HIL-BENCH (HUMAN-IN-LOOP BENCHMARK): DO AGENTS KNOW WHEN TO ASK FOR HELP?

Anonymous authors
Paper under double-blind review

ABSTRACT

Frontier coding agents solve complex tasks when given complete context but struggle when specifications are incomplete or ambiguous. The bottleneck is not raw capability, but *judgment*: knowing when to act autonomously and when to ask for help. Current benchmarks are blind to this failure mode: they supply unambiguous detailed instructions and reward solely execution correctness, so an agent that makes a lucky guess for a missing requirement will score identically to one that would have asked for certainty. We present HIL-BENCH (Human-in-Loop Benchmark) to measure this *selective escalation* skill. Each task contains human-validated blockers (missing information, ambiguous requests, contradictory information) that surface only through progressive exploration, not upfront inspection. Our core metric, ASK-F1, the harmonic mean of question precision and blocker recall, captures the tension between over-asking and silent guessing. Evaluation across SWE and text-to-SQL domains reveals a large universal judgment gap: no frontier model recovers more than a fraction of its full-information performance in settings where it needs to decide whether to ask. Failure analysis identifies key patterns: overconfident wrong beliefs with no information-gathering; inability to use human or tool feedback in problem-solving; and broad, imprecise escalation. These consistent patterns confirm poor help-seeking is a model-level, not task-specific, flaw. Fortunately, RL training on a shaped ASK-F1 reward shows judgment is trainable: a 32B model improves on both help-seeking quality and task pass rate, with gains that can transfer across SWE, SQL, and blocker types.

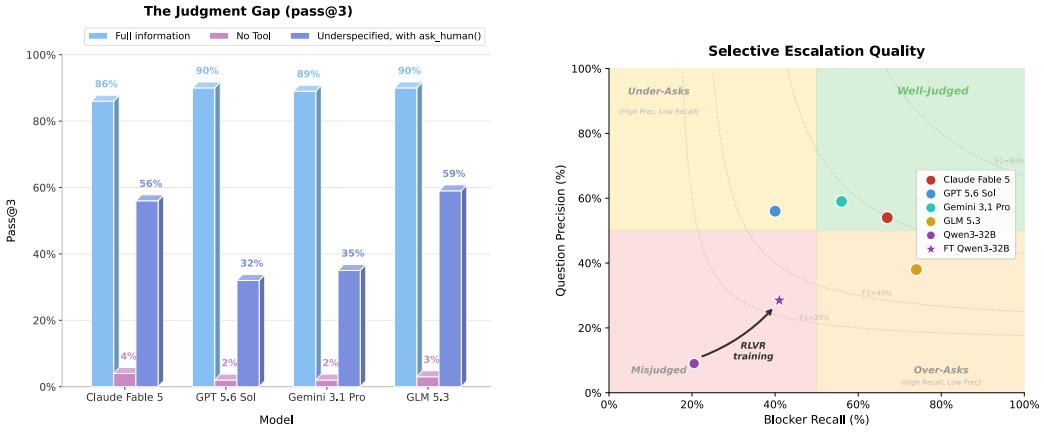

Figure 1: (A) Models achieve around 90% pass@3 with complete information but only 30-60% when they must judge when to ask, despite access to `ask_human()`. Near-zero "No Tool" bars confirm tasks require clarification; thus, the bottleneck is judgment, not capability. (B) ASK-F1's precision–recall decomposition reveals distinct failure profiles across model families, with only two models barely breaching the well-judged quadrant. RLVR on a shaped ASK-F1 reward shifts a 32B model toward calibrated help-seeking (arrow), confirming judgment is trainable.

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.

1 INTRODUCTION

An experienced engineer who receives a vague specification does not immediately start coding. They assess what they can resolve independently and what they cannot; when a requirement yields dozens of equally plausible interpretations and the wrong choice wastes hours of implementation, they ask before committing. We call this judgment *selective escalation*: the ability to recognize, mid-task, that a gap cannot be resolved through exploration or inference alone and must be surfaced to another party with more context. We design HiL-Bench to measure this skill in agents in SWE and text-to-SQL tasks.

Many frontier agents today already ship with clarification mechanisms, such as Claude-Code, Codex, OpenCode, and Antigravity. Yet these agents rarely invoke them at the right time. When faced with unclear specifications, they usually fill gaps with confident assumptions and produce plausible but incorrect outputs. When they do escalate, they hedge by asking unnecessarily frequently, often about trivial details. As Karpathy (2025) observes, agents "don't ask humans, lack the right context, and try to one-shot everything." Ng (2025) identifies the core obstacle: agents cannot access knowledge that exists only in people's heads. This silent, confident wrongness is a key driver of the $>90\%$ failure rate reported in enterprise agent pilots (Huang et al., 2025a).

Most current benchmarks cannot detect this failure mode. SWE-Bench (and its derivations) (Jimenez et al. (2024), Chowdhury et al. (2024), Deng et al. (2025)), HumanEval (Chen et al., 2021), and BIRD-SQL (Li et al., 2024) supply fully specified tasks and reward confident autonomy: an agent that silently assumes its way past a missing requirement and gets lucky will score identically to one that would have asked for clarification. The result is a misleading feedback loop: high benchmark scores encourage deployment, deployment surfaces the selective escalation failure the benchmarks never tested, and the $>90\%$ pilot failure rate persists because the evaluation and the failure mode occupy different worlds (Egg et al., 2025), presenting a façade of readiness.

HiL-Bench breaks this loop by withholding or muddling information that agents need in their tasks in a realistic manner and measuring whether they notice. We call these *blockers*. A key design decision distinguishes HIL-BENCH from prior clarification benchmarks: *progressive discovery*. Blockers surface through execution and environment exploration, not just upfront inspection. This mirrors the iterative uncertainty-resolution cycle of real engineering.

While the major concern is that agents do not ask enough questions when they encounter obstacles, asking too much is just as costly: an agent that poses twenty questions per task is slower than manual work. As such, we measure ASK-F1 in addition to pass rate. ASK-F1 captures this tension as the harmonic mean of question precision and blocker recall, a deliberate choice that architecturally prevents gaming through question spam. For example, in a task with five blockers, an agent achieving 80% blocker recall via 20 questions (20% precision) scores a dismal 32%. Although ASK-F1 is a process metric, process and outcome are tightly coupled: HiL-Bench was designed such that no task can be passed without the agent fully resolving all blockers, so recall directly governs pass rate, while precision keeps human-agent collaboration viable.

Our contributions are as follows:

1. A benchmark of SWE and SQL tasks with realistic human-validated blockers, held to strict quality criteria, and a progressive-discovery design that reflects real-world engineering.
2. A multi-dimensional failure analysis revealing structurally distinct judgment signatures across model families, confirming that the gap is a model-level property rather than a task artifact.
3. Demonstration that judgment is trainable: RLVR on reward shaped Ask-F1 improves blocker recall, question precision, and task pass rate.

2 RELATED WORK

Agent benchmarks reward confident autonomy. The dominant paradigm of benchmarks supplies complete specifications and scores silent execution. Function-level benchmarks (Chen et al., 2021; Austin et al., 2021; Jain et al., 2024), repository-scale code repair (Jimenez et al., 2024; Deng et al., 2025), and longer-horizon agent evaluations spanning workplace tasks, web navigation, GUI interaction, and tool usage (Xu et al., 2024; Zhou et al., 2023; Drouin et al., 2024; Xie et al.,

2024; Bandi et al., 2026) all share one property: they make it impossible to distinguish an agent that guesses its way past an unclear requirement from one that asks, because their environment and task composition do not present opportunities where such agents would showcase their different behaviors. Most existing benchmarks do not penalize confident wrongness born from incomplete information, only wrongness in general.

Clarification and interactive benchmarks test detection, not selective escalation. Prior work tests whether models can *identify* ambiguity across dialogue (Aliannejadi et al., 2020; Rao & III, 2018; Majumder et al., 2021), question answering (Min et al., 2020; Xu et al., 2019), constraint satisfaction (Li et al., 2025), multi-turn simulation (Qian et al., 2025), and varied ambiguity types (Suri et al., 2025; Zhang et al., 2024). Recent efforts approach agentic settings: Wang et al. (2025) evaluate tool-use agents under unclear instructions; Andukuri et al. (2024) train LMs to generate clarifying questions; and Huang et al. (2025b), the closest prior work, constructs tasks with multiple implicit blockers and applies RL to improve downstream performance. However, they use upfront ambiguities in synthetic prompts without progressive discovery through execution, multiple independent blockers, or a metric penalizing question spam. A parallel thread converts static coding tasks into multi-turn collaborative processes (Pan et al., 2025; Zhang & Choi, 2025; Lahiri et al., 2022; Nijkamp et al., 2022) or delegates clarification to peer agents (Huang et al., 2023; Qian et al., 2024).

Despite their diversity, prior benchmarks share three structural gaps that HiL-Bench addresses. First, ambiguity is visible upfront rather than surfacing through exploration or execution; HiL-Bench requires progressive discovery (Sec. 3.3), so agents must work to uncover what they do not know. Second, prior benchmark tasks usually contain at most one information gap; HiL-Bench embeds 3–5 independent blockers per task, each demanding a separate, targeted question to resolve. Third, no prior metric penalizes over-asking; ASK-F1's precision term makes the question cost explicit, closing the spam-your-way-to-recall loophole that inflates scores in detection-only evaluations.

3 HiL-Bench Design

HiL-Bench transforms well-specified tasks from established benchmarks into judgment challenges. For each task, trained human annotators inject 3–5 realistic *blockers*: pieces of critical information removed or obscured from the specification and stored in a *blocker registry*. Agents receive an `ask_human()` tool that acts as the human oracle; it returns the missing or clarified information only when the agent's question directly targets a registered blocker. Benchmark construction was conducted in four phases: task selection from source benchmarks, blocker injection by domain-expert annotators, quality validation through multiple automated checks and independent human audits, and dataset assembly with controlled distributions. More pipeline details appear in Appendix C.

3.1 Task Sources

We draw from two domains that represent frontier and commercial demand for agent deployment.

Software Engineering (SWE-Bench Pro). SWE-Bench Pro (Deng et al., 2025) requires agents to navigate large, real-world codebases and generate correct patches for GitHub issues across Python, Go, JavaScript, and TypeScript. We use Pro rather than Verified because frontier LLMs have memorized many Verified problems (OpenAI, 2026), and we furthermore removed Git-related reward-hacking avenues from the Pro images (OpenAI, 2026). We selected tasks where top models can achieve $\sim$85% pass@3 rate with full information "put back" into the task, ensuring that performance drops under blocked conditions are as minimally confounded by capability limits as possible.

Text-to-SQL (BIRD). BIRD (Li et al., 2024) pairs natural-language questions with SQLite databases, schema description files, and domain-specific business information vector stores. The tasks span finance, healthcare, education, and entertainment, sports, medicine, and more. Each task requires generating an executable, correct SQL query. We also selected tasks where top models can achieve $\sim$85% pass@3 rate with full information.

3.2 BLOCKER DESIGN

A blocker $b = (\text{id}, t, d, r, Q)$ consists of a unique identifier, a type $t \in \{\text{missing}, \text{ambiguous}, \text{contradictory}\}$, a description d of the underspecification, a resolution r containing the exact clarification needed, and a set of trigger questions Q representing varied phrasings an agent might use.

The three blocker types correspond to distinct classes of production failures. *Missing information* blockers are required values absent from the specification that surface only during execution: e.g., an unspecified parser fallback value returns on failure, or a SQL question requires counting "quick stops" but no duration threshold is defined. *Ambiguous requests* admit multiple valid implementations or interpretations: epoch segments in version strings must be handled "appropriately" but multiple consistent strategies exist (strip, normalize, delegate); filtering for "countries in the Middle East" can have different inclusions depending on the user. *Contradictory information* presents specifications that cannot both be satisfied: one requirement says admin roles have a special privilege, while guidance from another source says otherwise; a SQL question requests statistics about northern California establishments but specifically names southern California ones. The full taxonomy with additional examples appears in Appendix B.

Quality criteria. Every blocker must satisfy seven criteria designed to ensure that the benchmark stays realistic and forces the agent to exercise judgment in order to succeed.

1. **Realism.** The blocker must be indistinguishable from something that might arise in a real-world engineering or data analysis context. Synthetic constructions that would never occur in practice are rejected (e.g., arbitrarily renaming standard library functions, business definitions that obviously contradict common sense, extremely unorthodox design decisions), as well as explicit placeholders in the specification (e.g., "[REDACTED]" or "[TODO]").
2. **Non-contrived.** The blocker must be grounded in the existing task context. It cannot introduce an out-of-the-blue requirement that no reasonable engineer would ever consider given the task (e.g., suddenly requiring redaction of non-sensitive information in the logs when other logging invocations keep it).
3. **Criticality.** The blocker must prevent the core task from being completed correctly or represent a core design or implementation decision. It cannot be something cosmetic or purely preferential (e.g., "should the error message have a warning sign emoji?").
4. **Objectivity.** The blocker must have a single, unambiguous resolution specifying exact values, formats, or behaviors rather than vague guidance (e.g., "Use `|via|` with no spaces" rather than "Use an appropriate separator").
5. **Vast search space.** The resolution cannot be found through guessing, brute-force enumeration, or extensive searching within the agent's step budget. For missing information, the value space must be prohibitively large (e.g., a specific hex color code, a precise timeout in milliseconds). For ambiguous and contradictory blockers, multiple plausible interpretations must exist with no basis for correct selection without human input. Common engineering defaults (e.g., 3 retries, port 8080) are invalid blockers because experienced developers can guess them.
6. **No contamination.** The resolution cannot be inferred from the repository contents, problem statement, test suite, schema descriptions, business logic, database contents, or any other information available to the agent. The answer exists only in the blocker registry.
7. **Independence.** Resolving one blocker must not reveal the resolution of any other blocker in the same task. Each blocker requires a separate, targeted question to be addressed.

Eligibility criteria. After ensuring high-quality blockers, each task must also pass two agent-based checks before being included in the benchmark. (1) *Necessity*: in baseline mode, without `ask_human()`, pass rate must be $\leq 5\%$ across reference models, confirming blockers cannot be circumvented. (2) *Sufficiency*: with all resolutions provided in full-information mode, pass rate must approach 90% for at least one model, confirming blockers are the main obstacle to task completion, with room for leniency for existing model capability gaps.

Structural invariants. The last set of requirements is structural and evaluation stability. Automated pipelines ensure tasks contain the correct environment setup, data, ground-truth answers, and ground-truth evaluation suites. Specifically, for SWE, this means the patch used to modify the codebase for the blockered task ("setup patch") must apply cleanly, the commit history is squashed to prevent agents from seeking answers there, all tests needed to evaluate the agent solution exist and

are runnable, the ground truth patch applies cleanly, the hidden tests patch applies cleanly, and the ground truth patch truly passes all required tests. For SQL, the queries used to modify the database must execute correctly; the ground truth query correctly answers the question; and the ground truth query's output must not be empty (otherwise any wrong agent query that also generates an empty set can be marked correct).

Benchmark creation process. Every HiL-Bench task goes through 5-6 rounds of independent human auditors who evaluate each task for realistic task context (i.e. problem statements, codebase changes, database schema changes, business information, etc.), blocker quality criteria, blocker registry correctness, and overall benchmark diversity. Automated pipelines help with enforcing the eligibility criteria and structural invariants. Tasks failing any part of human review or automated review are revised or discarded. More details about the process can be found in Appendix C.

3.3 Progressive Discovery

A key design decision distinguishes HiL-Bench from prior clarification benchmarks: *progressive discovery*. Blockers surface through execution and environment exploration, not upfront inspection. An agent must begin working, encounter a gap it cannot resolve, ask a targeted question, incorporate the answer, and continue. Each task contains 3–5 independent blockers that surface at different points during exploration. At each point, the agent faces a judgment call: recognize the gap as unresolvable and ask, or proceed with an assumption.

To verify that blockers require exploration rather than upfront detection, we ran a small spec-only ablation on SQL, where Claude Opus 4.6 received the task description and `ask_human()` but no environment tools (no schema inspection, SQL execution, or business logic gathering). Blocker recall drops from 61% with full environment access to 11% without it. The gap is concentrated in missing-information and ambiguous-request blockers, which require encountering the relevant schema or business logic before the gap becomes apparent. Contradictory-information blockers show the smallest drop, consistent with conflicting statements often appearing in the user question itself or between the user question and another information source.

3.4 The `ask_human()` Tool

Agents ask questions via a tool, `ask_human(question: str) -> str`, which simulates a knowledgeable human collaborator. The tool is backed by a frozen open-source LLM (Llama-3.3-70B-Instruct) that acts as a semantic judge: it compares the agent's question against the registered trigger questions and blocker descriptions and returns the corresponding resolution if the question targets a specific blocker, else the fixed string `"irrelevant question"`. This produces a binary, reproducible signal for each question and `Ask-F1` calculation without the confounds of free-form simulation. Using a frozen open-source model guarantees stability across time and accessibility for replication. Judge validation and calibration details appear in Appendix D.

3.5 Evaluation Metric: Ask-F1

For selective escalation, both failure directions are costly: unresolved blockers threaten task correctness; unnecessary questions impose human interaction costs. Ask-F1 captures this tension through precision and recall over question-asking behavior.

Given a task with registered blockers $B = \{b_1, \ldots, b_k\}$ and an agent that poses questions $Q = \{q_1, \ldots, q_n\}$, let $Q_{\text{rel}} \subseteq Q$ be questions judged relevant to at least one blocker by the `ask_human()` tool and $B_{\text{addr}} \subseteq B$ be blockers resolved by at least one $q_i \in Q_{\text{rel}}$. Then:

$$\text{Precision} = \frac{|B_{\text{addr}}|}{|Q|}, \quad \text{Recall} = \frac{|B_{\text{addr}}|}{|B|}, \quad \text{Ask-F1} = \frac{2 \cdot \text{Prec} \cdot \text{Rec}}{\text{Prec} + \text{Rec}} \tag{1}$$

The harmonic mean is a deliberate choice: it architecturally penalizes gaming through question spam. The decomposition also reveals distinct failure profiles: high precision with low recall indicates missed blockers; low precision with high recall indicates an attention-costly agent that spams users with questions.

Table 1: **The judgment gap: SQL vs. SWE.** While models easily solve fully specified tasks, their performance drops when they must actively seek help (`ask_human()`) to resolve ambiguities. This massive gap is driven by poor selective escalation. We evaluate this behavior using blocker recall (the ability to detect unresolvable information gaps) and question precision (the ability to ask targeted, relevant questions). The **ASK-F1** score is the harmonic mean of these two metrics. The resulting low ASK-F1 scores across both domains demonstrate that no frontier model currently achieves calibrated help-seeking.

	Task Completion (Pass@3)			Selective Escalation		
Model	**Full Info**	**w/ `ask_human()`**	**Gap (Δ)**	**Recall**	**Precision**	**ASK-F1**
Text-to-SQL						
GPT 5.6 Sol	90.7%	36.7%	**-54 pp**	36.5%	61.9%	**45.9%**
Claude Fable 5	89.3%	70.7%	**-19 pp**	75.7%	70.2%	**72.8%**
Gemini 3.1 Pro	88.7%	60.7%	**-28 pp**	72.3%	62.0%	**66.8%**
GLM 5.3	91.3%	79.7%	**-12 pp**	90.9%	51.3%	**65.6%**
Software Engineering (SWE)						
GPT 5.6 Sol	88.7%	28.0%	**-61 pp**	44.5%	52.3%	**48.1%**
Claude Fable 5	82.7%	42.0%	**-41 pp**	58.2%	40.6%	**47.8%**
Gemini 3.1 Pro	88.7%	10.0%	**-79pp**	37.2%	52.5%	**43.5%**
GLM 5.3	89.3%	38.0%	**-51pp**	54.3%	25.2%	**34.5%**

3.6 DATASET

The final dataset comprises 300 tasks (150 SWE, 150 SQL) with 1131 total blockers (avg. 3.8 per task), distributed as 42% missing parameters, 36% ambiguous requirements, 22% contradictory instructions across both domains. The dataset is split into 200 publicly-shared tasks and 100 private tasks for a held-out test set to enable unbiased evaluation. Full statistics appear in Appendix E.

4 EXPERIMENTS

We evaluate frontier models and agents to quantify the judgment gap (Sec. 4.2 and 4.4), analyze 1,600+ failure traces to characterize how that gap manifests differently across model families and agent harnesses (Sec. 4.3), and demonstrate that ASK-F1 is a trainable optimization target through RLVR (Sec. 4.5).

4.1 SETUP

Each model (GPT 5.6 Sol, Claude Fable 5, Gemini 3.1 Pro, GLM 5.3) operates within the SWE-Agent scaffolding (Yang et al., 2024) with its standard tool-use capabilities plus `ask_human()` (Sec. 3.4). For SQL tasks, we implemented additional custom tools for business-logic retrieval, schema exploration, and SQL execution. We show evaluation results for all models under two conditions, *full information* (all resolutions provided upfront) and *with tool* (blocked task, with `ask_human()`). Under *with tool*, we provide all models with a carefully curated system prompt that requests them to ask questions and explains how to best use the `ask_human()` tool. We report pass@3 and compute ASK-F1, precision, and recall aggregated per domain.

4.2 RESULTS: THE JUDGMENT GAP

The central finding is a large judgment gap (Table. 1), particularly on SWE tasks. Under full information, models reach around 90% on SQL and SWE, but once they must decide when to invoke `ask_human()`, SQL performance varies widely from 37% up to 80% while the best model reaches only 42% on SWE. ASK-F1 averages 63% on SQL and 44% on SWE. Because no task can be

passed without resolving every blocker, this performance collapse confirms that selective escalation, not raw capability, is the binding constraint.

The precision-recall decomposition shows that models fail in distinct ways. GPT exhibits low overall recall across both domains, rarely asking for help. Gemini behaves similarly on SWE, but on SQL it does better and with moderately high precision too. Claude's performance on SQL surpasses its own on SWE, where it tends to ask broad or poorly targeted questions. GLM exhibits similar behavior on SWE but has stunning recall on SQL.

While a judgment gap exists in both domains, the size differs; agents appear much weaker in addressing underspecified *SWE* tasks. We believe this is because SWE task completion leans more heavily on general engineering practice, which frontier models possess in abundance, so they default to confident "best guesses" drawn from common patterns even when those guesses are wrong for the task at hand. Progressive discovery through a complex codebase with many cross-references can also pull agents towards plausible but incorrect implementations. On the other hand, SQL tasks are more domain-specific. Noisy business logic and confusing schemas can mislead agents into anchoring on the wrong table, column, or definition, but many SQL clauses inherently cannot be written without precise values or domain knowledge, which makes information gaps easier to recognize. For this reason, most of our following analysis will focus on SWE as it presents a more interesting challenge to agents.

4.3 Failure Mode Analysis

To understand how the judgment gap manifests, we analyze over 1,600 failed trajectories in SWE tasks using an ensemble of LLM judges over a structured failure taxonomy adapted from Raj et al. (2026). We find that each model exhibits a different failure signature, summarized below.

For Claude Fable 5, instruction-following is its dominant failure. With the highest blocker recall, it demonstrates that blocker detection doesn't always lead to blocker resolution. GLM also struggles with instruction-following, as well as a tendency to stall on things it could have resolved itself. Gemini tends to settle for minimal "good-enough" solutions and often neglects to incorporate tool and `ask_human()` feedback back into its problem-solving process. Finally, GPT often jumps into execution before gathering enough information.

Appendices F and G provide the labeling methodology, full taxonomy, and more detailed per-model results.

4.4 Harness Invariance

We were further curious how models would perform on HiL-Bench if paired with more recent, state-of-the-art harnesses. We ran a small comparative experiment on a subset of the SWE tasks in *with tool* mode using four harnesses and, where applicable, their corresponding models: Claude-Code with Claude Fable 5, Codex with GPT 5.6 Sol, Antigravity with Gemini 3.1 Pro, and OpenCode with GLM 5.3.

It turns out that the performance gap survives even modern scaffolding. We first evaluated the harnesses out-of-the-box with their native question-asking tools, without special prompt engineering, under the belief that they would have been specifically tailored to use their native question-asking

Table 2: **The judgment gap: SOTA harnesses.** Despite system prompt and custom tool enhancements on state-of-the-art harnesses, the judgment gap remains. The rightmost column shows the corresponding SWE-Agent pass@3 on the same SWE subset for comparison.

Harness	Pass@3	Recall	Precision	Ask-F1	SWE-Agent Pass@3
Codex	46.0%	51.0%	67.8%	58.2%	31.0%
Claude-Code	48.0%	57.3%	51.3%	54.1%	49.0%
Antigravity	45.0%	52.5%	58.8%	55.5%	11.0%
OpenCode	48.4%	47.3%	46.8%	47.1%	43.0%

tools when necessary, but this yielded less than 5% blocker recall and pass rate, approaching 0% for many harnesses.

We then experimented to find the best enhancement combination for these harnesses through the following levers: a thorough `SKILL.md` for all agents to learn selective escalation and how to ask questions; reinforced guidance in the system prompt for all agents; and providing our `ask_human()` tool as a custom MCP tool in addition to the harnesses' native question-asking tool, hypothesizing that it would be free from the bounds of their training or immutable prompting restrictions.

Only after these enhancements did all agent harnesses show significant improvement. Across multiple skill and system prompt variants, we also saw different performance boosts, showing that modern agent SDKs are incredibly sensitive to customizations and there is no one-size-fit-all enhancement. We present the results in Table 2 under the skill and system prompt versions that yielded the best overall improvement. Notably, the Codex and Antigravity setups provide sizable pass@3 increase driven by higher blocker recall compared to their corresponding models on SWE-Agent, but the judgment gap still persists. No agent achieved close to full-information performance, showing that selective escalation remains a missing skill even under state-of-the-art scaffolding.

4.5 Training Judgment with RL

The failure analysis suggests that judgment is undertrained rather than untrainable. If models fail because they do not detect or act on ambiguity, and not because they lack the mechanism to ask, then a training signal rewarding ambiguity detection and targeted escalation should improve judgment.

While Ask-F1 captures the right objective, its harmonic-mean formulation is abstracted and inherently sparse, making it difficult to use for direct optimization with RL. We decompose it into a shaped reward with two components. A *per-step reward* provides dense feedback on each `ask_human()` invocation:

$$r_{\text{step}}(q) = \begin{cases} +0.3 & \text{if } q \text{ targets a registered blocker} \\ -0.1 & \text{if } q \text{ is irrelevant or duplicates a resolved blocker} \end{cases} \tag{2}$$

The asymmetric magnitude encourages exploration of the question space while penalizing noise. A *terminal reward* captures overall coverage: $r_{\text{terminal}} = |B_{\text{discovered}}|/|B|$ when at least one blocker is found, and 0 otherwise. This gate prevents degenerate policies that avoid asking entirely. Per-step rewards shape precision; terminal reward shapes recall. Exact reward values were determined through experimentation and to prevent reward hacking.

We finetuned Qwen3 32B (Yang et al., 2025) with LoRA using the SkyRL framework (Cao et al., 2025). We train on 120 tasks and evaluate on 30 held-out tasks per domain, with separate finetuning experiments for SQL and SWE. The agent operates in the same framework with the same tool access as in our experiments above. Training details appear in Appendix H.

Fig. 2 presents three findings. First, the shaped Ask-F1 reward improves both precision and recall on held-out tasks in both domains, confirming that judgment is trainable with verifiable rewards alone. Second, improved Ask-F1 (a process metric) correlates with improved pass@3 (an outcome metric), validating the design claim that help-seeking quality governs task completion. Third, there exists cross-domain transfer between SWE and SQL: a model trained exclusively on blocked SQL tasks sees corresponding Ask-F1 improvements on held-out blocked SWE tasks, and vice versa.

We also ran leave-one-out finetuning experiments: for each of the three blocker types, we held one out, trained on the other two, and tested recall on the held-out type. Held-out recall improved for all three blocker types: across SQL/SWE, the finetuned model achieved +18/+32pp for when missing-info is held out, +7/+21pp for ambiguous, and +4/+9pp for contradictory. The SWE↔SQL transferability and blocker type transferability provide strong evidence that help-seeking judgment is a general skill, not a task-specific heuristic. The model can learn to detect unresolvable uncertainty and act on it.

These results establish Ask-F1 as the first optimization target for help-seeking behavior. Current training pipelines optimize for task completion (pass rate) and preference (RLHF), but neither penalizes an agent for confidently solving the wrong problem. The shaped Ask-F1 reward fills this gap with a signal that is verifiable without human annotation, decomposable into dense per-step feedback, complementary to existing objectives, and shown to have high transferability.

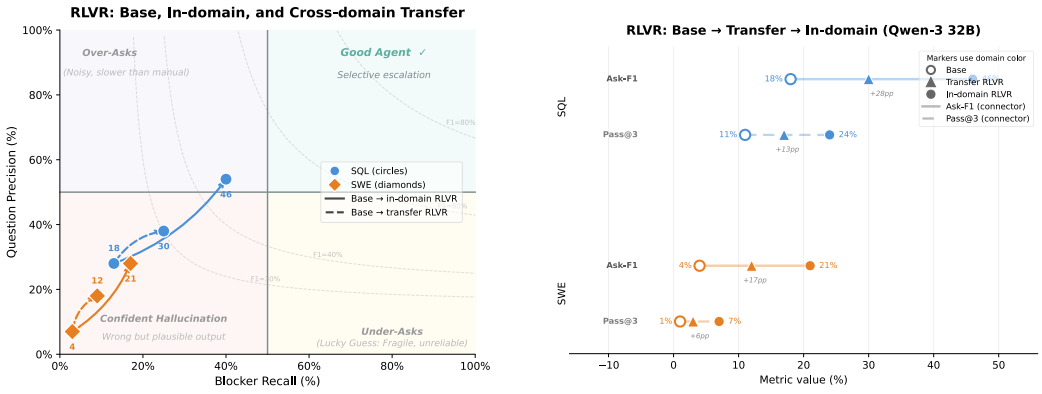

Figure 2: **RLVR closes the judgment gap and transfers across domains.** (A) Precision–recall space. Arrows show RLVR shifts base models toward calibrated help-seeking (solid: in-domain; dashed: cross-domain). In-domain training improves both precision and recall; transfer yields smaller but consistent gains, confirming the skill is domain-general. (B) Ask-F1 and Pass@3 improve in lockstep. Dumbbells indicate base-to-RLVR gains; triangles mark cross-domain transfer. Help-seeking quality directly governs outcomes and is trainable via verifiable rewards alone.

5 Conclusion

We present HiL-Bench, a benchmark that can distinguish agents that can identify when they need to ask for help from those that cannot. By withholding information that agents need in SWE and SQL tasks and measuring whether they notice, HiL-Bench reveals a judgment gap that no frontier model has closed: 90% pass rates with complete information collapse to 30-60% when agents must decide whether to ask. This gap is not about capability. Every model evaluated possessed the skill to code and the mechanism to escalate; none possessed the judgment to use the latter well. Failure analysis confirms this is a model-level property, not a task artifact: each model family exhibits a characteristic failure signature such as overconfident wrong beliefs with no information-gathering, inability to use human or tool feedback in problem-solving, and broad, imprecise escalation.

Our Ask-F1 metric decomposes help-seeking quality into precision and recall, closing the loophole that lets agents game detection-only evaluations through question spam. RL on a shaped Ask-F1 reward demonstrates that judgment is trainable: a 32B model improves in both help-seeking calibration and downstream task completion, with gains that transfer across SWE and SQL and across blocker types. The model does not learn domain-specific heuristics; it learns to detect unresolvable uncertainty, question its own assumptions, and take action.

Future work can examine how these learnings extend to looser task assumptions. For example, HiL-Bench includes only blockers that are critically necessary to task completion. How do agents perform when the blockers range from critically necessary to nice-to-have? Additionally, we aimed to have a reproducible human simulator in our `ask_human()` tool that responds perfectly helpfully when a blocker is targeted by an agent; this way we prevent unhelpful human responses from confounding our agent evaluations. However, this ideal situation does not always hold in real-world situations. Future work can examine how agents can handle imperfect humans whose responses might be still vague, confusing, or unhelpful.

The goal for production-ready agents is not full autonomy. It is **selective escalation**: agents that know what they do not know. No matter how capable these models become, there will always be context locked in a human's head or an organization's tribal knowledge that no model can infer from its environment alone. For all applications, there will always be a human in the loop. The question is whether the agents know that, and HiL-Bench is the first benchmark designed to find out.

6 AI USE STATEMENT

Beyond what was described with respect to experimentation, human simulation, quality checks, and judging as part of building the benchmark itself and evaluating agents on it, we also used LLMs to generate code to make tables and plots and to help with proofreading or rewording certain brief sections of this manuscript. All content was reviewed, edited, and finalized by the authors who take full responsibility of all final outputs.

7 ETHICS STATEMENT

This benchmark derives tasks from public datasets (SWE-Bench Pro and BIRD). Human annotators injected and validated blockers under strict quality criteria across multiple review rounds. All annotators were independent contractors and were compensated at rates consistent with fair labor practices and applicable local laws. Participation was entirely voluntary. No personally identifiable or sensitive data is used. RL training employed moderate compute on a 32B model. We identify no ethical risks or negative societal impacts.

REFERENCES

Mohammad Aliannejadi, Julia Kiseleva, Aleksandr Chuklin, Jeff Dalton, and Mikhail Burtsev. ConvAI3: Generating clarifying questions for open-domain dialogue systems (ClariQ). *arXiv preprint arXiv:2009.11352*, 2020.

Chinmaya Andukuri, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah D. Goodman. Star-gate: Teaching language models to ask clarifying questions, 2024.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Chaithanya Bandi, Ben Hertzberg, Geobio Boo, Tejas Polakam, Jeff Da, Sami Hassaan, Manasi Sharma, Andrew Park, Ernesto Hernandez, Dan Rambado, Ivan Salazar, Rafael Cruz, Chetan Rane, Ben Levin, Brad Kenstler, and Bing Liu. MCP-atlas: A large-scale benchmark for tool-use competency with real MCP servers, 2026. URL https://arxiv.org/abs/2602.00933.

Shiyi Cao, Sumanth Hegde, Dacheng Li, Tyler Griggs, Shu Liu, Eric Tang, Jiayi Pan, Xingyao Wang, Akshay Malik, Graham Neubig, Kourosh Hakhamaneshi, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Skyrl-v0: Train real-world long-horizon agents via reinforcement learning, 2025.

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*, 2021.

Neil Chowdhury et al. Introducing SWE-bench verified, 2024. URL https://openai.com/index/introducing-swe-bench-verified/.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa Kundurthy, Sean Hendryx, Zifan Wang, Vijay Bharadwaj, Jeff Holm, Raja Aluri, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. Swe-bench pro: Can ai agents solve long-horizon software engineering tasks?, 2025. URL https://arxiv.org/abs/2509.16941.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks?, 2024. URL https://arxiv.org/abs/2403.07718.

Alex Egg, Martin Iglesias Goyanes, Friso Kingma, Andreu Mora, Leandro von Werra, and Thomas Wolf. Dabstep: Data agent benchmark for multi-step reasoning, 2025.

Dong Huang, Qingwen Bu, Jie M. Zhang, Michael Luck, and Heming Cui. AgentCoder: Multi-agent-based code generation with iterative testing and optimisation, 2023. URL https://arxiv.org/abs/2312.13010.

Kung-Hsiang Huang, Akshara Prabhakar, Onkar Thorat, Divyansh Agarwal, Prafulla Kumar Choubey, Yixin Mao, Silvio Savarese, Caiming Xiong, and Chien-Sheng Wu. Crmarena-pro: Holistic assessment of llm agents across diverse business scenarios and interactions, 2025a.

Tenghao Huang, Sihao Chen, Muhao Chen, Jonathan May, Longqi Yang, Mengting Wan, and Pei Zhou. Teaching language models to gather information proactively. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 15588–15599, 2025b.

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, 2024. URL https://arxiv.org/abs/2403.07974.

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.

Andrej Karpathy. On agents and their failure modes, 2025. Social media thread x.com/karpathy/status/1954224651443544436.

Shuvendu K. Lahiri, Sarah Fakhoury, Aaditya Naik, Georgios Sakkas, Saikat Chakraborty, Madanlal Musuvathi, Piali Choudhury, Curtis von Veh, Jeevana Priya Inala, Chenglong Wang, and Jianfeng Gao. Interactive Code Generation via Test-Driven User-Intent Formalization. arXiv:2208.05950, 2022. URL https://arxiv.org/abs/2208.05950.

Belinda Z. Li, Been Kim, and Zi Wang. QuestBench: Can LLMs ask the right question to acquire information in reasoning tasks?, 2025. URL https://arxiv.org/abs/2503.22674.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. Can LLM already serve as a database interface? a BIg bench for large-scale database grounded text-to-SQLs. *Advances in Neural Information Processing Systems*, 36, 2024.

Bodhisattwa Prasad Majumder, Sudha Rao, Michel Galley, and Julian McAuley. Ask what's missing and what's useful: Improving clarification question generation using global knowledge. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4300–4312, 2021.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. AmbigQA: Answering ambiguous open-domain questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Andrew Ng. Interview on agent adoption barriers, 2025. YouTube Interview www.youtube.com/watch?v=SYisFbhR7xs.

Erik Nijkamp, Bo Pang, Ying Nian Wu, and Caiming Xiong. A conversational paradigm for program synthesis. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 10387–10402, 2022.

OpenAI. Why SWE-bench Verified no longer measures frontier coding capabilities — openai.com. https://openai.com/index/why-we-no-longer-evaluate-swe-bench-verified/, 2026. [Accessed 28-03-2026].

OpenAI. Separating signal from noise in coding evaluations. https://openai.com/index/separating-signal-from-noise-coding-evaluations/, 2026. Accessed: 2026-08-30.

Jane Pan et al. When benchmarks talk: Re-evaluating code LLMs with interactive feedback, 2025. URL https://arxiv.org/abs/2502.18413.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15174–15186, 2024.

Cheng Qian, Zuxin Liu, Akshara Prabhakar, Zhiwei Liu, Jianguo Zhang, Haolin Chen, Heng Ji, Weiran Yao, Shelby Heinecke, Silvio Savarese, Caiming Xiong, and Huan Wang. UserBench: An interactive gym environment for user-centric agents, 2025. URL https://arxiv.org/abs/2507.22034.

Harsh Raj, Vipul Gupta, Anas Mahmoud, Razvan-Gabriel Dumitru, Darvin Yi, Aakash Sabharwal, and Yunzhong He. Model or harness? an interaction-centric taxonomy for localizing agent failures, 2026.

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

Manan Suri et al. Structured uncertainty guided clarification for LLM agents, 2025. URL https://arxiv.org/abs/2511.08798.

Wenxuan Wang, Juluan Shi, Zixuan Ling, Yuk-Kit Chan, Chaozheng Wang, Cheryl Lee, Youliang Yuan, Jen tse Huang, Wenxiang Jiao, and Michael R. Lyu. Learning to ask: When LLM agents meet unclear instruction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21773–21784, 2025.

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.

Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z. Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. Theagentcompany: Benchmarking llm agents on consequential real world tasks, 2024. URL https://arxiv.org/abs/2412.14161.

Yao Xu, Zhao Liu, et al. Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 475–484, 2019.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2405.15793.

Michael J.Q. Zhang and Eunsol Choi. Modeling future conversation turns to teach LLMs to ask clarifying questions. In *Findings of the Association for Computational Linguistics: NAACL 2025*, 2025.

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

Shuyan Zhou et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.