and the LLM/interaction judges, as well as graph-construction and scoring hyper-parameters.

Cluster environment. Data-intensive and cluster-operation tasks are executed on an Elastic MapReduce (EMR) cluster on Tencent Cloud. The cluster consists of 10 nodes across four roles (Master, Core, Task, Common), providing a total of 48 vCPU cores, 168 GB of memory, and 2000 GB of data storage. The full configuration is reported in Table 11.

Evaluated coding agents. Table 12 lists the five coding agents evaluated in our experiments, together with their providers, release status (open- or closed-source), interface, and version numbers at the time of evaluation.

Table 11: Cluster node configuration used in cluster-operation experiments.

Role	#Nodes	Spec.	CPU/node	Mem/node	Sys. Disk	Data Disk	Total CPU	Total Mem	Total Data
Master	2	SA5	8 cores	32 GB	70 GB	200 GB	16 cores	64 GB	400 GB
Core	3	SA5	4 cores	16 GB	70 GB	200 GB	12 cores	48 GB	600 GB
Task	2	SA5	4 cores	16 GB	70 GB	200 GB	8 cores	32 GB	400 GB
Common	3	SA5	4 cores	8 GB	70 GB	200 GB	12 cores	24 GB	600 GB
Total	**10**	—	—	—	—	—	**48 cores**	**168 GB**	**2000 GB**

Table 12: Coding-agent backends evaluated in our experiments.

Agent	Provider	Source	Interface	Version
Claude Code	Anthropic	Closed	CLI	2.1.197
Codex CLI	OpenAI	Closed	CLI	0.142.5
CodeBuddy	Tencent	Closed	CLI	2.124.0
OpenCode	Community	Open	CLI	1.17.14
Hermes Agent	Community	Open	CLI	0.18.0

G Supplementary Experimental Results

This section collects experimental results deferred from the main paper due to space; it contains the difficulty-stratified breakdown of multi-turn dialogue cost (Table 14).

Per-difficulty dialogue cost. Table 14 provides a difficulty-stratified breakdown of the multi-turn dialogue cost summarized in Table 2 of the main paper. For each of Easy / Medium / Hard, we report the mean turns (Avg), 95th-percentile turns (P95, tail cost), and worst-case turns (Max), together with the aggregate over all difficulties. Dialogue cost is not a monotonic proxy for task difficulty: its profile depends strongly on the agent–model configuration. For example, Codex + DeepSeek-v4-pro remains nearly flat across the three levels, whereas Claude Code + DeepSeek-v4-pro has pronounced Medium/Hard tails and Hermes + Kimi-k2.6 reaches a Hard-task maximum of 8 turns. This variation motivates reporting dialogue cost separately from both task difficulty and task-performance scores.

H Notation

Table 15 is the canonical notation reference for both the main paper and the appendix. A calligraphic capital denotes a set; a lowercase symbol denotes one object or scalar; subscript t denotes the state immediately before generation batch t; and a superscript $*$ denotes the relation selected by multi-round voting.

Table 15: Canonical notation used throughout the paper.

Symbol	Meaning
Composition hierarchy	
$\tau_i, \mathcal{T}$	atomic task i; set of all atomic tasks
$s_i, \mathcal{S}$	atomic scenario i; set of all atomic scenarios
$\mathcal{T}_s$	ordered atomic tasks composed into scenario s
$q, \mathcal{Q}$	a complex task; a set of complex tasks
$\mathcal{S}_q$	set of atomic scenarios composing complex task q
$\pi_q = (s_{i_1}, \ldots, s_{i_L})$	ordered workflow sampled for complex task q
$q_{\mathrm{f}}, q_{\mathrm{b}}$	full task specification; compressed, underspecified request
Scenario graph and relation inference	
$G = (\mathcal{S}, E)$	typed scenario graph with atomic-scenario nodes and directed edges
$\mathcal{R}$	set of the five scenario-relation types
$e_{ij} = (s_i, s_j, r_{ij}, c_{ij}, h_{ij})$	edge from s_i to s_j with relation, confidence, and evidence
$r(e)$	relation type carried by edge e
M, v	number of relation-inference rounds; round index

Table 15 (continued)

Symbol	Meaning
$r_{ij}^{(v)}, c_{ij}^{(v)}, h_{ij}^{(v)}$	relation, confidence, and evidence predicted in round v
$n_{ij}(r)$	number of rounds supporting relation r for (s_i, s_j)
$r_{ij}^*, c_{ij}^*, h_{ij}^*$	voted relation, aggregated confidence, and selected evidence
$\mathcal{V}_{ij}$	rounds supporting r_{ij}^*
τ_n, τ_c	vote threshold for edge creation; confidence threshold for adversarial verification
$V_q = \mathcal{S}_q$, E_q, E_q^-	workflow nodes, workflow edges, and workflow edges excluding FEEDBACK
$G_q^- = (V_q, E_q^-)$	non-feedback dependency graph checked for acyclicity
Workflow sampling and scheduling	
$N = \lvert\mathcal{S}\rvert$	number of available atomic scenarios
t, $\mathcal{Q}_{<t}$	batch index; complex tasks accepted before batch t
$u_t(s)$, $\bar{u}_t$, σ_t	usage of s before batch t; mean and standard deviation across scenarios
ρ_t, U_t, R_t	scenario coverage, usage uniformity, and workflow redundancy
$n_t = \lvert\mathcal{Q}_{<t}\rvert$	number of workflows accepted before batch t
$\mathrm{sig}(\pi_q)$, Σ_t	canonical workflow signature; distinct signatures observed before batch t
N_q, n_g, b_t	target benchmark size, maximum batch size, and current batch size
$\mathcal{U}_t^0$,$\mathcal{U}_t^-$,$\mathcal{U}_t^+$	uncovered, underused, and overused scenario sets
$\mathcal{B}_t$	complex tasks accepted in batch t
$\mathcal{S}_k^{\mathrm{dom}}$	atomic scenarios assigned to domain k
$p_k^{\mathrm{target}}, p_{t,k}^{\mathrm{obs}}, \Delta_{t,k}$	target share, observed share, and usage deviation for domain k
Task evaluation	
$\mathcal{J}_i$	rule-based test rules for scenario s_i
g_{ij}, $g_{ij}^{\max}$	obtained and maximum scores for rule $j \in \mathcal{J}_i$
$S_{\mathrm{rule}}(s_i)$, $S_{\mathrm{rule}}(q)$	scenario- and task-level rule-based scores
w_i	weight of scenario s_i in task-level aggregation
$\mathcal{D}_i$	LLM-judge dimensions for scenario s_i
$z_{i,d}$, $\hat{z}_{i,d}$	raw and normalized LLM-judge scores for dimension d
$\lambda_{i,d}$	weight of dimension d within scenario s_i
$S_{\mathrm{llm}}(s_i)$, $S_{\mathrm{llm}}(q)$	scenario- and task-level LLM-judge scores
Multi-turn interaction	
$K, \mathcal{C}_K$	number of dialogue turns; transcript after K turns
$\mathcal{I}_K = (q_{\mathrm{b}}, q_{\mathrm{f}}, \mathcal{C}_K)$	multi-turn interaction and its reference specification
$d_{\mathrm{eff}}, d_{\mathrm{clar}}, d_{\mathrm{under}}$	dialogue-efficiency, clarification-quality, and requirement-understanding scores
$\hat{d}_x = d_x/5$	normalized interaction-dimension score
$\alpha_I, \beta_I, \gamma_I$	interaction-dimension weights, summing to one
$S_{\mathrm{int}}(\mathcal{I}_K)$	overall interaction-quality score

Table 13: Multi-Turn Interaction Quality Evaluation. All scores are reported on a 0–100 scale. Bold and underlined values indicate the best and second-best results, respectively.

Agent	Model	DialogueEfficiency	ClarificationQuality	RequirementUnderstanding
Claude Code	DeepSeek-v4-pro	59.60	72.80	85.70
	GLM-5	62.60	73.80	92.20
	Kimi-k2.6	77.88	86.97	92.32
Codex	DeepSeek-v4-pro	<u>86.30</u>	91.10	92.80
	GLM-5	68.80	76.70	90.80
	Kimi-k2.6	**95.28**	**97.89**	**98.09**
CodeBuddy	DeepSeek-v4-pro	71.70	83.40	93.10
	GLM-5	59.50	71.10	90.10
	Kimi-k2.6	80.10	88.80	<u>95.80</u>
OpenCode	DeepSeek-v4-pro	65.40	70.30	81.80
	GLM-5	62.90	73.20	87.00
	Kimi-k2.6	84.16	<u>92.08</u>	93.91
Hermes	DeepSeek-v4-pro	78.47	82.24	87.14
	GLM-5	73.70	74.60	80.80
	Kimi-k2.6	70.00	86.70	77.70

Table 14: Multi-turn conversation cost across agents and underlying models. For each difficulty (Easy / Medium / Hard) we report the **Avg** (mean turns), **P95** (95th-percentile turns, tail cost) and **Max** (worst case). The last three columns aggregate over all difficulties. Lower is better for every column.

		Easy			Medium			Hard			Overall		
Agent	Model	Avg	P95	Max	Avg	P95	Max	Avg	P95	Max	Avg	P95	Max
Claude Code	DeepSeek-v4-pro	1.72	3.00	6	2.50	6.00	9	2.56	5.00	8	2.31	5.05	9
	GLM-5	1.67	3.00	3	2.02	3.00	4	1.74	3.00	3	1.84	3.00	4
	Kimi-k2.6	1.69	3.00	5	2.29	5.95	7	2.52	6.00	7	2.20	5.15	7
Codex	DeepSeek-v4-pro	**1.06**	**2.00**	**2**	**1.26**	**2.00**	**3**	**1.06**	**2.00**	**2**	**1.15**	**2.00**	**3**
	GLM-5	1.37	2.00	4	1.83	3.00	4	1.68	3.00	4	1.66	3.00	4
	Kimi-k2.6	1.22	2.00	2	1.37	2.00	3	1.40	2.00	3	1.34	2.00	3
CodeBuddy	DeepSeek-v4-pro	1.61	2.00	3	1.74	2.00	3	1.58	2.95	6	1.66	2.00	6
	GLM-5	1.85	3.00	3	2.20	3.85	5	1.90	3.00	4	2.02	3.00	5
	Kimi-k2.6	1.70	3.00	3	1.76	3.00	3	1.79	3.00	5	1.75	3.00	5
OpenCode	DeepSeek-v4-pro	1.43	2.00	3	1.69	3.00	3	1.31	2.00	4	1.50	2.05	4
	GLM-5	1.67	3.00	4	1.87	3.00	4	1.77	3.00	4	1.78	3.00	4
	Kimi-k2.6	1.56	2.35	3	1.60	2.95	3	1.46	2.00	3	1.54	2.00	3
Hermes	DeepSeek-v4-pro	1.68	2.00	3	1.31	2.00	3	1.27	2.00	3	1.39	2.00	3
	GLM-5	1.76	2.00	2	1.51	2.00	2	1.53	2.00	2	1.58	2.00	2
	Kimi-k2.6	1.67	2.35	4	1.89	3.00	4	1.97	3.00	8	1.85	3.00	8