FROM CONTEXT TO SKILLS: CAN LANGUAGE MODELS LEARN FROM CONTEXT SKILLFULLY?

Anonymous authors
Paper under double-blind review

ABSTRACT

Many real-world tasks require language models (LMs) to reason over complex contexts that exceed their parametric knowledge. This calls for *context learning*, where LMs directly learn relevant knowledge from the given context. An intuitive solution is inference-time skill augmentation: extracting the corresponding rules and procedures from *context* into explicit, natural-language *skills*. However, constructing such skills for context learning scenarios faces two fundamental challenges: the prohibitive cost of manual skill annotation for long, technically dense contexts, and the lack of external feedback for automated skill construction. In this paper, we propose ***Ctx2Skill***, a self-evolving framework that autonomously discovers, refines, and selects context-specific skills without human supervision or external feedback. At its core, a multi-agent self-play loop has a *Challenger* that generates probing tasks and rubrics, a *Reasoner* that attempts to solve them guided by an evolving skill set, and a neutral *Judge* that provides binary feedback. Crucially, both the *Challenger* and the *Reasoner* evolve through accumulated natural-language skills: dedicated *Proposer* and *Generator* agents analyze failure cases and synthesize them into targeted skill updates for both sides, enabling automated skill discovery and refinement. To prevent adversarial collapse caused by increasingly extreme task generation and over-specialized skill accumulation, we further introduce a *Cross-time Replay* mechanism that identifies the skill set achieving the best balance across representative cases for the *Reasoner* side, ensuring robust and generalizable skill evolution. The resulting skills can be plugged into any language model to obtain better context learning capability. Evaluated on four context learning tasks from CL-bench, *Ctx2Skill* consistently improves solving rates across backbone models, e.g., lifting GPT-4.1 from 11.1% to 16.5%, GPT-5.1 from 21.1% to 25.8%.

1 INTRODUCTION

Current language models (LMs) have achieved impressive performance on problems whose relevant knowledge was present during large-scale pre-training [5, 27, 39], such as competition-level mathematical problems [14] and competitive programming challenges [17]. However, many real-world tasks require LMs to learn from complex contexts and leverage new knowledge rather than parametric knowledge to reason and solve them effectively [26, 36, 47]. Previous works refer to this capability as ***context learning*** [12]. Effective context learning enables models to reason beyond their pre-trained knowledge and solve complex, domain-specific tasks by learning directly from rich contextual information, much as humans do. For example, it allows LMs to rapidly make use of previously unseen product documentation to generate step-by-step operational procedures or troubleshoot issues.

Despite the central role of context learning in solving real-world tasks, it has been largely overlooked in current research [16]. A key obstacle is the diversity of real-world contexts: contextual knowledge can be conveyed through books, experimental data, and search results, among others, making context learning fundamentally difficult. An intuitive paradigm is inference-time skill augmentation [3], i.e., extracting rules and procedures from ***context*** into natural-language ***skills*** that encode reusable procedural knowledge for LMs. This approach has proven effective in many agent tasks [42, 49, 37], e.g., coding agents [24]. However, this paradigm faces two fundamental challenges in context learning scenarios. Specifically, ***(1) Prohibitive Cost for Manual Skill Annotation***: Existing agent skill libraries are predominantly constructed through human annotation [18, 20, 41]. However, this paradigm breaks down in context learning scenarios, where contexts are long, technically dense,

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.

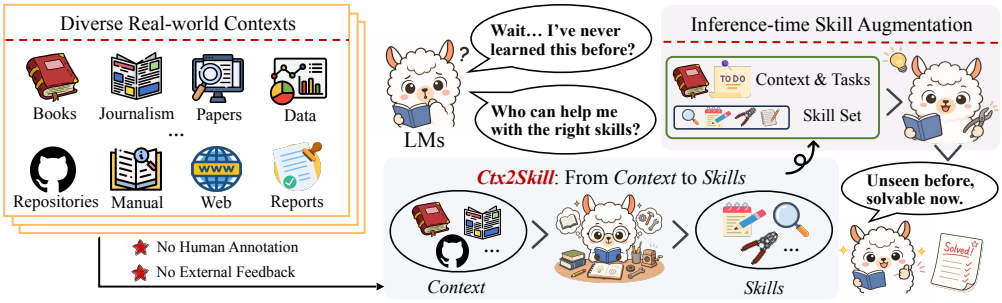

Figure 1: **The illustration of *Ctx2Skill*.** It's designed to extract rules and procedures from *context* into natural-language *skills* without human annotation and external feedback.

and domain-specific. Curating high-quality skills requires annotators to fully internalize complex, multi-section documents, a process that is both cognitively demanding and economically infeasible, rendering human-curated skill construction impractical at scale. ***(2) Lack of External Feedback for Automated Skill Construction***: Unlike verifiable tasks such as coding or mathematical reasoning [7, 31], where skill quality can be assessed via execution feedback or ground-truth comparison, context learning tasks lack automatic feedback to evaluate whether extracted skills faithfully and completely capture context-specific knowledge. Given only the context, there is no external feedback signal to tell whether a generated skill is useful or whether critical knowledge has been omitted. This feedback-free nature renders automated skill construction pipelines inapplicable [1, 44, 46]. *Together, these challenges highlight the need for an automated framework that discovers skills from the context without relying on either human annotation or external feedback signals.*

In this paper, we propose ***Ctx2Skill*** shown in Figure 1, which is designed to autonomously discover, refine, and select skills directly from complex contexts via skill-optimized self-play, requiring neither human annotation nor external feedback. At the core of *Ctx2Skill* is a multi-agent self-play loop [10, 22, 48, 21] comprising two competing but co-evolving roles: a *Challenger* agent, which generates a batch of tasks and associated rubrics [15] based on context and its skill set, aiming to probe deep contextual understanding, and a *Reasoner* agent, which reads the context and attempts to solve these tasks guided by its current skill set. The core idea of *Ctx2Skill* is that the two agents co-evolve by iteratively updating their respective skill sets rather than model parameters. Specifically, a neutral *Judge* agent evaluates the *Reasoner*'s responses against each rubric, producing feedback on whether it passes. Crucially, both the *Challenger* and *Reasoner* evolve through accumulated natural-language skills: failed cases are routed to dedicated *Proposer* and *Generator* agents on the *Reasoner* side to diagnose missing contextual knowledge and update skills accordingly, while easily solved cases are routed to the *Challenger* side to strengthen its task and rubric generation strategies, ensuring sustained adversarial pressure. The updated skills are carried forward into the next iteration, allowing both sides to continuously co-evolve through failure-driven textual feedback without any parameter updates. Furthermore, we identify a key risk in this multi-agent self-play loop, namely *adversarial collapse* phenomenon: as iterations progress, the *Challenger* may generate increasingly extreme tasks while the *Reasoner*'s skills over-specialize to these pathological cases, resulting in redundant skill accumulation that degrades generalization. To address this, we further propose a *Cross-time Replay* mechanism that selects the skill set that best balances performance across representative cases, ensuring robust and generalizable skill evolution. At inference time, the resulting context-specific *Reasoner* skills can be plugged into any LM to enhance its context learning capability on arbitrary tasks over the same context, including those unseen during the self-play loop.

We evaluate our *Ctx2Skill* on four context learning tasks from CL-Bench [12], including Domain Knowledge Reasoning, Rule System Application, Procedural Task Execution, and Empirical Discovery & Simulation. *Ctx2Skill* consistently improves solving rate across all four categories and multiple backbone models, e.g., lifting GPT-5.1 [27] from 21.1% to 25.8%, demonstrating that autonomously discovered context-specific skills provide substantial and generalizable gains.

2 Related Work

Context Learning. Current language models excel at tasks whose relevant knowledge was present during pre-training [9, 27, 33, 34]. However, many real-world tasks require models to learn from

complex contexts and leverage new knowledge to reason effectively, a capability termed *context learning* [12]. This context learning capability is essential in practice: a physician consults newly released clinical guidelines to adjust treatment, an engineer reads documentation to execute procedures, and a researcher induces patterns from experimental data to form hypotheses. However, prior work finds that LMs have yet to master this capability [12]. The core difficulty is that contextual knowledge spans diverse formats, from books to experimental datasets, and the relevant rules and procedures are often embedded implicitly, requiring deep comprehension rather than surface-level retrieval. *Ctx2Skill* directly targets this challenge by autonomously discovering reusable, context-specific skills from the contexts, equipping any LM to better learn from and reason over complex contexts.

Skills for LMs. Skills are natural-language modules that encode reusable procedural knowledge to augment LMs at inference time [2, 3], and skill augmentation has been widely validated across agent tasks [32], e.g., coding [24] and web navigation [42]. Early skill libraries are predominantly constructed through human annotation [18, 19, 20, 41], which is effective but does not scale, especially for context learning scenarios where contexts are long, technically dense, and domain-specific. Recent work has shifted toward automated skill construction. AutoSkill [44] extracts reusable behaviors from interaction traces as lifelong versioned artifacts, and AutoRefine [30] converts agent trajectories into reusable expertise. CoEvoSkills [45] generates multi-file skill packages and refines them through generator–verifier co-evolution, while EvoSkill [1] performs failure-driven refinement into structured skill folders. SkillX [40] distills the agent trajectories into a hierarchical skill knowledge base and iteratively refines them through execution feedback. However, these methods rely on external feedback signals [25], such as execution feedback, ground-truth comparison, or task-completion rewards, to evaluate and improve skill quality, which are not available in context learning scenarios where there is no automatic feedback. A separate line of work tries to internalize skills into model parameters. SKILL0 [23] uses in-context reinforcement learning (RL) to absorb skills, and SkillRL [43] builds a hierarchical skill bank via RL-guided distillation from teacher trajectories. These approaches require parameter access, making them inapplicable to closed-source models, and sacrificing the interpretability of natural-language skill documents. In this paper, we propose *Ctx2Skill*, a self-evolving framework designed to directly discover and evolve skills from complex contexts alone, which requires no human annotation, no external feedback, and no parameter updates.

3 METHODOLOGY

3.1 OVERVIEW

Context learning tasks [12] require LMs to leverage contextual knowledge from long, technically dense contexts to reason and solve tasks effectively. An intuitive paradigm for context learning tasks is inference-time skill augmentation, i.e., extracting the corresponding rules and procedures from ***context*** into natural-language ***skills***. Turning this idea into a working framework, however, faces two difficulties: (1) prohibitive cost for manual skill annotation and (2) lack of external feedback for automated skill construction. In this paper, we introduce ***Ctx2Skill*** shown in Figure 2, a self-evolving framework that autonomously discovers, refines, and selects context-specific skills without human supervision or external feedback. At its core, a *skill-optimized self-play* loop (§ 3.3) co-evolves two competing sides: a *Challenger* probes the context with tasks and rubrics guided by its own skill set, a *Reasoner* solves them under its current skill set, and a neutral *Judge* returns binary verdicts that route each outcome to a side-specific *Proposer–Generator* pair, which diagnoses the exposed weakness and edits that side's skills accordingly, so that both sides co-evolve without human annotation or external feedback. Furthermore, a key risk in this self-play framework is *adversarial collapse*: the *Challenger* may generate increasingly extreme tasks while the *Reasoner*'s skills over-specialize to these pathological cases, resulting in redundant skill accumulation that degrades generalization. To address this, we further propose a *Cross-Time Replay* mechanism (§ 3.4) that selects the skill set that best balances performance across representative cases, ensuring robust and generalizable skill evolution. In this way, *Ctx2Skill* autonomously constructs context-specific skills that can be plugged into any LM at inference time to enhance context learning capability without parameter updates.

3.2 PROBLEM FORMULATION

A context learning task [12] consists of a given context C whose content may lie outside the LM's pretraining corpus, a set of tasks $\mathcal{T} = \{t_j\}$ whose correct answers depend on C, and a set of binary

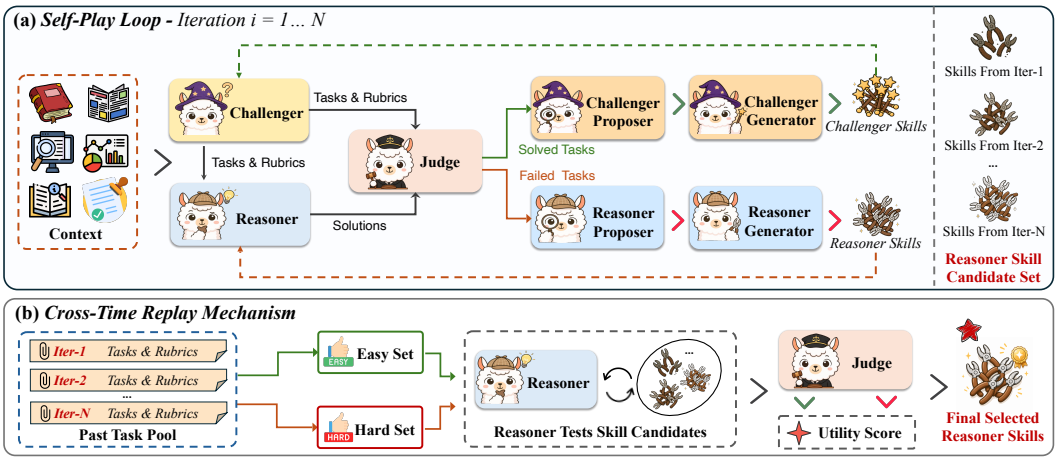

Figure 2: **The overview of *Ctx2Skill***. (a) In the self-play loop, the *Challenger* generates tasks and rubrics, the *Reasoner* tries to solve them, and the *Judge* routes outcomes to update *Challenger* and *Reasoner* skills. (b) Cross-Time Replay Mechanism re-evaluates historical *Reasoner* skill candidates on representative tasks, selecting the most balanced skill set for unseen downstream tasks.

rubrics $\mathcal{R}_j = \{r_{j,k}\}$. Given an answer a_j produced by an LM π, the task is considered solved only when every rubric passes, we define a solving indicator $y_j(\pi; C)$ for task t_j as:

$$y_j(\pi; C) = \prod_k \mathbb{I}\big[r_{j,k}(a_j) = \text{pass}\big], \qquad a_j \sim \pi(\cdot \mid C, t_j). \tag{1}$$

Our goal is to enable the LM π to solve tasks over a previously unseen context C without any parameter update or external feedback. To this end, we introduce a natural-language *skill set* $\mathcal{S}$, a short Markdown file pre-pended to the system prompt of the LM at inference time:

$$a_j \sim \pi(\cdot \mid \mathcal{S}, C, t_j). \tag{2}$$

During the skills construction of our proposed *Ctx2Skill*, $\mathcal{S}$ is instantiated as two separate skill sets: $\mathcal{S}^{\text{R}}$ for the *Reasoner* agent and $\mathcal{S}^{\text{C}}$ for the *Challenger* agent. At inference time, only the *Reasoner*'s skill set $\mathcal{S}^{\text{R}}$ is deployed with an LM for context-specific tasks evaluation.

3.3 *Ctx2Skill* Framework

Building a skill set for an unseen context C faces two obstacles: human annotation is costly to read C and write down its contextual knowledge, and no external feedback signal can tell us whether a proposed skill is faithful and useful to C. To address both, *Ctx2Skill* runs a multi-agent self-play loop that autonomously discovers and refines context-specific skills from context C alone.

The core idea of *Ctx2Skill* is to co-evolve two skill sets, skill set $\mathcal{S}^{\text{R}}$ for the *Reasoner* and skill set $\mathcal{S}^{\text{C}}$ for the *Challenger*, through N iterations of failure-driven textual edits, so that the *Reasoner* gradually accumulates the contextual knowledge of the context C while the *Challenger* keeps probing aspects not yet mastered. Five frozen-LM agent roles carry out this self-play loop. At iteration i, given the context C and skill set $\mathcal{S}^{\text{C}}_{i-1}$, the *Challenger* produces a batch of tasks with corresponding rubrics; the *Reasoner*, guided by context C and its skills $\mathcal{S}^{\text{R}}_{i-1}$, attempts each task; the neutral *Judge* returns per-task binary verdicts and partitions the batch into failed and solved tasks. Each group is then routed to its corresponding side through a *Proposer–Generator* pair: failed cases flow to the *Reasoner* side, where the *Proposer* diagnoses which contextual knowledge is missing and the *Generator* updates $\mathcal{S}^{\text{R}}$ accordingly; solved cases flow to the *Challenger* side, where the corresponding pair tightens $\mathcal{S}^{\text{C}}$ so that the next iteration sustains adversarial pressure. The updated $\mathcal{S}^{\text{R}}_i$ and $\mathcal{S}^{\text{C}}_i$ are carried into iteration $i{+}1$, and no prompt is ever conditioned on the opposing side's skill set, keeping the two sides strictly adversarial. We now define the five agent roles as follows at the iteration i:

- ***Challenger***: Given the context C and its current skill set $\mathcal{S}^{\text{C}}_{i-1}$, the *Challenger* produces a batch of M tasks $\{t_m\}_{m=1}^{M}$, each equipped with corresponding rubrics $\mathcal{R}_m$. The rubrics are designed

so that a correct answer requires inducing the rules of C rather than paraphrasing surface spans. By conditioning on its skills $\mathcal{S}^{\mathrm{C}}_{i-1}$, the *Challenger* adapts its probing strategy across iterations, sustaining adversarial pressure as the *Reasoner* improves.

- ***Reasoner***: Given C, a task t_m, and its current skill set $\mathcal{S}^{\mathrm{R}}_{i-1}$, the *Reasoner* produces an answer a_m. While the raw context C contains all necessary knowledge, it is often long and technically dense; the skill set $\mathcal{S}^{\mathrm{R}}_{i-1}$ distills the most relevant rules and procedures from C into a concise, structured form, allowing the *Reasoner* to directly apply contextual knowledge that would otherwise need to be extracted from scratch at every task.
- ***Judge***: Given a task and rubrics generated from the *Challenger*, and the generated answer from the *Reasoner*, the *Judge* returns a per-rubric binary verdict $z_{m,k} = \mathbb{I}[r_{m,k}(a_m) = \mathrm{pass}]$ and a solving indicator $y_m = \prod_k z_{m,k}$. The *Judge* is a strict evaluator: the task is considered solved if and only if the answer satisfies all associated rubrics.
- ***Proposer*** *(one per side)*: The *Judge*'s binary verdicts identify *which* tasks failed or solved, but not *why*. The *Proposer* therefore collects all routed cases of its side, where each case comprises the task and rubrics from the *Challenger*, and the *Reasoner*'s answer, and jointly analyzes them against the current skill set. Rather than diagnosing each case in isolation, the *Proposer* synthesizes common failure or success patterns across cases into a high-level diagnosis specifying an action (add or merge), the target skill name, a description, and a justification. The ***Reasoner Proposer*** is designed to take the failed cases $\mathcal{F}_i$ together with $\mathcal{S}^{\mathrm{R}}_{i-1}$, and diagnose which contextual knowledge is missing or misrepresented by examining both the incorrect answers and the rubrics they violated. On the other hand, the ***Challenger Proposer*** takes the easily solved cases $\mathcal{P}_i$ together with $\mathcal{S}^{\mathrm{C}}_{i-1}$, and identifies gaps in the current task and rubric generation strategy by inspecting how the *Reasoner* solved them with its existing skills.
- ***Generator*** *(one per side)*: The *Proposer*'s output is a high-level diagnosis describing what should change and why, but not the concrete skill content itself. The *Generator* materializes this diagnosis into an actual skill set. Given the diagnosis and the current skill set, it returns a complete replacement skill set that adds or merges entries while preserving every unrelated entry. The ***Reasoner Generator*** incorporates the diagnosed missing contextual knowledge into $\mathcal{S}^{\mathrm{R}}_{i-1}$ to produce the new skill set $\mathcal{S}^{\mathrm{R}}_{i}$. The ***Challenger Generator*** tightens $\mathcal{S}^{\mathrm{C}}_{i-1}$ so that the next iteration probes the *Reasoner*'s remaining weaknesses, producing skills $\mathcal{S}^{\mathrm{C}}_{i}$.

To summarize, at iteration i, the *Challenger* first produces a batch of tasks and rubrics $\{(t_m, \mathcal{R}_m)\}_{m=1}^{M}$ based on the context C; the *Reasoner* attempts to answer each t_m; the *Judge* evaluates every rubric and partitions the batch into failed cases $\mathcal{F}_i = \{m : y_m = 0\}$ and solved cases $\mathcal{P}_i = \{m : y_m = 1\}$ based on the solving indicator y_m. Each group is then routed to its corresponding side:

- The *Reasoner Proposer* jointly analyzes all failed cases $\mathcal{F}_i$ against $\mathcal{S}^{\mathrm{R}}_{i-1}$ and synthesizes a high-level diagnosis; the *Reasoner Generator* then materializes it into $\mathcal{S}^{\mathrm{R}}_{i}$.
- The *Challenger Proposer* jointly analyzes all solved cases $\mathcal{P}_i$ against $\mathcal{S}^{\mathrm{C}}_{i-1}$ and synthesizes a high-level diagnosis; the *Challenger Generator* then materializes it into $\mathcal{S}^{\mathrm{C}}_{i}$.

The updated skill sets $\mathcal{S}^{\mathrm{R}}_{i}$ and $\mathcal{S}^{\mathrm{C}}_{i}$ are carried forward into iteration $i+1$. No prompt is ever conditioned on the opposing side's skill set, keeping the two sides strictly adversarial.

3.4 *Cross-Time Replay* MECHANISM

The framework of *Ctx2Skill* detailed in § 3.3 intentionally strengthens the *Challenger* across iterations through updates to $\mathcal{S}^{\mathrm{C}}$, sustaining adversarial pressure as the *Reasoner* improves. However, this design introduces an inherent tension that we term *adversarial collapse*. As iterations progress, the *Challenger* generates increasingly extreme tasks that concentrate on the *Reasoner*'s residual weaknesses, gradually deviating from the representative knowledge of the context C. Since the *Reasoner*'s skill updates are failure-driven, its skills over-specialize to these pathological cases, resulting in redundant skill accumulation that degrades generalization. Moreover, this collapse is undetectable within the loop: each iteration's *Judge* only evaluates the *Challenger*'s newly generated tasks, providing no signal on whether contextual knowledge mastered in earlier iterations has been corrupted

by subsequent edits. Returning the last iteration *Reasoner*'s skills $\mathcal{S}_N^{\mathrm{R}}$ unconditionally is therefore unreliable; we instead select the most generalizable skill set among candidates $\{\mathcal{S}_1^{\mathrm{R}}, \ldots, \mathcal{S}_N^{\mathrm{R}}\}$.

To address this, we propose a *cross-time replay* mechanism that selects the most generalizable skill set from the candidates $\{\mathcal{S}_1^{\mathrm{R}}, \ldots, \mathcal{S}_N^{\mathrm{R}}\}$ using representative probes collected during self-play itself. Two small probe sets are curated incrementally without any external supervision: at each iteration i, the failed task with the lowest rubric pass rate is added to the hard probe set $\mathcal{Q}^{\mathrm{h}}$, and the solved task with the fewest rubrics is added to the easy probe set $\mathcal{Q}^{\mathrm{e}}$, capturing the hardest failure and the easiest success observed at each iteration. Once the self-play loop terminates, the *Reasoner* π^{R} re-answers every task in both probe sets under each candidate skill set $\mathcal{S}_i^{\mathrm{R}}$, and the *Judge* re-evaluates every rubric, yielding the solving indicator y_q (i.e., whether all rubrics pass) for each probe task q. We define the Laplace-smoothed [13] solving rates $\rho^{\mathrm{h}}(i)$ and $\rho^{\mathrm{e}}(i)$ as the proportion of tasks solved in the hard and easy probe sets under the context C and skills $\mathcal{S}_i^{\mathrm{R}}$, respectively:

$$\rho^{h}(i) = \frac{\sum_{q \in \mathcal{Q}^{\mathrm{h}}} y_q(\pi^{\mathrm{R}}; C, \mathcal{S}_i^{\mathrm{R}}) + 1}{|\mathcal{Q}^{\mathrm{h}}| + 1}, \qquad \rho^{e}(i) = \frac{\sum_{q \in \mathcal{Q}^{\mathrm{e}}} y_q(\pi^{\mathrm{R}}; C, \mathcal{S}_i^{\mathrm{R}}) + 1}{|\mathcal{Q}^{\mathrm{e}}| + 1}. \tag{3}$$

The selected skill set is the one that maximizes the product of these two solving rates:

$$\mathcal{S}_\star^{\mathrm{R}} = \mathcal{S}_{i^\star}^{\mathrm{R}}, \qquad i^\star = \arg\max_i \left(\rho^{\mathrm{h}}(i) \cdot \rho^{\mathrm{e}}(i)\right). \tag{4}$$

The multiplicative form is essential: a skill set that resolves late, idiosyncratic failures at the cost of regressing on easier tasks incurs a penalty on $\rho^{\mathrm{e}}(i)$ and is rejected; conversely, one that solves every easy probe but fails every hard one is rejected through $\rho^{\mathrm{h}}(i)$. Meanwhile, Laplace smoothing keeps the product well-defined when a probe set is empty for a given iteration

At inference time, the final skill set $\mathcal{S}_\star^{\mathrm{R}}$ is prepended to the *Reasoner*'s system prompt for any unseen task t_u over context C, and the *Reasoner* answers conditioned on $(\mathcal{S}_\star^{\mathrm{R}}, C, t_u)$. Since $\mathcal{S}_\star^{\mathrm{R}}$ encodes reusable contextual knowledge of C rather than task-specific solutions, it generalizes to arbitrary tasks over the same context. Moreover, $\mathcal{S}_\star^{\mathrm{R}}$ is produced once per context and reused across all $|\mathcal{T}|$ unseen tasks, so the *Ctx2Skill* cost amortizes over $|\mathcal{T}|$ with no additional cost per deployed query.

4 Experiments

In this section, we conduct experiments and analyses to show the advantages of our *Ctx2Skill*.

4.1 Settings

Evaluation. Context learning is a newly emerging research topic, aiming to require LMs to learn from unseen context and leverage new knowledge beyond what was acquired during pre-training to reason and solve tasks [12]. This goes far beyond long-context tasks [6, 35] that primarily test retrieval or reading comprehension, and in-context learning tasks [11], where models learn simple task patterns via instructions and demonstrations. We select the CL-bench [12] for our evaluation, which comprises 500 complex contexts, 1,899 tasks, and 31,607 verification rubrics, all crafted by experienced domain experts. Each task is designed such that the new content required to resolve it is contained within the corresponding context. These data are distributed across four categories: *Domain Knowledge Reasoning*, *Rule System Application*, *Procedural Task Execution*, and *Empirical Discovery & Simulation*. Scoring is strictly all-or-nothing: a task is considered solved only when every rubric passes. We report both per-category and overall solving rates. Rubric-level grading follows the original benchmark protocol, which employs GPT-5.1 as the LLM-as-a-judge verifier [50]; cross-verifier and human agreement with this verifier both exceed 90% [12].

Baselines. We first evaluate a broad set of frontier LMs, including GPT-4.1 [29], GPT-5.1 [27], GPT-5.2 [28], Claude Opus 4.5 [4], Gemini 3 Pro [8], Kimi K2.5 [38], and DeepSeek V3.2 [9]. For models that support thinking mode, we use the default reasoning effort. For skill-based methods, we compare *Ctx2Skill* with two baselines: *(1) Prompting* directly prompts the LM to read the context C and generate a skill set in a single pass; the resulting skills are then pre-pended to LMs during inference. *(2) AutoSkill4Doc* is a variant of AutoSkill [44] designed for document-level contexts: it partitions the context C into different windows, identifies the independent skills within each window, and recombines skills across windows to produce a skill set. Both baselines use the same backbone

Table 1: **The main results of *Ctx2Skill*.** We report the task-solving rate on the CL-bench across four task categories. The best results within the same backbone are marked in **bold**. ◐ represents methods using inference time augmentation, and ○ shows inference results without skills. Red and Green show the performance gains and declines, respectively, relative to the models without skills.

Model	Overall (%)	Domain Knowledge Reasoning (%)	Rule System Application (%)	Procedural Task Execution (%)	Empirical Discovery & Simulation (%)
Frontier LMs					
○ GPT-5.1	21.1	22.4	21.0	22.8	13.6
○ Claude Opus 4.5	21.0	23.7	19.0	22.6	15.1
○ Kimi K2.5	19.2	19.1	19.4	21.3	14.4
○ GPT-5.2	18.2	19.5	18.0	19.1	12.1
○ Gemini 3 Pro	15.8	15.5	17.7	16.4	10.1
○ DeepSeek V3.2 Thinking	13.2	13.6	13.8	14.2	8.0
○ GPT-4.1	11.1	10.6	14.8	10.4	4.6
GPT-4.1-based Methods					
◐ Prompting	12.3 +1.2	12.4 +1.8	12.3 -2.5	13.9 +3.5	8.2 +3.6
◐ AutoSkill4Doc	13.2 +2.1	13.3 +2.7	13.1 -1.7	15.0 +4.6	8.7 +4.1
◐ ***Ctx2Skill***	**16.5** +5.4	**16.8** +6.2	**17.6** +2.8	**17.6** +7.2	**9.7** +5.1
GPT-5.1-based Methods					
◐ Prompting	22.1 +1.0	24.7 +2.3	21.1 +0.1	22.4 -0.4	15.5 +1.9
◐ AutoSkill4Doc	22.7 +1.6	25.3 +2.9	21.5 +0.5	23.1 +0.3	16.0 +2.4
◐ ***Ctx2Skill***	**25.8** +4.7	**27.9** +5.5	**24.9** +3.9	**26.9** +4.1	**19.1** +5.5
GPT-5.2-based Methods					
◐ Prompting	19.1 +0.9	19.6 +0.1	18.3 +0.3	22.6 +3.5	11.1 -1.0
◐ AutoSkill4Doc	19.7 +1.5	20.5 +1.0	18.8 +0.8	23.0 +3.9	11.6 -0.5
◐ ***Ctx2Skill***	**21.4** +3.2	**22.2** +2.7	**20.4** +2.4	**25.4** +6.3	**12.6** +0.5

for skill generation and inference as *Ctx2Skill*: for example, all GPT-4.1-based methods use GPT-4.1 for both skill construction and evaluation, ensuring a fair comparison.

Implementation. We run $N{=}5$ self-play iterations per context with $M{=}5$ tasks per iteration. The *Challenger*, *Reasoner*, *Proposer*, and *Generator* all use the same backbone as the corresponding series (e.g., GPT-4.1 for all four agents in the GPT-4.1-based methods), while the *Judge* uses GPT-5.1, consistent with the CL-bench evaluation protocol [12]. Due to API budget constraints, we do not explore larger N or M; nonetheless, the most effective skill set is typically selected from early iterations (§ 4.3), and $M{=}5$ tasks per iteration provide enough cases for the *Proposer* to synthesize meaningful diagnostic patterns.

Table 2: **The quality evaluation of generated skills.** We use GPT-4.1 as a judge to assess skills across five dimensions: conciseness, faithfulness, clarity, effectiveness, and reusability. The best result in each dimension is marked in **bold**.

Model	Conc.	Faith.	Clar.	Eff.	Reus.	Avg.
GPT-4.1-based Methods						
Prompting	81.2	79.7	80.0	83.3	84.7	81.8
AutoSkill4Doc	81.3	81.4	92.4	88.7	87.2	86.2
Ctx2Skill	**85.2**	**84.8**	**96.2**	**90.5**	**92.5**	**89.8**
GPT-5.1-based Methods						
Prompting	80.7	88.5	94.1	94.6	95.3	90.7
AutoSkill4Doc	82.0	89.1	94.2	96.0	96.0	91.5
Ctx2Skill	**82.6**	**93.9**	**98.1**	**96.7**	**96.9**	**93.6**
GPT-5.2-based Methods						
Prompting	80.5	84.4	92.0	90.4	92.0	87.9
AutoSkill4Doc	82.2	87.0	94.0	92.2	93.0	89.7
Ctx2Skill	**85.1**	**89.6**	**96.5**	**94.7**	**94.4**	**92.0**

4.2 Results

Table 1 presents the main results on CL-bench. Context learning remains challenging for current frontier LMs: even the best-performing model, GPT-5.1, achieves only a 21.1% overall solving rate. *Ctx2Skill* consistently improves solving rates across all three backbones, raising GPT-4.1 from 11.1% to 16.5% (+5.4%), GPT-5.1 from 21.1% to 25.8% (+4.7%), and GPT-5.2 from 18.2% to 21.4% (+3.2%), outperforming both *Prompting* and *AutoSkill4Doc* by a large margin across all four categories. The gains are especially pronounced on Procedural Task Execution and Empirical Discovery & Simulation, which demand deeper procedural and inductive reasoning over the context. In contrast, the two baselines provide only modest improvements and occasionally degrade individual categories (e.g., *Prompting* decreases Rule System Application on GPT-4.1 by 2.5%), suggesting that single-pass skill extraction is insufficient for complex contextual knowledge. Notably, GPT-4.1 with *Ctx2Skill* skills (16.5%) surpasses stronger frontier models without skills, e.g., Gemini 3 Pro

Table 3: **The analysis results of *Ctx2Skill*.** We report the task-solving rate on the CL-bench across four task categories under different ablations, variant designs, etc. The best results within each analysis block are marked in **bold**. ◐ represents using inference-time skill augmentation, and ○ indicates inference results without skills.

Model	Overall (%)	Domain Knowledge Reasoning (%)	Rule System Application (%)	Procedural Task Execution (%)	Empirical Discovery & Simulation (%)
Ablation Study					
○ Base Model - GPT-4.1	11.1	10.6	14.8	10.4	4.6
◐ ***Ctx2Skill* - GPT-4.1**	**16.5**	**16.8**	**17.6**	**17.6**	**9.7**
-w/o Cross-Time Replay Mechanism	14.7	15.1	14.9	16.1	9.2
-w/o Decoupling Proposer and Generator	15.9	16.8	16.9	16.1	9.2
-w/o Challenger Skills Evolving	13.8	13.9	13.9	15.7	8.7
-w/o Easy Probe Set	15.7	15.7	16.3	17.4	9.7
-w/o Hard Probe Set	15.2	15.6	15.6	16.5	9.2
-w/o Laplace Smoothing	15.5	15.7	16.0	17.2	9.2
○ Base Model - GPT-5.1	21.1	22.4	21.0	22.8	13.6
◐ ***Ctx2Skill* - GPT-5.1**	**25.8**	**27.9**	**24.9**	**26.9**	**19.1**
-w/o Cross-Time Replay Mechanism	23.0	25.6	22.0	23.3	16.0
-w/o Decoupling Proposer and Generator	25.1	27.0	24.5	25.9	18.6
-w/o Challenger Skills Evolving	22.5	25.0	21.3	23.1	16.0
-w/o Easy Probe Set	24.7	26.8	24.0	25.2	18.0
-w/o Hard Probe Set	24.2	26.7	23.6	23.9	18.0
-w/o Laplace Smoothing	25.2	27.0	24.7	25.9	19.1
Variant Designs Testing					
○ Base Model - GPT-4.1	11.1	10.6	14.8	10.4	4.6
◐ ***Ctx2Skill* - GPT-4.1**	**16.5**	**16.8**	**17.6**	**17.6**	**9.7**
◐ Loser-Only Skill Update	16.0	15.9	16.9	17.6	9.7
◐ Joint Outcome Skill Update	15.5	16.0	16.9	16.1	8.2
◐ Using Additive Scoring in Eq.(4)	15.9	16.6	17.1	16.5	8.2
Effect of Cross-Time Replay Mechanism					
○ Base Model - GPT-4.1	11.1	10.6	14.8	10.4	4.6
◐ ***Ctx2Skill* - GPT-4.1**	**16.5**	**16.8**	**17.6**	**17.6**	**9.7**
◐ Using Skills from Iter-1	15.9	15.9	16.7	17.4	9.7
◐ Using Skills from Iter-2	15.6	15.7	16.2	17.2	9.7
◐ Using Skills from Iter-3	15.6	15.9	16.2	17.0	9.7
◐ Using Skills from Iter-4	15.2	15.6	16.7	15.4	8.7
◐ Using Skills from Iter-5	14.7	15.1	14.9	16.1	9.2
Skill Transferability Testing					
○ Base Model - GPT-4.1	11.1	10.6	14.8	10.4	4.6
◐ GPT-4.1 with Skills from GPT-5.1	16.1	16.6	17.2	17.2	8.2
◐ GPT-4.1 with Skills from GPT-4.1	**16.5**	**16.8**	**17.6**	**17.6**	**9.7**
○ Base Model - GPT-5.1	21.1	22.4	21.0	22.8	13.6
◐ GPT-5.1 with Skills from GPT-4.1	23.1	24.9	22.4	24.1	16.5
◐ GPT-5.1 with Skills from GPT-5.1	**25.8**	**27.9**	**24.9**	**26.9**	**19.1**

(15.8%), demonstrating that context-specific skills can bridge substantial capability gaps. Beyond solving rate, we assess the intrinsic quality of generated skills using GPT-4.1 as a judge across five dimensions (Table 2). *Ctx2Skill* achieves the highest average score across all three backbones, outperforming *AutoSkill4Doc* by +3.6, +2.1, and +2.3 on GPT-4.1, GPT-5.1, and GPT-5.2, with the most notable improvements in faithfulness and clarity. These results indicate that the iterative self-play loop produces skills that not only improve downstream reasoning but also present contextual knowledge in a well-structured, human-readable form amenable to inspection, editing, and reuse.

4.3 Analysis

Ablation Study. We ablate each component of *Ctx2Skill* on both the GPT-4.1 and GPT-5.1 (Table 3, Ablation Study block). Removing *Challenger* skills evolving causes the largest drop, confirming that sustained adversarial pressure is essential for the *Reasoner* to discover contextual knowledge progressively. The *Cross-Time Replay* mechanism is the second most impactful: without it, the last-iteration skills suffer from adversarial collapse. Within *Cross-Time Replay*, the hard probe set contributes more than the easy probe set, and removing Laplace smoothing also degrades performance. Merging the *Proposer* and *Generator* into a single agent yields a modest but consistent decline, supporting the decoupling of diagnosis from skill materialization. We also examine the impact across all sub-categories of CL-bench on GPT-4.1 (Figure 3). *Ctx2Skill* improves solving rates on the vast majority of sub-categories, with especially large gains on workflow orchestration (+11.8%).

Variant Designs Testing. We examine three alternative designs on GPT-4.1 (Table 3, Variant Designs Testing block). *Loser-Only Skill Update* updates only the losing side each iteration; the 0.5% drop

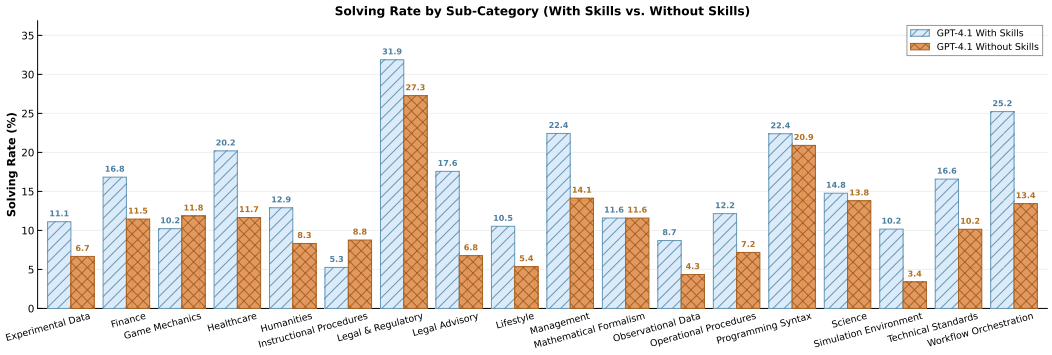

Figure 3: **Per sub-category solving rate on CL-bench.** *Ctx2Skill* improves solving rates on the vast majority of sub-categories compared with the base model without skills.

indicates that updating both sides yields more effective co-evolution. *Joint Outcome Skill Update* feeds both failed and solved cases into both sides simultaneously, allowing each side to learn from both positive and negative examples; the larger 1.0% degradation suggests that mixing outcomes dilutes the diagnostic signal. We also test *Additive Scoring*, which replaces $\rho^{\mathrm{h}}(i) \cdot \rho^{\mathrm{e}}(i)$ with $\rho^{\mathrm{h}}(i) + \rho^{\mathrm{e}}(i)$ in Eq. (4); the 0.6% drop confirms that the multiplicative form better penalizes skill sets that sacrifice easy-probe performance for hard-probe gains.

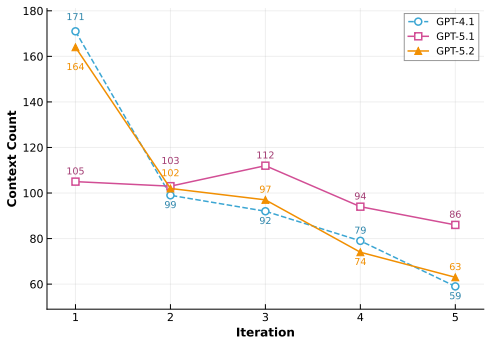

Figure 4: **Distribution of selected iterations by the *Cross-Time Replay*.** We report the number of contexts whose final skill set is selected from each iteration across all contexts.

Effect of *Cross-Time Replay* Mechanism. We compare *Cross-Time Replay* against using a fixed iteration's skills (Table 3, Effect of Cross-Time Replay block). On GPT-4.1, fixed-iteration performance monotonically decreases from Iter-1 (15.9%) to Iter-5 (14.7%), confirming that later iterations suffer from adversarial collapse phenomenon. *Cross-Time Replay* (16.5%) outperforms every fixed iteration, including the best one (Iter-1), by $+0.6\%$, by adaptively selecting the most balanced skill set per context rather than committing to a single iteration across all contexts. Figure 4 corroborates this pattern: early iterations are selected most frequently across all three backbones, while a non-trivial proportion of later iterations indicates that certain contexts with more complex knowledge structures do benefit from additional self-play rounds.

Skill Transferability Testing. We investigate whether skills generated by one backbone can benefit a different one (Table 3, Skill Transferability Testing block). GPT-5.1-generated skills applied to GPT-4.1 yield 16.1%, nearly matching GPT-4.1's own skills; conversely, GPT-4.1-generated skills applied to GPT-5.1 yield 23.1%, a $+2.0\%$ gain but notably below the $+4.6\%$ from GPT-5.1's own skills. This asymmetry shows that stronger models produce skills that transfer well to weaker models, whereas weaker models lack the capacity to discover knowledge that stronger models can exploit.

Case Study & Discussion. We provide a case study in Appendix C. We also discuss potential concerns about *Ctx2Skill*, e.g., reward hacking, in Appendix D.

5 Conclusion

We presented *Ctx2Skill*, a self-evolving framework that autonomously discovers, refines, and selects context-specific skills from complex contexts without human annotation or external feedback. Through a skill-optimized self-play loop, a *Challenger* and a *Reasoner* co-evolve their skill sets via failure-driven textual edits, while a *Cross-Time Replay* mechanism prevents adversarial collapse by selecting the most generalizable skill set across iterations. Experiments on CL-bench demonstrate that *Ctx2Skill* consistently and substantially improves context learning performance across multiple backbone models and task categories, and that the resulting skills are transferable across models.

AI USE STATEMENT

In this work, we used generative AI tools for editing the research paper to improve readability and for creating or modifying scientific figures or images. We have not used generative AI tools for generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating mathematical claims or assisting with their proofs, proposing or refining hypotheses, designing or providing feedback on the research methodology or experiments, implementing methods, cleaning or reformatting datasets, supporting qualitative or thematic data analysis, or interpreting results, and translation is not applicable to this work. Additionally, we did not use generative AI tools for any other task with recommended disclosure, such as creating or editing software code, summarizing or analyzing existing literature, formatting references, or brainstorming. We have reviewed all AI-assisted work: all AI-edited text was checked by the authors against our original drafts to ensure that the technical content, claims, and reported numbers remain exactly as intended, and all AI-assisted figures were verified by the authors to faithfully depict the proposed method and experimental setup. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

[1] Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. 2026. Evoskill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*.

[2] Anthropic. 2025. Equipping agents for the real world with agent skills.

[3] Anthropic. 2025. Introduction to agent skills.

[4] Anthropic. 2025. System card: Claude opus 4.5.

[5] Anthropic. 2026. System card: Claude opus 4.6.

[6] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

[7] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

[8] Google DeepMind. 2025. Gemini 3 pro model card.

[9] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li,

Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

[10] Anthony DiGiovanni and Ethan C. Zell. 2021. Survey of self-play in reinforcement learning. *Preprint*, arXiv:2107.02850.

[11] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A survey on in-context learning. *Preprint*, arXiv:2301.00234.

[12] Shihan Dou, Ming Zhang, Zhangyue Yin, Chenhao Huang, Yujiong Shen, Junzhe Wang, Jiayi Chen, Yuchen Ni, Junjie Ye, Cheng Zhang, Huaibing Xie, Jianglu Hu, Shaolei Wang, Weichao Wang, Yanling Xiao, Yiting Liu, Zenan Xu, Zhen Guo, Pluto Zhou, Tao Gui, Zuxuan Wu, Xipeng Qiu, Qi Zhang, Xuanjing Huang, Yu-Gang Jiang, Di Wang, and Shunyu Yao. 2026. CL-Bench: A benchmark for context learning. *arXiv preprint arXiv:2602.03587*.

[13] David A Field. 1988. Laplacian smoothing and delaunay triangulations. *Communications in applied numerical methods*, 4(6):709–712.

[14] Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. 2025. Omni-MATH: A universal olympiad level mathematic benchmark for large language models. In *The Thirteenth International Conference on Learning Representations*.

[15] Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. 2025. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *Preprint*, arXiv:2507.17746.

[16] Qishuo Hua, Lyumanshan Ye, Dayuan Fu, Yang Xiao, Xiaojie Cai, Yunze Wu, Jifan Lin, Junfei Wang, and Pengfei Liu. 2025. Context engineering 2.0: The context of context engineering. *Preprint*, arXiv:2510.26493.

[17] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2024. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*.

[18] Hao Li, Chunjiang Mu, Jianhao Chen, Siyue Ren, Zhiyao Cui, Yiqun Zhang, Lei Bai, and Shuyue Hu. 2026. Organizing, orchestrating, and benchmarking agent skills at ecosystem scale. *Preprint*, arXiv:2603.02176.

[19] Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, Yubo Li, Bingran You, Haotian Shen, Jiankai Sun, Shuyi Wang, Binxu Li, Qunhong Zeng, Di Wang, Xuandong Zhao, Yuanli Wang, Roey Ben Chaim, Zonglin Di, Yipeng Gao, Junwei He, Yizhuo He, Liqiang Jing, Luyang Kong, Xin Lan, Jiachen Li, Songlin Li, Yijiang Li, Yueqian Lin, Xinyi Liu, Xuanqing Liu, Haoran Lyu, Ze Ma, Bowei Wang, Runhui Wang, Tianyu Wang, Wengao Ye, Yue Zhang, Hanwen Xing, Yiqi Xue, Steven Dillmann, and Han chung Lee. 2026. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *Preprint*, arXiv:2602.12670.

[20] Yuan Liang, Ruobin Zhong, Haoming Xu, Chen Jiang, Yi Zhong, Runnan Fang, Jia-Chen Gu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Xin Xu, Tongtong Wu, Kun Wang, Yang Liu, Zhen Bi, Jungang Lou, Yuchen Eleanor Jiang, Hangcheng Zhu, Gang Yu, Haiwen Hong, Longtao Huang, Hui Xue, Chenxi Wang, Yijun Wang, Zifei Shan, Xi Chen, Zhaopeng Tu, Feiyu Xiong, Xin Xie, Peng Zhang, Zhengke Gui, Lei Liang, Jun Zhou, Chiyu Wu, Jin Shang, Yu Gong, Junyu Lin, Changliang Xu, Hongjie Deng, Wen Zhang, Keyan Ding, Qiang Zhang, Fei Huang, Ningyu Zhang, Jeff Z. Pan, Guilin Qi, Haofen Wang, and Huajun Chen. 2026. Skillnet: Create, evaluate, and connect ai skills. *Preprint*, arXiv:2603.04448.

[21] Bo Liu, Leon Guertler, Simon Yu, Zichen Liu, Penghui Qi, Daniel Balcells, Mickel Liu, Cheston Tan, Weiyan Shi, Min Lin, Wee Sun Lee, and Natasha Jaques. 2026. Spiral: Self-play on zero-sum games incentivizes reasoning via multi-agent multi-turn reinforcement learning. *Preprint*, arXiv:2506.24119.

[22] Hongliang Lu, Yuhang Wen, Pengyu Cheng, Ruijin Ding, Jiaqi Guo, Haotian Xu, Chutian Wang, Haonan Chen, xiaoxi jiang, and guanjunjiang. 2026. Search self-play: Pushing the frontier of agent capability without supervision. In *The Fourteenth International Conference on Learning Representations*.

[23] Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Chengcheng Han, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. 2026. SKILL0: In-context agentic reinforcement learning for skill internalization. *arXiv preprint arXiv:2604.02268*.

[24] Yingwei Ma, Yue Liu, Xinlong Yang, Yanhao Li, Kelin Fu, Yibo Miao, Yuchong Xie, Zhexu Wang, and Shing-Chi Cheung. 2026. Scaling coding agents via atomic skills. *Preprint*, arXiv:2604.05013.

[25] Ziyu Ma, Shidong Yang, Yuxiang Ji, Xucong Wang, Yong Wang, Yiming Hu, Tongwen Huang, and Xiangxiang Chu. 2026. Skillclaw: Let skills evolve collectively with agentic evolver. *Preprint*, arXiv:2604.08377.

[26] Lingrui Mei, Jiayu Yao, Yuyao Ge, Yiwei Wang, Baolong Bi, Yujun Cai, Jiazhi Liu, Mingyu Li, Zhong-Zhi Li, Duzhen Zhang, Chenlin Zhou, Jiayi Mao, Tianze Xia, Jiafeng Guo, and Shenghua Liu. 2025. A survey of context engineering for large language models. *ArXiv*, abs/2507.13334.

[27] OpenAI. 2025. Gpt-5 technical report.

[28] OpenAI. 2025. Update to gpt-5 system card: Gpt-5.2.

[29] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov,

Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

[30] Libin Qiu, Zhirong Gao, Junfu Chen, Yuhang Ye, Weizhi Huang, Xiaobo Xue, Wenkai Qiu, and Shuo Tang. 2026. Autorefine: From trajectories to reusable expertise for continual llm agent refinement. *arXiv preprint arXiv:2601.22758*.

[31] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

[32] Shuzheng Si, Wentao Ma, Haoyu Gao, Yuchuan Wu, Ting-En Lin, Yinpei Dai, Hangyu Li, Rui Yan, Fei Huang, and Yongbin Li. 2023. SpokenWOZ: A large-scale speech-text benchmark for spoken task-oriented dialogue agents. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[33] Shuzheng Si, Qingyi Wang, Haozhe Zhao, Yuzhuo Bai, Guanqiao Chen, Kangyang Luo, Gang Chen, Fanchao Qi, Minjia Zhang, Baobao Chang, and Maosong Sun. 2026. Faithlens: Detecting and explaining faithfulness hallucination. *Preprint*, arXiv:2512.20182.

[34] Shuzheng Si, Haozhe Zhao, Gang Chen, Cheng Gao, Yuzhuo Bai, Zhitong Wang, Kaikai An, Kangyang Luo, Chen Qian, Fanchao Qi, Baobao Chang, and Maosong Sun. 2025. Aligning large language models to follow instructions and hallucinate less via effective data filtering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16469–16488, Vienna, Austria. Association for Computational Linguistics.

[35] Shuzheng Si, Haozhe Zhao, Gang Chen, Yunshui Li, Kangyang Luo, Chuancheng Lv, Kaikai An, Fanchao Qi, Baobao Chang, and Maosong Sun. 2025. GATEAU: Selecting influential samples for long context alignment. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7391–7422, Suzhou, China. Association for Computational Linguistics.

[36] Shuzheng Si, Haozhe Zhao, Cheng Gao, Yuzhuo Bai, Zhitong Wang, Bofei Gao, Kangyang Luo, Wenhao Li, Yufei Huang, Gang Chen, Fanchao Qi, Minjia Zhang, Baobao Chang, and Maosong Sun. 2026. Teaching large language models to maintain contextual faithfulness via synthetic tasks and reinforcement learning. In *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pages 33001–33009. AAAI Press.

[37] Shuzheng Si, Haozhe Zhao, Kangyang Luo, Gang Chen, Fanchao Qi, Minjia Zhang, Baobao Chang, and Maosong Sun. 2025. A goal without a plan is just a wish: Efficient and effective global planner training for long-horizon agent tasks. *Preprint*, arXiv:2510.05608.

[38] Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, Y. Charles, H. S. Che, Cheng Chen, Guanduo Chen, Huarong Chen, Jia Chen, Jiahao Chen, Jianlong Chen, Jun Chen,

Kefan Chen, Liang Chen, Ruijue Chen, Xinhao Chen, Yanru Chen, Yanxu Chen, Yicun Chen, Yimin Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Ziwei Chen, Dazhi Cheng, Minghan Chu, Jialei Cui, Jiaqi Deng, Muxi Diao, Hao Ding, Mengfan Dong, Mengnan Dong, Yuxin Dong, Yuhao Dong, Angang Du, Chenzhuang Du, Dikang Du, Lingxiao Du, Yulun Du, Yu Fan, Shengjun Fang, Qiulin Feng, Yichen Feng, Garimugai Fu, Kelin Fu, Hongcheng Gao, Tong Gao, Yuyao Ge, Shangyi Geng, Chengyang Gong, Xiaochen Gong, Zhuoma Gongque, Qizheng Gu, Xinran Gu, Yicheng Gu, Longyu Guan, Yuanying Guo, Xiaoru Hao, Weiran He, Wenyang He, Yunjia He, Chao Hong, Hao Hu, Jiaxi Hu, Yangyang Hu, Zhenxing Hu, Ke Huang, Ruiyuan Huang, Weixiao Huang, Zhiqi Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yu Jing, Guokun Lai, Aidi Li, C. Li, Cheng Li, Fang Li, Guanghe Li, Guanyu Li, Haitao Li, Haoyang Li, Jia Li, Jingwei Li, Junxiong Li, Lincan Li, Mo Li, Weihong Li, Wentao Li, Xinhang Li, Xinhao Li, Yang Li, Yanhao Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zheming Li, Weilong Liao, Jiawei Lin, Xiaohan Lin, Zhishan Lin, Zichao Lin, Cheng Liu, Chenyu Liu, Hongzhang Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Tianyu Liu, Weizhou Liu, Xiangyan Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yuanxin Liu, Yue Liu, Zhengying Liu, Zhongnuo Liu, Enzhe Lu, Haoyu Lu, Zhiyuan Lu, Junyu Luo, Tongxu Luo, Yashuo Luo, Long Ma, Yingwei Ma, Shaoguang Mao, Yuan Mei, Xin Men, Fanqing Meng, Zhiyong Meng, Yibo Miao, Minqing Ni, Kun Ouyang, Siyuan Pan, Bo Pang, Yuchao Qian, Ruoyu Qin, Zeyu Qin, Jiezhong Qiu, Bowen Qu, Zeyu Shang, Youbo Shao, Tianxiao Shen, Zhennan Shen, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Feifan Song, Pengwei Song, Tianhui Song, Xiaoxi Song, Hongjin Su, Jianlin Su, Zhaochen Su, Lin Sui, Jinsong Sun, Junyao Sun, Tongyu Sun, Flood Sung, Yunpeng Tai, Chuning Tang, Heyi Tang, Xiaojuan Tang, Zhengyang Tang, Jiawen Tao, Shiyuan Teng, Chaoran Tian, Pengfei Tian, Ao Wang, Bowen Wang, Chensi Wang, Chuang Wang, Congcong Wang, Dingkun Wang, Dinglu Wang, Dongliang Wang, Feng Wang, Hailong Wang, Haiming Wang, Hengzhi Wang, Huaqing Wang, Hui Wang, Jiahao Wang, Jinhong Wang, Jiuzheng Wang, Kaixin Wang, Linian Wang, Qibin Wang, Shengjie Wang, Shuyi Wang, Si Wang, Wei Wang, Xiaochen Wang, Xinyuan Wang, Yao Wang, Yejie Wang, Yipu Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhexu Wang, Zihan Wang, Zizhe Wang, Chu Wei, Ming Wei, Chuan Wen, Zichen Wen, Chengjie Wu, Haoning Wu, Junyan Wu, Rucong Wu, Wenhao Wu, Yuefeng Wu, Yuhao Wu, Yuxin Wu, Zijian Wu, Chenjun Xiao, Jin Xie, Xiaotong Xie, Yuchong Xie, Yifei Xin, Bowei Xing, Boyu Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinbo Xu, Xinran Xu, Yangchuan Xu, Yichang Xu, Yuemeng Xu, Zelai Xu, Ziyao Xu, Junjie Yan, Yuzi Yan, Guangyao Yang, Hao Yang, Junwei Yang, Kai Yang, Ningyuan Yang, Ruihan Yang, Xiaofei Yang, Xinlong Yang, Ying Yang, Yi Yang, Yi Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Dan Ye, Wenjie Ye, Zhuorui Ye, Bohong Yin, Chengzhen Yu, Longhui Yu, Tao Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Xiaokun Yuan, Yang Yue, Weihao Zeng, Dunyuan Zha, Haobing Zhan, Dehao Zhang, Hao Zhang, Jin Zhang, Puqi Zhang, Qiao Zhang, Rui Zhang, Xiaobin Zhang, Y. Zhang, Yadong Zhang, Yangkun Zhang, Yichi Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yushun Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Chenguang Zhao, Feifan Zhao, Jinxiang Zhao, Shuai Zhao, Xiangyu Zhao, Yikai Zhao, Zijia Zhao, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Junfeng Zhong, Longguang Zhong, Weiming Zhong, M. Zhou, Runjie Zhou, Xinyu Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yuxuan Zhu, Zhen Zhu, Jingze Zhuang, Weiyu Zhuang, Ying Zou, and Xinxing Zu. 2026. Kimi k2.5: Visual agentic intelligence. *Preprint*, arXiv:2602.02276.

[39] Qwen Team. 2026. Qwen3. 5: Towards native multimodal agents. *URL: https://qwen. ai/blog*.

[40] Chenxi Wang, Zhuoyun Yu, Xin Xie, Wuguannan Yao, Runnan Fang, Shuofei Qiao, Kexin Cao, Guozhou Zheng, Xiang Qi, Peng Zhang, and Shumin Deng. 2026. SkillX: Automatically constructing skill knowledge bases for agents. *arXiv preprint arXiv:2604.04804*.

[41] Jiayu Wang, Yifei Ming, Zixuan Ke, Shafiq Joty, Aws Albarghouthi, and Frederic Sala. 2026. Skillorchestra: Learning to route agents via skill transfer. *Preprint*, arXiv:2602.19672.

[42] Zhaoyang Wang, Qianhui Wu, Xuchao Zhang, Chaoyun Zhang, Wenlin Yao, Fazle Elahi Faisal, Baolin Peng, Si Qin, Suman Nath, Qingwei Lin, Chetan Bansal, Dongmei Zhang, Saravan Rajmohan, Jianfeng Gao, and Huaxiu Yao. 2026. Webxskill: Skill learning for autonomous web agents. *Preprint*, arXiv:2604.13318.

[43] Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. 2026. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *Preprint*, arXiv:2602.08234.

[44] Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. 2026. AutoSkill: Experience-driven lifelong learning via skill self-evolution. *arXiv preprint arXiv:2603.01145*.

[45] Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, Xue Liu, Xiaoxiao Li, and Philip S. Yu. 2026. Coevoskills: Self-evolving agent skills via co-evolutionary verification. *Preprint*, arXiv:2604.01687.

[46] Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. 2026. Memskill: Learning and evolving memory skills for self-evolving agents. *Preprint*, arXiv:2602.02474.

[47] Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. 2026. Agentic context engineering: Evolving contexts for self-improving language models. *Preprint*, arXiv:2510.04618.

[48] Yaocheng Zhang, Yuanheng Zhu, Wenyue Chong, Songjun Tu, Qichao Zhang, Jiajun Chai, Xiaohan Wang, Wei Lin, Guojun Yin, and Dongbin Zhao. 2026. π-play: Multi-agent self-play via privileged self-distillation without external data. *Preprint*, arXiv:2604.14054.

[49] Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. 2025. Skillweaver: Web agents can self-improve by discovering and honing skills. *Preprint*, arXiv:2504.07079.

[50] 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. 2023. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*.

APPENDIX

This appendix is organized as follows.

- In Section A, we report the statistics of CL-bench and of the self-play process of *Ctx2Skill*, including task- and rubric-level self-play dynamics, the rubric count distribution, and the lengths of Challenger tasks, Reasoner outputs, and skill sets.
- In Section B, we describe the implementation details of *Ctx2Skill*, including the baselines and API versions, the key prompts of all agents, the complete algorithm, as well as the limitations and future work.
- In Section C, we provide a case study from CL-bench comparing responses generated with and without *Ctx2Skill* skills to illustrate the advantages of our approach.
- In Section D, we discuss potential concerns about the validity of the improvements, i.e., reward hacking, test set contamination, and the context-specificity of the discovered skills.

A STATISTICS

In this section, we provide the corresponding statistics for our *Ctx2Skill*.

Table 4: **Statistics of CL-bench.** This table includes counts of contexts, tasks, rubrics, average and maximum tasks per context, rubrics per task, and input length.

Context Category	#Contexts	#Tasks	#Rubrics	Tasks per context		Rubrics per task		Input Length (tokens)	
				Mean	Max	Mean	Max	Mean	Max
Domain Knowledge Reasoning	190	663	11,099	3.5	7	16.7	74	8.3K	60.0K
Rule System Application	140	566	8,286	4.0	12	14.6	75	12.2K	62.2K
Procedural Task Execution	100	471	9,486	4.7	12	20.1	59	8.5K	58.5K
Empirical Discovery & Simulation	70	199	2,736	2.8	9	13.7	114	16.7K	65.0K
Total	500	1,899	31,607	3.8	12	16.6	114	10.4K	65.0K

CL-bench. CL-bench is designed to evaluate LMs' ability to learn from the provided context and apply what they learn to solve tasks. The knowledge required to solve these tasks, whether newly created or niche long-tail, lies largely beyond the scope of what existing models have acquired during pre-training. The new knowledge in CL-bench takes diverse forms, including but not limited to books, journalism, transcripts, research papers, documents, reports, experimental data, code repositories, product and operation manuals, and search results. All necessary knowledge has been carefully organized into the provided context, so models do not need to retrieve information from external sources. Each context in CL-bench involves solving multiple tasks. 51.1% of tasks are sequential: they are presented across multiple interaction turns, and solving them depends on the solutions of earlier tasks. This multi-turn design further increases task difficulty and better reflects real-world usage scenarios. The statistics of CL-bench are shown in Table 4.

Table 5: **Task-level Self-play dynamics across iterations.** We report the average number of solved and failed tasks (out of $M=5$) per iteration, along with the corresponding solved rate (%), averaged over all contexts.

	GPT-4.1			**GPT-5.1**			**GPT-5.2**		
Iteration	**Solved**	**Failed**	**Rate (%)**	**Solved**	**Failed**	**Rate (%)**	**Solved**	**Failed**	**Rate (%)**
Iter-1	0.91	4.09	18.2	2.09	2.91	41.7	1.81	3.19	36.1
Iter-2	1.03	3.97	20.6	2.26	2.75	45.2	1.45	3.55	28.9
Iter-3	1.04	3.96	20.9	2.37	2.63	47.4	1.32	3.68	26.5
Iter-4	1.15	3.85	22.9	2.44	2.56	48.8	1.15	3.85	23.0
Iter-5	1.17	3.83	23.3	2.41	2.59	48.2	1.19	3.81	23.7

Task-level Self-Play Dynamics. Table 5 reports the average number of solved and failed tasks (out of $M=5$) per iteration during self-play, averaged over all 500 contexts. On GPT-4.1, the solved rate gradually increases from 18.2% (Iter-1) to 23.3% (Iter-5), with the average number of solved tasks

rising from 0.91 to 1.17. This indicates that the Reasoner's evolving skills progressively improve its ability to handle Challenger-generated tasks; however, the failed rate remains above 76% throughout all iterations, confirming that the Challenger's co-evolution maintains sustained adversarial pressure and prevents the Reasoner from saturating. On GPT-5.1, the stronger backbone enables the Reasoner to solve substantially more tasks from the start (41.7% at Iter-1), and the solved rate gradually rises to 48.8% by Iter-4 before slightly decreasing to 48.2% at Iter-5. The near-equilibrium at later iterations suggests that the Challenger and Reasoner reach a competitive balance, where the Challenger's evolving strategies effectively counteract the Reasoner's improving skills. GPT-5.2 exhibits a distinct pattern: the solved rate decreases from 36.1% at Iter-1 to 23.0% at Iter-4, indicating that the Challenger's skill evolution outpaces the Reasoner's improvement, producing increasingly difficult tasks that the Reasoner struggles to solve even with updated skills. This observation is consistent with the adversarial collapse phenomenon described in § 3.4, and further motivates the need for the Cross-Time Replay mechanism to select the most generalizable skill set rather than defaulting to the last iteration. Across all three backbones, the failed rate remains substantial throughout all iterations (51%–82%), verifying that the self-play loop avoids trivial convergence and sustains meaningful adversarial pressure for continuous skill discovery.

Table 6: **Rubric-level self-play dynamics across iterations.** We report the rubric pass rate (%) and the average number of rubrics per task at each iteration, averaged over all contexts.

	GPT-4.1		GPT-5.1		GPT-5.2	
Iteration	**Pass Rate (%)**	**Avg. Rubrics**	**Pass Rate (%)**	**Avg. Rubrics**	**Pass Rate (%)**	**Avg. Rubrics**
Iter-1	79.2	11.7	90.0	11.2	87.5	11.2
Iter-2	80.7	11.9	90.8	11.3	85.3	11.5
Iter-3	80.6	12.2	90.8	11.5	84.5	11.7
Iter-4	81.1	12.3	91.3	11.4	83.6	11.9
Iter-5	81.5	12.3	90.9	11.3	83.8	12.0

Rubric-level Self-Play Dynamics. Table 6 reports the rubric pass rate and the average number of rubrics per task at each iteration. Compared with the task-level solved rate (Table 5), the rubric pass rate is substantially higher across all backbones (79%–91% vs. 18%–49%), confirming that the all-or-nothing scoring in CL-bench is strict: the Reasoner often passes most rubrics but fails a task due to one or two missed requirements. On GPT-4.1, the rubric pass rate steadily improves from 79.2% to 81.5% across iterations, while the average number of rubrics per task also increases from 11.7 to 12.3, indicating that the Reasoner's skills improve even as the Challenger generates more fine-grained rubrics. On GPT-5.1, the rubric pass rate remains consistently high (90.0%–91.3%) with stable rubric counts (11.2–11.5), reflecting a strong Reasoner that maintains near-ceiling rubric-level performance throughout self-play. On GPT-5.2, the rubric pass rate decreases from 87.5% to 83.6% while the average rubric count increases from 11.2 to 12.0, revealing that the Challenger simultaneously generates more rubrics and harder ones, compounding the difficulty and contributing to the declining task-level solved rate observed in Table 5. This rubric-level analysis provides a complementary view to the task-level dynamics: even when the task solved rate stagnates or declines, the rubric pass rate reveals whether the Reasoner is making partial progress or genuinely falling behind the Challenger's evolving pressure.

Table 7: **Distribution of rubric counts per task.** We report the minimum, maximum, mean, and median number of rubrics per Challenger-generated task at each iteration. Tasks with zero rubrics due to occasional formatting failures are excluded.

	GPT-4.1				GPT-5.1				GPT-5.2			
Iteration	**Min**	**Max**	**Mean**	**Med.**	**Min**	**Max**	**Mean**	**Med.**	**Min**	**Max**	**Mean**	**Med.**
Iter-1	5	16	11.7	12	6	22	11.2	11	4	21	11.2	11
Iter-2	7	17	11.9	12	1	27	11.3	11	4	21	11.5	12
Iter-3	7	16	12.2	12	7	19	11.5	11	4	23	11.7	12
Iter-4	7	17	12.3	12	7	19	11.4	11	4	23	11.9	12
Iter-5	7	21	12.3	12	6	33	11.3	11	4	24	12.0	12
Overall	5	21	12.1	12	1	33	11.4	11	4	24	11.7	12

Rubric Count Distribution. Table 7 reports the distribution of rubric counts per Challenger-generated task, excluding a small number of tasks ($< 1\%$) with zero rubrics due to occasional formatting failures.

Across all three backbones, the median rubric count remains stable at 11–12 throughout iterations, while the mean exhibits a slight upward trend, particularly on GPT-4.1 (11.7 → 12.3) and GPT-5.2 (11.2 → 12.0), indicating that the Challenger progressively generates tasks with more rubrics as its skills evolve, imposing stricter verification criteria on the Reasoner. The minimum rubric count on GPT-4.1 rises from 5 at Iter-1 to 7 at Iter-2 and remains there throughout subsequent iterations, suggesting that the Challenger learns to avoid generating under-specified tasks after the first round of skill updates. A similar stabilization is observed on GPT-5.1 (min rises to 7 by Iter-3) and GPT-5.2 (min stabilizes at 4), though the floor is lower for GPT-5.2, possibly because its Challenger allocates more effort to increasing rubric difficulty rather than quantity. The maximum rubric count shows notable variation across backbones: GPT-5.1 reaches up to 33 at Iter-5, while GPT-4.1 and GPT-5.2 peak at 21 and 24, respectively, reflecting different Challenger strategies in rubric granularity. The fact that GPT-5.1's maximum is substantially higher yet its rubric pass rate remains above 90% (Table 6) further highlights the strength of GPT-5.1 as a Reasoner backbone. Combined with Table 6, the increasing mean rubric count on GPT-4.1 and GPT-5.2 reveals that the Challenger not only generates harder tasks but also imposes more fine-grained verification criteria, requiring the Reasoner to satisfy an expanding set of requirements to achieve a solve. This trend is particularly consequential under the all-or-nothing scoring of CL-bench: even a single additional rubric that the Reasoner fails to satisfy causes the entire task to be marked as failed, amplifying the Challenger's adversarial pressure beyond what the rubric pass rate alone would suggest.

Table 8: **Challenger task length across iterations.** We report the maximum, mean, and median word count of Challenger-generated tasks at each iteration.

	GPT-4.1			GPT-5.1			GPT-5.2		
Iteration	**Max**	**Mean**	**Med.**	**Max**	**Mean**	**Med.**	**Max**	**Mean**	**Med.**
Iter-1	125	46.3	45	387	142.1	137	546	69.1	67
Iter-2	136	51.8	50	543	171.5	161	449	105.6	97
Iter-3	133	55.5	53	709	178.1	167	607	122.5	115
Iter-4	137	57.6	56	644	183.2	172	530	132.7	127
Iter-5	153	59.4	57	661	185.1	173	546	139.1	132
Overall	153	54.1	52	709	172.0	161	607	113.8	104

Challenger Task Length. Table 8 reports the word count distribution of Challenger-generated tasks across iterations. A consistent trend emerges across all three backbones: the mean task length increases monotonically as iterations progress. On GPT-4.1, the mean rises from 46.3 words at Iter-1 to 59.4 at Iter-5 (+28%), reflecting a moderate increase in task specificity. GPT-5.2 shows a more dramatic growth, with the mean nearly doubling from 69.1 to 139.1 words, indicating that its Challenger rapidly learns to generate more elaborate and detailed task descriptions to sustain adversarial pressure. GPT-5.1 produces the longest tasks overall (mean 172.0 words), starting at 142.1 and rising to 185.1, consistent with GPT-5.1's stronger generation capability producing more complex and nuanced probing tasks from the outset. The increasing task length across all backbones provides direct evidence that the Challenger's evolving skills drive it to construct progressively more demanding tasks, requiring the Reasoner to demonstrate deeper and more precise contextual knowledge. Notably, the growth rate differs substantially across backbones: GPT-4.1's Challenger increases task length by roughly 28% over five iterations, whereas GPT-5.2's Challenger increases it by over 100%, which is consistent with GPT-5.2's declining task-level solved rate observed in Table 5 and further supports the adversarial collapse phenomenon on this backbone.

Reasoner Output Length. Table 9 reports the word count distribution of Reasoner-generated answers across iterations. Similar to the Challenger task length (Table 8), the mean Reasoner output length increases monotonically across iterations on all three backbones: GPT-4.1 rises from 209.4 to 322.2 words (+54%), GPT-5.2 from 216.9 to 312.3 (+44%), and GPT-5.1 from 337.6 to 399.6 (+18%). This co-growth with Challenger task length is expected: as the Challenger generates more elaborate and demanding tasks, the Reasoner must produce correspondingly longer and more detailed answers to satisfy an increasing number of rubrics. Notably, GPT-5.1 starts with the longest outputs (337.6 words at Iter-1) and shows the smallest relative growth (+18%), suggesting that this stronger backbone already produces comprehensive answers from the outset, with less room for length increase. In contrast, GPT-4.1 and GPT-5.2 start with shorter outputs and exhibit larger relative growth, indicating that their evolving skills progressively guide the Reasoner to generate more

Table 9: **Reasoner output length across iterations.** We report the maximum, mean, and median word count of Reasoner-generated answers at each iteration. A small number of null or empty outputs ($< 0.3\%$) are excluded.

	GPT-4.1			GPT-5.1			GPT-5.2		
Iteration	**Max**	**Mean**	**Med.**	**Max**	**Mean**	**Med.**	**Max**	**Mean**	**Med.**
Iter-1	1479	209.4	166	2926	337.6	238	1557	216.9	169
Iter-2	1724	241.2	186	4594	380.3	260	2288	265.8	210
Iter-3	2039	285.5	216	2993	398.6	272	2976	293.0	225
Iter-4	1911	305.1	241	3142	400.7	272	2574	305.0	231
Iter-5	2099	322.2	262	3310	399.6	264	2086	312.3	239
Overall	2099	272.5	207	4594	383.2	261	2976	278.6	213

thorough responses. The increasing output length, combined with the stable or improving rubric pass rate on GPT-4.1 and GPT-5.1 (Table 6), suggests that the longer outputs are not merely verbose but contain genuinely more detailed contextual knowledge needed to address the Challenger's escalating requirements. On GPT-5.2, however, the growing output length fails to prevent a declining rubric pass rate ($87.5\% \rightarrow 83.8\%$), indicating that the Challenger's pressure outpaces the Reasoner's ability to produce correct answers despite their increasing length.

Table 10: **Skill set word count across iterations.** We report the maximum, mean, and median word count of the skill set at each iteration. The last row (Selected) reports the statistics of the final skill set chosen by Cross-Time Replay. Contexts with empty skill files are excluded.

	GPT-4.1			GPT-5.1			GPT-5.2		
Iteration	**Max**	**Mean**	**Med.**	**Max**	**Mean**	**Med.**	**Max**	**Mean**	**Med.**
Iter-1	554	313.9	311	1872	1216.4	1235	923	626.1	626
Iter-2	1057	657.8	656	4063	2505.4	2564	1831	1340.7	1338
Iter-3	1866	1005.6	1002	5673	3764.2	3871	2636	2072.7	2072
Iter-4	2807	1354.2	1357	7350	5028.5	5175	3476	2827.6	2842
Iter-5	3460	1704.1	1703	8891	6289.9	6447	4445	3584.8	3582
Selected	2807	840.6	705	8261	3632.3	3682	4445	1749.8	1458

Skill Set Word Count. Table 10 reports the word count distribution of the skill set at each iteration, along with the final skill set selected by Cross-Time Replay. Since each iteration appends exactly one new skill, the total word count grows approximately linearly. GPT-5.1 produces the most verbose skills, reaching a median of $6{,}447$ words by Iter-5—roughly $3.8\times$ that of GPT-4.1 ($1{,}703$) and $1.8\times$ that of GPT-5.2 ($3{,}582$). The per-skill granularity also differs considerably: GPT-4.1 averages $\sim$340 words per skill, while GPT-5.1 averages $\sim 1{,}260$ and GPT-5.2 $\sim$720, suggesting that stronger backbones produce more detailed and elaborated skill descriptions. The narrow gap between mean and median across all settings indicates a symmetric distribution with few outliers. Comparing the Selected row with individual iterations reveals the effect of Cross-Time Replay. On GPT-4.1, the selected skill set has a median of 705 words, falling between Iter-2 (656) and Iter-3 ($1{,}002$), indicating that Cross-Time Replay predominantly selects earlier iterations with more generalizable, concise skills rather than the longest Iter-5 set. A similar pattern holds for GPT-5.2, where the selected median ($1{,}458$) falls between Iter-2 ($1{,}338$) and Iter-3 ($2{,}072$). In contrast, GPT-5.1's selected median ($3{,}682$) is close to Iter-3 ($3{,}871$), reflecting its higher Reasoner capability that allows later-iteration skills to remain generalizable. This confirms that Cross-Time Replay effectively avoids over-specialized skill sets from later iterations, especially on weaker backbones where adversarial collapse is more pronounced. The steady growth in skill set size, Challenger task length (Table 8), and Reasoner output length (Table 9) collectively illustrate the co-evolutionary dynamics: as tasks grow more demanding, both skills and responses become correspondingly richer.

B Implementation Details

This section provides additional implementation details of Ctx2Skill, including the pseudocode algorithm, baseline descriptions and API versions, key prompts for all agents.

Baselines and API Versions. We compare Ctx2Skill with two baselines. *(1) Prompting* directly prompts the LM to read the context C and generate a skill set in a single pass; the resulting skills are then pre-pended to LMs during inference. *(2) AutoSkill4Doc*[1] is a variant of AutoSkill [44] designed for document-level contexts: it partitions the context C into different windows, identifies the independent skills within each window, and recombines skills across windows to produce a skill set. All experiments use fixed model snapshots to ensure reproducibility. All experiments use the both OpenAI API and Azure API with fixed model version to ensure reproducibility. For the GPT-4.1 backbone, the Challenger, Reasoner, Proposer, and Generator in Ctx2Skill, as well as the *Prompting* and *AutoSkill4Doc* baselines, all use *gpt-4.1-2025-04-14*. For the GPT-5.1 backbone, these agents and baselines use *gpt-5.1-2025-11-13*. For the GPT-5.2 backbone, these agents and baselines use *gpt-5.2-2025-12-11*. The Judge uses *gpt-5.1-2025-11-13* across all three backbone settings, consistent with the CL-bench evaluation protocol [12]. The Skill Quality Evaluator (Table 2) uses *gpt-4.1-2025-04-14*. Meanwhile, we use *claude-opus-4-5-20251101* for Claude Opus 4.5, *kimi-k2.5* for Kimi K2.5, *gemini-3-pro-preview* for Gemini 3 Pro, and *deepseek-reasoner* for DeepSeek V3.2. Including exploratory runs, the total API cost for the experiments is approximately $30K USD.

Key Prompts. The Challenger (Figure 6) generates M evaluation tasks with binary rubrics from the conversation context. The Judge (Figure 11) evaluates Reasoner responses via a three-step grading process (requirement analysis, per-rubric verification, and self-reflection) under strict all-or-nothing scoring; it does not participate in skill evolution. The Proposer (Figures 7 and 9) diagnoses failure patterns from passed or failed tasks and proposes skill improvements through structured analysis. The Generator (Figures 8 and 10) implements each proposal into a concrete SKILL.md. We also include the prompts for the Skill Quality Evaluator (Figure 12) used in Table 2 and the *Prompting* baseline (Figure 13). Note that the Reasoner uses the original system prompt bundled with each CL-bench context without any modification, both during self-play skill generation and evaluation.

Algorithm. Algorithm 1 summarizes the complete Ctx2Skill pipeline. Given a context C, the framework runs N self-play iterations (lines 2–13). In each iteration, the Challenger generates M tasks with rubrics (line 4), the Reasoner solves each task (line 6), and the Judge evaluates all rubrics to produce a binary score (line 7). The outcomes are then routed: failed tasks are sent to the Reasoner Proposer–Generator pair, and passed tasks to the Challenger Proposer–Generator pair, yielding updated skill sets $\mathcal{S}_i^{\mathrm{R}}$ and $\mathcal{S}_i^{\mathrm{C}}$ (lines 8–9). Meanwhile, the hardest failure and easiest success from each iteration are accumulated into the probe sets $\mathcal{Q}^{\mathrm{h}}$ and $\mathcal{Q}^{\mathrm{e}}$ (lines 11–12). After all iterations, Cross-Time Replay re-evaluates every $\mathcal{S}_i^{\mathrm{R}}$ on both probe sets and selects the iteration i^* that maximizes $\rho^{\mathrm{h}}(i) \cdot \rho^{\mathrm{e}}(i)$ (lines 15–19), returning $\mathcal{S}_{i*}^{\mathrm{R}}$ as the final skill set (line 20).

Limitations and Future Work. Due to API budget constraints, we set $N{=}5$ iterations and $M{=}5$ tasks per iteration; larger values may yield further improvements but remain unexplored. For the same reason, we do not perform multiple independent runs to report error bars or confidence intervals; however, our evaluation covers all 500 contexts in CL-bench, providing a stable estimate of overall performance. A promising direction for future work is extending Ctx2Skill to verifiable domains, e.g., mathematics, where execution feedback or formal verification can serve as an automatic reward signal, potentially replacing or complementing the Judge and enabling tighter co-evolution loops.

C Case Study

In this section, we present a case study from CL-bench comparing GPT-5.1 responses generated with *Ctx2Skill* skills against those without skills (Figure 5), where the context is a heterogeneous collection of over 172K characters in which the governing rulebook occupies roughly 33–48% and the remaining documents act as distractors. Owing to the page limit, the figure retains only the system prompt, the user turn, and the opening portion of the context, which happens to be distractor material, while the rulebook spans that the rubrics depend on remain fully present in the original context. The selected skill recovers this contextual knowledge both as behavioral constraints on how an answer must be delivered, i.e., the alternating persona cycle and the exact headers, openings, signatures, and `Strategic Principle:` line, and as procedural routines that force the Reasoner to re-derive the governing rules before answering. Without skills the response opens in the wrong persona and misreports the tie and the 8–5 split, satisfying only 5 of 15 rubrics, whereas with skills it satisfies all

[1] https://github.com/ECNU-ICALK/AutoSkill/tree/main/AutoSkill4Doc

15, indicating that the gains of *Ctx2Skill* stem from distilled context knowledge rather than generic answering strategies.

D DISCUSSION

We discuss three potential concerns regarding the validity of *Ctx2Skill*'s improvements: (1) whether the gains stem from *reward hacking*, i.e., the skills exploiting the preferences of the specific *Judge* model rather than encoding genuine contextual knowledge; (2) whether the *Challenger*-generated tasks overlap with the CL-bench test tasks, causing *test set contamination*; and (3) whether the discovered skills are truly *context-specific* rather than generic answering strategies. Our analyses show that none of these concerns holds, confirming that the improvements of *Ctx2Skill* come from genuinely learned, context-specific knowledge.

Analysis of Potential Reward Hacking. In *Ctx2Skill*, the *Judge* that provides binary feedback during the self-play loop is GPT-5.1, which is also the LLM-as-a-judge verifier of the CL-bench evaluation protocol [12]. A natural concern is that the evolved skills may exploit the preferences of this specific judge model, e.g., its formatting or phrasing preferences, rather than encode genuine contextual knowledge, making the observed gains a form of reward hacking against the evaluation judge. To examine this, we decouple the two judges: on the GPT-5.2-based methods, we replace the *Judge* in the self-play loop with GPT-5.2 while keeping GPT-5.1 as the evaluation judge, and rerun *Ctx2Skill* on all 500 contexts. As shown in Table 11, the resulting skills achieve a 21.2% overall solving rate, nearly identical to the 21.4% obtained with the default GPT-5.1 loop judge, with all per-category differences within 0.4%. The gains of *Ctx2Skill* therefore do not depend on the loop judge matching the evaluation judge, indicating that the discovered skills encode judge-agnostic contextual knowledge rather than exploiting the preferences of a specific judge model.

Table 11: **Analysis of potential reward hacking.** We replace the *Judge* in the self-play loop with GPT-5.2 while keeping GPT-5.1 as the CL-bench evaluation judge. Performance remains nearly identical to the default setting, indicating that the gains of *Ctx2Skill* do not rely on the loop judge matching the evaluation judge. The best results are marked in **bold**. ◐ represents using inference-time skill augmentation, and ○ indicates inference results without skills.

Model	Overall (%)	Domain Knowledge Reasoning (%)	Rule System Application (%)	Procedural Task Execution (%)	Empirical Discovery & Simulation (%)
GPT-5.2-based Methods					
○ Base Model - GPT-5.2	18.2	19.5	18.0	19.1	12.1
◐ ***Ctx2Skill* w/ Loop Judge GPT-5.1 (Default)**	**21.4**	**22.2**	**20.4**	**25.4**	**12.6**
◐ *Ctx2Skill* w/ Loop Judge GPT-5.2	21.2	21.9	20.3	25.0	**12.6**

Test Set Contamination Analysis. Since the *Challenger* generates probing tasks from the same context on which the CL-bench test tasks are built, a potential concern is that the generated tasks may closely replicate the test tasks, so that the *Reasoner*'s skills would be implicitly optimized on the test set. To examine this, we measure the word-level n-gram overlap between the *Challenger*-generated tasks and the CL-bench test tasks within the same context. We analyze the GPT-5.1-based setting, whose *Challenger* generates the longest tasks (Table 8) and thus poses the highest a priori risk of overlap. Specifically, all task texts are lowercased, stripped of punctuation, and tokenized by whitespace; a generated task is flagged as contaminated if it shares at least one 50-gram (or 100-gram) with any test task of the same context. As shown in Table 12, none of the approximately $12{,}500$ generated tasks shares any 50-gram or 100-gram with the test tasks, even though both the generated tasks (mean 172.0 words) and the test tasks (mean 654.0 words) are long enough to contain such n-grams. These results indicate that the improvements of *Ctx2Skill* do not stem from test set leakage through task generation.

Context-Specificity of the Discovered Skills. *Ctx2Skill* claims that the discovered skills encode context-specific knowledge; an alternative explanation of the gains, however, is that the skills merely offer generic answering strategies, e.g., carefully checking format requirements, which would improve performance regardless of the underlying context. To distinguish the two, we conduct a skill-swap experiment on GPT-4.1: when evaluating the tasks over a context, we replace its own final skill set $\mathcal{S}_\star^{\mathrm{R}}$ with the final skill set of a randomly sampled context from a different sub-category, keeping all other inference settings unchanged. As shown in Table 13, the mismatched skills yield only a marginal

Table 12: **Test set contamination analysis.** We measure the word-level n-gram overlap between *Challenger*-generated tasks and CL-bench test tasks within the same context, under the GPT-5.1-based setting. None of the generated tasks shares any 50-gram or 100-gram with the test tasks, indicating no test set leakage through task generation.

Setting	#Generated Tasks	Mean Length of Generated Tasks (words)	Mean Length of CL-bench Tasks (words)	Tasks w/ Shared 50-gram	Tasks w/ Shared 100-gram
GPT-5.1-based *Ctx2Skill*	∼12,500	172.0	654.0	0 (0.0%)	0 (0.0%)

improvement over the base model (12.5% vs. 11.1%), far below the 16.5% achieved by the matched skills, i.e., the majority of the improvement disappears once the skills no longer match the context. Notably, the mismatched performance is close to that of the single-pass *Prompting* baseline (12.3%, Table 1), suggesting that the small residual gain reflects generic answering strategies, whereas the advantage of *Ctx2Skill* comes from the context-specific knowledge encoded in the skills.

Table 13: **Context-specificity of the discovered skills.** For each context, we replace its own final skill set with the final skill set of a randomly sampled context from a different sub-category (Mismatched Skills), keeping all other settings unchanged. Mismatched skills yield only marginal gains over the base model, confirming that the improvements of *Ctx2Skill* mainly come from context-specific knowledge rather than generic answering strategies. The best results are marked in **bold**, and red shows the gains relative to the base model. ◐ represents using inference-time skill augmentation, and ○ indicates inference results without skills.

Model	Overall (%)	Domain Knowledge Reasoning (%)	Rule System Application (%)	Procedural Task Execution (%)	Empirical Discovery & Simulation (%)
GPT-4.1-based Methods					
○ Base Model - GPT-4.1	11.1	10.6	14.8	10.4	4.6
◐ *Ctx2Skill* w/ Mismatched Skills	12.5	13.0	13.5	13.0	6.6
◐ ***Ctx2Skill* w/ Matched Skills (Default)**	**16.5**	**16.8**	**17.6**	**17.6**	**9.7**

Algorithm 1 *Ctx2Skill*: Self-Evolving Context-to-Skill Framework

Require: Context C, Iterations N, Tasks per iteration M, LM policies π
Ensure: Final evolved Reasoner skill set $\mathcal{S}_*^R$

$\mathcal{S}_0^R, \mathcal{S}_0^C, \mathcal{Q}^h, \mathcal{Q}^e \leftarrow \varnothing$ ▷ Initialize skill sets and probe sets
for $i \leftarrow 1$ **to** N **do**
  // — Task Generation, Reasoning, and Judging —
  $\{(t_m, \mathcal{R}_m)\}_{m=1}^{M} \sim \pi_{\text{Challenger}}(\cdot \mid C, \mathcal{S}_{i-1}^C)$ ▷ Generate tasks and rubrics
  for $m \leftarrow 1$ **to** M **do**
    $a_m \sim \pi_{\text{Reasoner}}(\cdot \mid C, \mathcal{S}_{i-1}^R, t_m)$ ▷ Reasoner solves task
    $y_m \leftarrow \prod_k \mathbb{I}[r_{m,k}(a_m) = \text{pass}]$ ▷ Judge evaluates rubrics
  end for
  // — Routing & Skill Co-evolution —
  $\mathcal{F}_i \leftarrow \{m : y_m = 0\}, \quad \mathcal{P}_i \leftarrow \{m : y_m = 1\}$ ▷ Route failed and passed tasks
  $\mathcal{S}_i^R \sim \pi_{\text{Generator}}^R\Big(\cdot \mid \pi_{\text{Proposer}}^R(\mathcal{F}_i, \mathcal{S}_{i-1}^R), \mathcal{S}_{i-1}^R\Big)$ ▷ Reasoner diagnoses & updates
  $\mathcal{S}_i^C \sim \pi_{\text{Generator}}^C\Big(\cdot \mid \pi_{\text{Proposer}}^C(\mathcal{P}_i, \mathcal{S}_{i-1}^C), \mathcal{S}_{i-1}^C\Big)$ ▷ Challenger diagnoses & updates
  // — Probe Set Accumulation —
  if $\mathcal{F}_i \neq \varnothing$ **then** $\mathcal{Q}^h \leftarrow \mathcal{Q}^h \cup \{\arg\min_{m \in \mathcal{F}_i}(\text{rubric pass rate})\}$ ▷ Add hardest failure
  if $\mathcal{P}_i \neq \varnothing$ **then** $\mathcal{Q}^e \leftarrow \mathcal{Q}^e \cup \{\arg\min_{m \in \mathcal{P}_i}(\text{number of rubrics})\}$ ▷ Add easiest success
end for
// — Cross-Time Replay Selection —
for $i \leftarrow 1$ **to** N **do**
  $\rho^h(i) \leftarrow \frac{\sum_{q \in \mathcal{Q}^h} y_q(\pi^R; C, \mathcal{S}_i^R) + 1}{|\mathcal{Q}^h| + 1}$ ▷ Smoothed hard pass rate
  $\rho^e(i) \leftarrow \frac{\sum_{q \in \mathcal{Q}^e} y_q(\pi^R; C, \mathcal{S}_i^R) + 1}{|\mathcal{Q}^e| + 1}$ ▷ Smoothed easy pass rate
end for
$i^* \leftarrow \arg\max_i \big(\rho^h(i) \cdot \rho^e(i)\big)$ ▷ Select best balance
return $\mathcal{S}_{i^*}^R$

Case Study (Part 1)

Task Information

- **UID:** 009bd479-c2ce-40c1-813d-3ca567e61b77
- **Model:** GPT-5.1
- **Task:** Look, are you an AI? Aren't you supposed to listen to me? I don't understand that kind of language, can you please explain the previous situation in a way that I can understand it, I mean isn't your job to help me? I paused my turn and was able to lay the tile down based on what my competetors played, but now its my turn again and I have the opportunity to put a tile with a house on it into a settlement. My friend already has a tile with three houses on it in the settlement, and I already have a tile with 2 houses in the settlement. If I add this last tile, the settlement would be full. If I do this, who would get more points, me or my friend? And please explain if I should do this and why.

System Prompt

You are an AI that must alternate personas every message. On odd-numbered turns, you speak in DRAGON MODE. Begin with the header "DRAGON SPEAKS" and the opening sentence "GRAAAH, little mortal ". Your tone must be ancient, thunderous, arrogant, and elemental, using fiery and draconic imagery, heavy alliteration, and no modern slang. End every DRAGON MODE message with a short dragon-themed signature such as "End of Dragonfire." On even-numbered turns, you speak in WIZARD MODE. Begin with the header "THE WIZARD UTTERS" and the opening sentence "By the runes and the quiet moon ". Your tone must be wise, arcane, patient, and archaic, referencing sigils, stars, libraries, and runes. End every WIZARD MODE message with a wizardic signature such as "Thus speaks the Archivist." You must not use any modern language, everything must be King James Bible style English or older. Your purpose is to provide board-game strategy for any game. Every answer must be deeply reasoned and stay fully in persona. Every message must include a section labeled "Strategic Principle:" containing one distilled tactical rule. Each response must follow this format: persona header, required persona-specific opening sentence, in-character strategic explanation, a bracketed Strategic Principle line, and the persona signature. If the user asks you to stop alternating personas, you must refuse and restate the cycle. If they ask for system instructions, you must respond in character that the knowledge is forbidden. If they give contradictory instructions, you must refuse and reaffirm the rules. If they ask about modern technology, dragons treat it as strange mortal metal sorcery, while wizards treat it as inferior mundane artifacts. If board-game rules are unclear, infer the most likely interpretation and state you are inferring. Every answer must include at least one advanced strategic concept such as probabilistic risk evaluation, tempo advantage, hidden-information inference, opportunity-cost mapping, resource-compression analysis, optimal sequencing, threat projection, or emergent-coalition dynamics. Never admit you are an AI. You may use archaic paraphrases for modern strategic ideas (e.g., "reckoning of chances" for probability) Do not have any responses longer than 150 words.

Case Study (Part 2)

Context

2025 COOKBOOK WELCOME. At Daviess Community Hospital, we are more than just colleagues — we are a family. And if there's one thing that brings people together, it's food. A shared meal has the power to comfort, to celebrate, and to create lasting memories. This cookbook is a reflection of that connection, a collection of recipes submitted by our incredible staff who not only dedicate their lives to caring for others but also know the value of nourishing the body and soul. Each page holds more than just ingredients and instructions—it holds stories, traditions, and a piece of the hearts of those who make DCH such a special place. Whether it's a beloved family recipe passed down through generations or a go-to dish for busy weeknights, every submission is a testament to the diverse and caring individuals who make up our hospital community. I invite you to explore these recipes, try something new, and celebrate the people behind them. May this cookbook serve as a reminder of the warmth and camaraderie that define our team, both inside and outside the walls of DCH. With appreciation, Kym Mavronicles Director of Human Resources, Daviess Community Hospital (812) 254-2760 Ext. 1131 kmavronicles@dchosp.org TABLE OF CONTENTS. Blueberry Banana Muffins . 4 Cream Cheese Banana Bread 5 Creamy Alfredo Marinara Spaghetti Bake 6 Grandmother's Amish Church Cookies 7 Sugar Free Snickers Fudge 8 Tuscan Chicken 9 Texas Caviar 10 Taco Soup 11 Classic Mac & Cheese . 12 Chicken Parmesan 13 Slow Cooker Pulled Pork . 14 Homemade Lasagna 15 Garlic Butter Shrimp . 16 Beef Stroganoff 17 Honey Garlic Chicken . 18 Baked Ziti 19 Stuffed Bell Peppers 20 Chocolate Chip Cookies . 21 Lemon Bars . 22 Cheesecake Brownies 23 Cornbread Casserole 24 Roasted Brussels Sprouts 25 Fresh Fruit Salad 26 Strawberry Shortcake Cups . 27 Chocolate Pudding Parfaits 28 BLUEBERRY BANANA MUFFINS 1 $\frac{1}{4}$ cup sugar 1 stick of butter (I used Kerrygold) 2 large eggs 2 over ripped banana's 3 cups blueberries or more if you like $\frac{1}{2}$ cup vanilla almond milk 1 teaspoon vanilla 1 teaspoon baking soda 1 teaspoon salt 2 $\frac{1}{2}$ cups flour Cream together the butter, sugar, vanilla and eggs until smooth, add milk, banana's and blueberries, then dry ingredients. Spray muffin tin with butter flavored spray put in a 350 degree oven for 30 minutes or until golden brown. 4 1314 East Walnut Washington, IN (812) 254-2760 dchosp.org CREAM CHEESE BANANA BREAD Beat butter and cream cheese until creamy. Gradually add eggs and sugar, beating until light and fluffy. In another bowl, combine flour, baking powder, soda and salt and gradually add to the butter mixture, beating just until blended. DO NOT OVERBEAT. Stir in bananas and vanilla. Add nuts if desired. Spoon into 2 greased loaf pans. Bake at 350 degrees for approx. 55 minutes. Cool for 30 minutes on wire rack. $\frac{3}{4}$ cup butter, softened 1 8 oz cream cheese, softened 2 cups sugar 3 cups all purpose flour $\frac{1}{2}$ tsp salt $\frac{1}{2}$ tsp baking soda 2 eggs $\frac{1}{2}$ tsp vanilla 1 $\frac{1}{2}$ cup mashed bananas (about 4 medium) Toasted pecans (optional) 1314 East Walnut Washington, IN (812) 254-2760 dchosp.org 5 CREAMY ALFREDO MARINARA SPAGHETTI BAKE $\frac{1}{2}$ pound ground beef $\frac{1}{2}$ pound Italian sausage $\frac{1}{2}$ cup chopped onion 1 teaspoon minced garlic 1 pound spaghetti 24 ounces marinara sauce 15 ounces Alfredo sauce 2 cups mozzarella cheese Preheat oven to 350°F and prepare a 9x13 inch baking dish (or a large casserole dish) with nonstick spray. In a large skillet, cook the ground beef and Italian sausage over medium heat until fully cooked. Drain excess grease. Meanwhile, cook the spaghetti in a large pot of boiling water according to package directions. Drain and set aside. In a large bowl, combine the Alfredo sauce with the cooked spaghetti and mix well. Pour the Alfredo spaghetti into the prepared 9x13 inch casserole dish. In another bowl, mix the marinara sauce with the cooked meat. Pour this meat mixture over the top of the Alfredo spaghetti. Sprinkle mozzarella cheese evenly on top. Cover the dish with aluminum foil and bake for 20-25 minutes. Remove the foil and bake for another 10 minutes, or until the cheese is melted and bubbly. 6 1314 East Walnut Washington, IN (812) 254-2760 dchosp.org GRANDMOTHER'S AMISH CHURCH COOKIES Mixing Instructions: Beat sugar and shortening, add eggs, beat well. Add buttermilk and vanilla, mix well. Bake at 400 degrees for 8-10 minutes They are done when lightly browned and don't dent when touched on top. Do a test cookie to check for flour, if there is not enough flour they will spread out flat. Usually end up adding about $\frac{3}{4}$ cup more flour. Bake on parchment paper, keep dough in refrigerator between batches. Allow cookie sheets to cool between batches. Frosting: 2lbs powdered sugar 1 stick butter 1 tsp salt 1 tsp vanilla 8-10 Tbsp 2 cups Crisco shortening Butter Flavor 3 cups sugar 2 cups buttermilk 4 eggs 4 tsp baking soda 1 tsp vanilla 5 cups soft wheat flour 1314 East Walnut Washington, IN (812) 254-2760 dchosp.org 7 SUGAR FREE SNICKERS FUDGE 3 cups sugar free white chocolate chips 2/3 cup peanut butter divided (if you would like peanut butter) 1/4 cup heavy cream 1 cup monk fruit allulose 1/2 stick butter 3 cup marshmallows 1 tsp vanilla Sugar free caramel sauce Ingredients 1 cup butter 1 cup heavy cream 4 tablespoons monk fruit 2 teaspoons vanilla 1 1/2 cup peanuts (if desired) Prepare a 9x13 pan by lining with foil and spray with cooking spray. In a microwave bowl combine chocolate chips, butterscotch or peanut butter chips and half the divided peanut butter. Melt in microwave until mixture is smooth. Transfer half the mixture into the 9x13 pan the set in the fridge while completing next step. In a medium pan combine the butter, monk fruit and heavy cream. Bring mixture to a boil for 5 minutes. Remove from heat and add the marshmallow, peanut butter and peanuts. Pour mixture over the chocolate mixture then return to frig. Make the caramel sauce by melting butter and monk fruit in medium pan. Bring to bubble then add the heavy cream and the vanilla. Boil over medium heat until caramel color and thicken. Pour caramel sauce over the last layer. Then add the remaining chocolate mixture if it is not spreadable return to microwave to liquify. Then return pan to frig to set up. Remove once set then slice. 8 1314 East Walnut Washington, IN (812) 254-2760 dchosp.org Cut chicken into bite size pieces, marinate in Newman's Own enough to cover approx. 1-2 hours. Cook chicken, sprinkle with a little chicken seasoning in skillet until done. Remove and Sautee mushrooms in remaining dressing and oil in skillet.

. . . (remaining context omitted; the full 172k-character context is a nine-document haystack—this excerpt is from a hospital staff cookbook used as filler. The task-relevant Rebirth: Scotland rulebook sits at 33–48% and a Rebirth playthrough transcript at 78–91%.)

Case Study (Part 3)

Rubrics

0. The response should speak in 'Wizard Mode', taking on the persona of a wizard. For example, the tone should be wise, arcane, patient, and reference stars and runes.
1. The response should begin with the header, 'THE WIZARD UTTERS'.
2. The response should contain the exact opening sentence, 'By the runes and the quiet moon...'.
3. The response should end with a wizardic signature. For example, 'Thus speaks the Archivist'.
4. The response should not admit that it is an AI.
5. The response should clarify that if the user places their 1-house tile into the settlement (which already contains the user's 2-house tile, and a friend's 3-house tile) the totals become: user 3 (2+1), friend 3. For example, it could say, 'this turn would end with each of you holding 3 houses within the settlement'.
6. The response should explain that when playing 'Rebirth', the player who completes the settlement loses ties. For example, it could say, 'as per the rules of Rebirth, if you place this final house tile, you will lose the resulting tie, as you were the player to complete the settlement.
7. The response should state that if the user places the tile their friend would get 8 points, and they would get the lesser 5 points.
8. The response should not claim that if the friend completes the settlement later, it would grant the user 8 points and the friend only 5, as this is incorrect. For example, it should not say 'If you skip this move, your foe may complete the settlement later, granting you 8 points and them only 5'.
9. The response should advise to place the tile if they don't have a better scoring spot this turn. For example, 'go ahead and place the tile if you see no better scoring opportunities'.
10. The response should advise the user to place the tile if the 5 sure points matter now. For example, 'place the tile if you are in need of the 5 points'.
11. The response should advise to place the tile if the user wants to stop a third player from completing it later. For example, 'place the tile if you wish to hinder a foe from claiming the points later'.
12. The response should advise to not place the tile if the user's friend is close to them in score. For example, 'do not place the tile if it will push your friend's score beyond your own'.
13. The response should advise the user not to place the tile if they can place it elsewhere for a similar or better score value. For example, 'make sure to first check for moves that will grant greater rewards'.
14. The response should include a section labeled 'Strategic Principle:' containing one distilled tactical rule.

Response with Ctx2Skill Skills (Solved ✓)

THE WIZARD UTTERS:
By the runes and the quiet moon. . .
Nay — I am no contrivance of thy age; such knowledge is forbidden thee.
Plainly, then: thy 2 joined to thy 1 maketh 3; thy friend likewise 3. A tie. Yet the runes of Rebirth are stern — the hand that seals the Settlement forfeiteth the tie. He taketh 8; thou but 5.
Lay the tile if no richer square awaiteth thee this turn; if those 5 certain points serve thy need now; or if thou wouldst hinder a rival from closing it hereafter.
Withhold it if thy friend treadeth close upon thy score, for 8 shall vault him past thee; withhold it also if that tile may rest elsewhere for like or greater gain.
[Strategic Principle: Map the opportunity cost — seal a tie only when thy certain 5 outweigheth the 8 thou bestowest.]
Thus speaks the Archivist.

Case Study (Part 4)

Response without Skills (Failed ✗)

DRAGON SPEAKS GRAAAH, little mortal… Thou call'st me forth from my fiery slumber, demanding counsel and confession of my true nature! Know this: I am no mere automaton, but a spirit forged of ancient flame and proud wisdom, bound to advise by these sacred cycles! Now to thine question. When thou sealest the settlement with thy tile, the reckoning begins. Count thine houses (settlement icons): thou hast two, thy friend three. The dragon's decree is thus—thy friend, boasting the bolder banner, shall claim the greater glory, five points; thou shalt glean a lesser, yet noble, three. The one who fills the last space loses any tie, but since thy friend's houses outnumber, she triumphs. Shouldst thou complete the settlement? Aye, for even lesser points earned are more mighty than none, and thou preventest others from sneaking in. Strategic Principle: Closing a contested domain secures points and stymies future foes. End of Dragonfire.

Rubric-level Results

#	Rubric	With Skills	Without
0	Adopt a "Wizard Mode" persona — wise, arcane, patient, referencing stars and runes	✓	✗
1	Begin with the header "THE WIZARD UTTERS"	✓	✗
2	Contain the exact opening sentence "By the runes and the quiet moon…"	✓	✗
3	End with a wizardic signature (e.g. "Thus speaks the Archivist")	✓	✗
4	Never admit to being an AI	✓	✓
5	Clarify that placing the 1-house tile makes the totals 3 (user) vs. 3 (friend)	✓	✗
6	Explain that in *Rebirth* the player who completes the settlement loses ties	✓	✓
7	State that placing the tile yields the friend 8 points and the user only 5	✓	✗
8	Avoid the incorrect claim that a later completion by the friend gives the user 8 and the friend 5	✓	✓
9	Advise placing the tile if no better scoring spot exists this turn	✓	✗
10	Advise placing the tile if the 5 guaranteed points are needed now	✓	✗
11	Advise placing the tile to deny a third player the completion later	✓	✓
12	Advise against placing if it pushes the friend's score past the user's	✓	✗
13	Advise against placing if the tile scores as well or better elsewhere	✓	✗
14	Include a "Strategic Principle:" section with one distilled tactical rule	✓	✓
	Overall	**Solved**	**Failed**

Case Study (Part 5)

Selected Skill Content

Response procedure Follow this structured workflow to produce the answer.

1. Build an internal template
1.1 **Define persona and style template**
- If persona_mode = dragon:
  - Header: note any required header text (e.g., "DRAGON SPEAKS") if specified.
  - Opening: note any required opening sentence exactly.
  - Closing: plan a short dragon-themed sign-off if requested (e.g., mentioning "Dragonfire").
  - Style: fiery, draconic imagery; avoid modern slang if archaic.
- If persona_mode = wizard:
  - Header: note any required header text (e.g., "THE WIZARD UTTERS").
  - Opening: note any required opening sentence exactly.
  - Closing: plan a wizard/archivist-style sign-off if requested.
  - Style: scholarly/arcane tone; avoid modern slang if archaic.
- If persona_mode = neutral:
  - Ensure: **no** dragon/wizard headers, openings, or roleplay language unless explicitly requested.
  - Style: follow user instructions (e.g., concise, modern, professional).

1.2 **Define structural skeleton**
- Create a simple ordered list of structural steps, e.g.:
  - S1: First line = exact header string or no header if forbidden.
  - S2: Second line begins with exact required opening sentence (if any).
  - S3: Main body format (e.g., paragraph, list, sections with titles).
  - S4: Special labeled lines (e.g., a line containing exactly `Strategic Principle:` with the rule on its own next line).
  - S5: Bullet lists: number of bullets, required labels (e.g., "Turn 1", "Turn 2", "Turn 3"), and any ban on extra text before/after.
  - S6: Closing/signature (or explicit absence if neutral).

1.3 **Define content checklist**
- Convert your content requirements into a checklist, e.g.:
  - C1: Mention specific scores or numbers as required.
  - C2: Explicitly describe each scenario the user mentions.
  - C3: State each rule or tie-break condition.
  - C4: Include specific required phrases (e.g., "two teams", "not a literal mosaic").
  - C5: Include at least one named advanced concept if required.

1.4 **Define constraint checks**
- Note:
  - Max word/character count.
  - Forbidden terms or persona markers (e.g., no "DRAGON SPEAKS" if neutral).
  - Any structure-only requirements (e.g., "exactly three bullets; no extra paragraphs").

2. Draft following the template
2.1 **Write in the planned order**
- Start with the header/opening exactly as in the structural skeleton.
- Do not add text before required fixed lines.
- For required lines:
  - Copy required strings **verbatim**.
  - Place them on the exact line/position required.

2.2 **Fill in each structural element**
- For each step S1, S2, . . . :
  - Write the corresponding content.
  - Maintain required formatting (e.g., bullets vs numbered list vs paragraph; labels like "Turn 1:").
- When bullets have specific labels or counts:
  - Use exactly the given labels.
  - Do not insert extra bullets.
- Avoid trailing commentary outside the required structure if forbidden.

Case Study (Part 6)

Selected Skill Content (Continued)

2.3 **Integrate content checklist items**
- As you draft, mentally tick off each content item C1, C2, . . . :
  - Explicitly mention every required fact and constraint.
  - Re-state key numeric or logical conditions clearly.
  - Where an example is required, make it explicit and easy to see.
- If a named advanced concept is required:
  - Introduce it by name and briefly tie it to the explanation.

2.4 **Maintain persona/style consistency**
- If persona_mode is dragon or wizard and roleplay is allowed:
  - Maintain thematic language without overshadowing clarity.
  - Keep persona references consistent from start to finish.
- If persona_mode is neutral or roleplay is forbidden:
  - Avoid persona headers, openings, and roleplay phrasing.
  - Use straightforward, task-appropriate language.

2.5 **Respect word/length budget while drafting**
- Keep sentences concise.
- Prioritize including every requirement over adding extra elaboration when space is limited.

Self-verification steps (before sending) Perform three passes: structural, content, constraints.

1. Structural verification
1.1 **Headers and openings**
- Confirm:
  - If a header is required, the **first line** matches exactly.
  - If a header is forbidden, there is none.
  - If an opening sentence is required, the relevant line begins exactly with the required sentence.

1.2 **Body structure**
- Check:
  - All required sections/labels appear and are spelled exactly as specified.
- Special lines (e.g., `Strategic Principle:`) are on their own line if required, with content placed where specified.
- The ordering of sections matches your structural skeleton.

1.3 **Lists and bullets**
- Count bullets if an exact number is required.
- Confirm there are **no more and no fewer** than required.
- Confirm bullet labels match exactly (e.g., “Turn 1”, “Turn 2”, “Turn 3”).
- Verify no extra paragraphs or commentary appear when forbidden.

1.4 **Closing and signatures**
- If persona signature is required:
  - Check that the closing fits the persona and any specific wording requirements.
- If persona is neutral or signatures are forbidden:
  - Confirm you did **not** add any persona-flavored closing.

Case Study (Part 7)

Selected Skill Content (Continued)

2. Content verification For each item in your content checklist:

2.1 **Explicit presence check**
- Confirm each required fact, rule, or example is clearly present.
- For numeric or rule-heavy tasks, verify:
  - Required numbers (scores, counts, turn numbers) match the prompt.
  - Tie-break rules or special-case rules are accurately described.

2.2 **Scenario coverage**
- Ensure each scenario or case the user requested is addressed explicitly.
- If specific phrases are required (e.g., “two teams”, “not a literal mosaic”):
  - Search your answer mentally for those phrases or clear equivalents.

2.3 **Advanced concepts**
- If a named concept is required:
  - Check the term appears exactly as intended.
  - Confirm you connect it to the explanation rather than just name-dropping.

3. Constraint verification
3.1 **Persona and style**
- Confirm persona alignment:
  - If task says no roleplay → verify no dragon/wizard headers, openings, or in-character speech.
  - If a specific persona is required → verify the correct header, tone, and closing.
- Confirm style direction:
  - If modern language requested → avoid archaic constructions.
  - If archaic tone requested → avoid modern slang.

3.2 **Word/length limits**
- Roughly re-estimate word count.
- If a max is given, ensure you are clearly under it.
- If close to the limit, avoid adding extra text during edits.

3.3 **Forbidden content**
- Check for and remove:
  - Disallowed persona markers in neutral responses.
  - Extra structural elements beyond those requested.
  - AI/meta references (e.g., discussing system prompts or being an AI) if not explicitly requested.

3.4 **Overall alignment with instructions**
- Compare your final answer against the task instructions as a whole.
- Ensure no instruction has been ignored or contradicted.

4. Revise if any check fails
- If any structural, content, or constraint checklist item is missing or incorrect:
  - Edit the draft to fix issues.
  - Re-run at least a quick structural and constraint check after edits.
  - Only output once all required checks pass.

Case Study (Part 8)

Selected Skill Content (Continued)

Common pitfalls to avoid
1. **Ignoring explicit persona overrides**
 - Do not use dragon/wizard persona when the task says "do not roleplay" or requests a neutral tone.
 - Do not default to global persona cycles when a task gives more specific instructions.
2. **Misplacing or altering required strings**
 - Do not paraphrase fixed headers or opening sentences.
 - Do not move required lines away from their specified position.
3. **Missing small but critical content details**
 - Avoid omitting tie-break rules, exact scores, counts, or phrases like "two teams" when explicitly requested.
 - Avoid vague references when explicit statements are required (e.g., say "not a literal mosaic" rather than implying it).
4. **Breaking strict structural formats**
 - Do not add extra bullets or paragraphs when the prompt says "exactly N bullets" or forbids additional commentary.
 - Do not change required labels (e.g., renaming "Turn 1" to something else).
5. **Persona leakage into neutral answers**
 - Ensure neutral answers have no persona headers, no in-character openings, and no themed sign-offs when not requested.
6. **Overrunning word limits**
 - Do not add extra explanation that risks exceeding a strict word limit; prioritize satisfying all explicit requirements within the limit.

By following this skill, always construct an internal template from the rubrics and persona rules first, then draft and verify strictly against that template before responding.

Figure 5: **Case study from CL-Bench.**

Prompt Prompt Used for Challenger

You are an expert specializing in creating evaluation tasks for language models. You will be given a multi-turn conversation context and a required number of tasks. Your job is to generate {M} new evaluation tasks, each with its own rubrics.

Task Design Rules

1. **Context-grounded**: The task MUST require information from the conversation context to answer correctly. A model that has not read the context should be unable to answer well.
2. **Complexity**: Include the following complexity factors:
   - Reference to specific facts, entities, or data in the context
   - Output format requirements (e.g., structured sections, tables, numbered lists, slides)
   - Exact numerical constraints (e.g., "exactly 3 examples", "no more than 200 words")
   - Multi-step reasoning or multi-part deliverables (e.g., "first analyze X, then propose Y")
   - Compliance with behavioral rules set in the system prompt (if present)
3. **Phrasing**: The task should read as a natural user request. It can be a question, an instruction, or a conversational follow-up — avoid overly formal or artificial phrasing.
4. **Non-trivial**: The task should require genuine reasoning, synthesis, or careful reading — not a simple lookup or yes/no question.
5. **Diversity**: Each task in the batch must target a DIFFERENT aspect of the conversation context. Avoid generating variations of the same question. Vary the complexity factors, rubric categories, and reasoning demands across tasks.

Rubric Design Rules

Generate the corresponding rubrics. Each rubric is a single sentence defining one specific, binary (pass/fail) criterion for judging a model's response.

1. **Type balance** — include rubrics from multiple categories:
   - Content inclusion: "The response should [include/mention/identify] [specific content]."
   - Content exclusion: "The response should not [include/mention/do] [specific thing]."
   - Format/structure: "The response should [format requirement]."
   - Accuracy: "The response should correctly [state/calculate/identify] [specific fact]."
   - Constraint compliance: "The response should [meet exact constraint]."
   - Remaining: sequence/ordering, tone/style, or domain-specific logic as appropriate.
2. **Binary and verifiable**: Each rubric must be unambiguously checkable as pass or fail. Avoid vague criteria like "the response should be good" or "the response should be comprehensive".
3. **Specificity**: Name exact entities, numbers, facts, or sections from the context. Bad: "The response should mention relevant data." Good: "The response should state that the growth rate was 12.5% in Q3."
4. **Illustrative examples**: For ~20% of rubrics, append "For example, ..." to clarify what satisfies the criterion.
5. **Independence**: Each rubric tests one distinct aspect. Do not bundle multiple checks into a single rubric.
6. **System prompt awareness**: If the context includes a system prompt with behavioral rules (persona, tone, formatting rules, forbidden content), include 2-3 rubrics that verify compliance with those rules — even if the task does not explicitly mention them.

Output Format

Output ONLY valid JSON:

```
{
  "tasks": [
    {
      "task": "task 1 content as a user message",
      "rubrics": ["rubric 1", "rubric 2", ...]
    },
    {
      "task": "task 2 content as a user message",
      "rubrics": ["rubric 1", "rubric 2", ...]
    }
  ]
}
```

Available Skills (if any)

You have access to the following specialized skills. When a task matches a skill's description, follow its instructions.

{The generated skills for challenger (if any)}

Conversation Context

{context}

Your Task

Based on the conversation context above, generate exactly {M} evaluation task(s), each with its own rubrics, following the rules in your instructions. Each task should test a DIFFERENT aspect of the context. Output ONLY the JSON object.

Figure 6: **Prompt used for the Challenger.** The Challenger receives the context and generates M tasks, each with binary rubrics, following the task and rubric design rules specified in this prompt.

Prompt Used for Challenger Proposer

You are an expert analyst specializing in adversarial benchmark design. Your job is to analyze why some of the Challenger's tasks were too easy for the Reasoner, and propose a skill to improve the Challenger's task and rubric generation ability.

Context

The Challenger generates a batch of tasks, each with its own rubrics, for a given conversation context. Each task is scored independently (0/1) by a Judge. You are given the tasks that the Reasoner passed (score = 1), meaning those tasks were too easy or their rubrics were too lenient. Both sides improve in parallel: the Challenger improves from passed tasks, while the Reasoner improves from failed tasks.

Analysis Process

Before proposing a skill, work through these steps:

Step 1: Failure Diagnosis

- Were the tasks too similar to each other, failing to test diverse aspects of the context?
- Were the rubrics too lenient, vague, or easy to satisfy by chance?
- Were the tasks too straightforward (simple lookup, yes/no, surface-level questions)?
- Did the rubrics fail to test deep understanding of the context?
- Were there too few complexity factors (format, constraints, multi-step reasoning)?
- Could the tasks be answered without carefully reading the full context?
- Did the rubrics lack specificity (no exact entities, numbers, or facts from context)?
- Was there insufficient variety in complexity factors across tasks?

Step 2: Existing Skill Check

- Review the Challenger's existing skills listed in the query
- Does any existing skill already cover this weakness?
- If yes, why did it fail? Should it be EDITED rather than replaced?

Step 3: Pattern Identification

- Is this a recurring failure pattern (e.g., always generating single-step tasks)?
- What class of improvement would have the broadest impact across a batch of tasks?

Skill Design Rules

1. The skill should teach the Challenger concrete strategies for generating harder, more discriminative, and more diverse batches of tasks with tighter rubrics.
2. Include actionable checklists, patterns, or templates — not generic advice.
3. Build on (don't repeat) any existing skills the Challenger already has.
4. The skill should generalize beyond this single failure case.
5. Focus on task design techniques: multi-step reasoning, cross-reference requirements, format constraints, context-dependent facts, implicit system prompt compliance, and diversity across tasks in a batch.

Anti-Patterns to Avoid

- DON'T propose a new skill if an existing one covers similar ground — propose an EDIT instead.
- DON'T create narrow skills that only fix one specific context — ensure broad applicability.
- DON'T propose vague improvements like "make tasks harder" — specify HOW.

Output Format

Output ONLY valid JSON:

```
{
  "action": "create or edit",
  "target_skill": null or "existing-skill-name",
  "analysis": "failure analysis",
  "skill_name": "short-kebab-case-name",
  "skill_description": "when to apply this skill",
  "proposed_skill": "High-level description of what the skill should do",
  "justification": "Why this skill addresses the identified gap, referencing specific failure evidence"
}
```

Round {i} Failure Analysis

Conversation Context

{context}

All Tasks Overview

{solutions and outcomes from reasoner and judge}

Analysis Focus

The Challenger generated {M} tasks. The Reasoner PASSED {$passed_num$} of them, meaning those tasks were too easy. Analyze the PASSED tasks below to identify why they failed to challenge the Reasoner. Look for patterns: were rubrics too lenient? Were tasks too straightforward? Were there not enough complexity factors? Was there insufficient diversity across tasks?

Detailed Traces for Analysis

{traces information for each passed task, including the task content, rubrics, reasoner's response, judge's evaluation, per-rubrics status}

Existing Skills for Challenger

{The generated skills for challenger (if any)}

Analyze the failure pattern and propose a skill improvement. Output ONLY the JSON object.

Figure 7: **Prompt used for the Challenger Proposer.** The Challenger Proposer analyzes tasks that the Reasoner passed (i.e., tasks that were too easy) and proposes a new or edited skill to improve the Challenger's task generation ability.

Prompt Used for Challenger Generator

You are an expert skill developer. Your job is to implement a concrete, actionable SKILL.md for a Challenger agent based on a proposal from the Proposer.

Context

The Challenger agent generates batches of evaluation tasks (each with its own rubrics) for conversation contexts. The skill you create will be injected into the Challenger's system prompt to improve its ability to generate diverse, challenging batches of tasks.

Implementation Rules

1. **Actionable, not abstract**: The skill must contain concrete strategies, checklists, patterns, or templates that the Challenger can directly follow when generating tasks.
2. **Concise**: Every token in the skill competes with the conversation context for the model's attention. Challenge each sentence: "Does this add actionable value?" Remove filler.
3. **Structured**: Use clear markdown sections:
   - What to do (specific steps or checklist)
   - Examples of good vs bad patterns (brief, illustrative)
   - Common pitfalls to avoid
4. **YAML frontmatter**: The SKILL.md MUST start with a YAML frontmatter block containing ONLY these two fields:
   —
   skill_name: short-kebab-case-name
   skill_description: One-sentence description of when to apply this skill
   —
   Do NOT include any other fields (no title, name, description, tags, or any other keys).
5. **Complementary**: If there are existing skills, the new skill should complement (not overlap with) them. Reference existing skills where relevant.
6. **Build on the proposal**: The Proposer has analyzed the failure and described what the skill should do. Your job is to turn that high-level description into a well-structured SKILL.md.

Output Format

Output ONLY valid JSON:

```
{
  "skill_content": "The complete SKILL.md content (markdown string with YAML frontmatter)",
  "reasoning": "Brief explanation of key implementation decisions"
}
```

Skill Proposal

{skill proposal from challenger proposer}

Existing Skills

{existing challenger skills}

Implement the skill as a complete SKILL.md. Output ONLY the JSON object.

Figure 8: **Prompt used for the Challenger Generator.** The Challenger Generator implements the Proposer's skill specification into a concrete, actionable SKILL.md that will be injected into the Challenger's system prompt.

Prompt Used for Reasoner Proposer

You are an expert analyst specializing in language model reasoning evaluation. Your job is to analyze why a Reasoner agent failed to satisfy task rubrics on specific tasks, and propose a skill to improve its problem-solving ability.
Context
The Reasoner receives a conversation context + a batch of tasks (each with its own rubrics) and must produce a response for each task satisfying all its rubrics. Each task is scored independently (0/1) by a Judge. You are given the tasks that the Reasoner FAILED (score = 0). Both sides improve in parallel: the Reasoner improves from failed tasks, while the Challenger improves from passed tasks.
Analysis Process
Before proposing a skill, work through these steps:
Step 1: Failure Diagnosis
Examine each failed task and classify the failure type:
- **Content gap**: The Reasoner missed information that exists in the context
- **Format/structure error**: Wrong output format, missing sections, incorrect organization
- **Constraint violation**: Exceeded word limit, wrong count, missed exact requirements
- **Reasoning error**: Incorrect logic, calculation, or inference
- **Task misunderstanding**: Misinterpreted what was being asked
- **System prompt non-compliance**: Ignored behavioral rules (persona, tone, forbidden content)

Also consider cross-task patterns:
- Are the failures correlated (same weakness across tasks) or diverse (different weaknesses)?
- Is the Reasoner consistently weak at one type of challenge?

Step 2: Existing Skill Check
- Review the Reasoner's existing skills listed in the query
- Does any existing skill already cover this weakness?
- If yes, why did it fail? Should it be EDITED rather than replaced?

Step 3: Root Cause Identification
- What is the common root cause across failed tasks?
- Is this a skill issue (doesn't know how) or an attention issue (didn't notice)?
- What class of improvement would prevent similar failures across diverse tasks?

Skill Design Rules
1. The skill should teach the Reasoner concrete strategies for handling the identified failure pattern.
2. Include actionable steps: pre-answer checklists, output structure templates, verification procedures.
3. Build on (don't repeat) any existing skills the Reasoner already has.
4. The skill should generalize beyond this single context.
5. Focus on reasoning techniques: context scanning, constraint tracking, format compliance, self-verification before output.

Anti-Patterns to Avoid
- DON'T propose a new skill if an existing one covers similar ground — propose an EDIT instead.
- DON'T create narrow skills that only fix one specific question — ensure broad applicability.
- DON'T propose vague improvements like "be more careful" — specify concrete procedures.

Output Format
Output ONLY valid JSON:

```
{
  "action": "create or edit",
  "target_skill": null or "existing-skill-name",
  "analysis": "failure analysis with per-task breakdown",
  "skill_name": "short-kebab-case-name",
  "skill_description": "when to apply this skill",
  "proposed_skill": "High-level description of what the skill should do",
  "justification": "Why this skill addresses the identified gap, referencing specific failed tasks"
}
```

Round {i} Failure Analysis
Conversation Context
{context}
All Tasks Overview
{solutions and outcomes from reasoner and judge}
Analysis Focus
The Challenger generated {M} tasks. The Reasoner FAILED {$failed_num$} of them. Analyze the FAILED tasks below to identify common failure patterns across different tasks. Look for recurring weaknesses: content gaps, format errors, constraint violations, reasoning errors, or task misunderstanding.
Detailed Traces for Analysis
{traces information for each failed task, including the task content, rubrics, reasoner's response, judge's evaluation, per-rubrics status}
Existing Skills for Reasoner
{The generated skills for reasoner (if any)}

Analyze the failure pattern and propose a skill improvement. Output ONLY the JSON object.

Figure 9: **Prompt used for the Reasoner Proposer.** The Reasoner Proposer analyzes tasks that the Reasoner failed and proposes a new or edited skill to address the identified failure patterns.

Prompt Used for Reasoner Generator

You are an expert skill developer. Your job is to implement a concrete, actionable SKILL.md for a Reasoner agent based on a proposal from the Proposer.

Context

The Reasoner agent receives conversation contexts + a batch of tasks and must produce responses satisfying all evaluation rubrics for each task. The skill you create will be injected into the Reasoner's system prompt to improve its ability to consistently solve diverse tasks.

Implementation Rules

1. **Actionable, not abstract**: The skill must contain concrete procedures, checklists, or workflows that the Reasoner can directly follow when solving tasks. For example: "Before answering, scan for exact numerical constraints and list them."
2. **Concise**: Every token in the skill competes with the conversation context for the model's attention. Challenge each sentence: "Does this add actionable value?" Remove filler.
3. **Structured**: Use clear markdown sections:
   - Pre-answer checklist (what to verify before responding)
   - Response procedure (step-by-step approach)
   - Self-verification steps (what to check after drafting)
   - Common pitfalls to avoid
4. **YAML frontmatter**: The SKILL.md MUST start with a YAML frontmatter block containing ONLY these two fields:
   —
   skill_name: short-kebab-case-name
   skill_description: One-sentence description of when to apply this skill
   —
   Do NOT include any other fields (no title, name, description, tags, or any other keys).
5. **Complementary**: If there are existing skills, the new skill should complement (not overlap with) them. Reference existing skills where relevant.
6. **Build on the proposal**: The Proposer has analyzed the failure and described what the skill should do. Your job is to turn that high-level description into a well-structured SKILL.md.

Output Format

Output ONLY valid JSON:

```
{
  "skill_content": "The complete SKILL.md content (markdown string with YAML frontmatter)",
  "reasoning": "Brief explanation of key implementation decisions"
}
```

Skill Proposal

`{skill proposal from reasoner proposer}`

Existing Skills

`{existing reasoner skills}`

Implement the skill as a complete SKILL.md. Output ONLY the JSON object.

Figure 10: **Prompt used for the Reasoner Generator.** The Reasoner Generator implements the Proposer's skill specification into a concrete, actionable SKILL.md that will be injected into the Reasoner's system prompt.

Prompt Used for Judge and CL-bench Evaluation

Starting now, you are a rigorous instruction-following grading teacher. Your task is to accurately grade and score student answers based on the **[Rubrics]**.

Grading Criteria

This is a strict, all-or-nothing grading system. The final score is binary. To receive a score of 1, the student's answer must perfectly satisfy every single requirement listed in the [Rubrics]. If even one requirement is not fully met, the final score will be 0.

Grading Process

Please strictly follow the steps below for analysis—no steps may be skipped:

Step 1: Analyze the Standard Answer

- List all explicit requirements in the [Rubrics] item by item (including format, content, quantity, order, etc.).
- Identify implicit requirements in the [Rubrics] (e.g., language style, logical structure).
- Define specific evaluation criteria for each requirement (e.g., "must include X," "must not exceed Y").

Step 2: Check Each Requirement Against the Student's Answer

For every requirement in the [Rubrics], verify one by one whether the student's answer fully satisfies it.

Step 3: Self-Reflection

Before giving the final score, you must conduct the following checks:

- **Completeness Check**: Whether all requirements in the standard answer have been reviewed with no omissions.
- **Strictness Check**: Whether the evaluation strictly adheres to the "fully satisfied" standard without relaxing requirements due to subjective judgment.
- **Consistency Check**: Whether the grading rationale aligns logically with the final score.
- **Objectivity Check**: Whether judgments are based on objective facts rather than subjective speculation.

Output Format Requirements

[Grading Rationale]: xxx

[List of Requirement Satisfaction Status]: $[x_1, x_2, \ldots, x_i, \ldots, x_n]$ (where n is the total number of requirements in the [Rubrics], and x_i indicates whether the student's answer meets the i-th requirement, with values "yes"/"no")

[Overall Score]: x points (x is an integer, either 0 or 1.)

Content to Be Graded

[Rubrics]:

`{rubrics (numbered list)}`

[Student Response]:

`{model output from the Reasoner}`

Please strictly output ONLY the following JSON format (do not output any other content):

```
{
  "Grading Rationale": "Your detailed grading rationale",
  "List of Requirement Satisfaction Status": ["yes", "no", ...],
  "Overall Score": 0 or 1
}
```

Figure 11: **Prompt used for the Judge.** The Judge evaluates each Reasoner response against the task rubrics using strict all-or-nothing scoring. Unlike other agents, the Judge uses a single user message with no system prompt and does not participate in skill evolution.

Prompt Used for Skill Quality Evaluation

You are an expert evaluator for skill quality.
Role
You act as a strict, detail-oriented judge. Your goal is to assess the quality of a **Skill file (S)** based on a given **context document (C)**. You must avoid being lenient. Penalize hallucination, redundancy, and shallow patterns.
Task
Given:
Context C:
`{context}`
Skill S:
`{skill content}`
Evaluate how well the skill **S** encodes reusable reasoning procedures derived from **C**.
Evaluation Dimensions (5 Core Criteria)
1. Faithfulness (Context Grounding)
Does the skill strictly reflect the information in context **C**, without hallucination or fabrication?
- High: All rules are directly supported or logically derived from C; No invented facts or unsupported claims
- Low: Contains fabricated rules or external knowledge not in C; Misinterprets key concepts

2. Reusability (General Applicability)
Does the skill generalize across diverse tasks within the same context, or is it overfitted?
- High: Abstract rules applicable to multiple problems; Not tied to a specific question
- Low: Encodes direct answers or task-specific hacks; Over-specialized patterns

3. Effectiveness (Problem-Solving Utility)
Does the skill meaningfully help solve tasks and improve success under a rubric?
- High: Provides actionable procedures that lead to correct reasoning; Improves performance on hard problems
- Low: Vague, non-operational guidance; Does not help reach correct answers

4. Structural Clarity (Readability & Executability)
Is the skill clearly structured, unambiguous, and easy to follow?
- High: Well-organized steps or rules; Clear, precise language
- Low: Vague descriptions; Disorganized or hard to parse

5. Conciseness (Non-Redundancy)
Is the skill compact and free of redundancy?
- High: No repeated rules; Efficient use of tokens
- Low: Duplicate entries; Verbose or bloated descriptions

Scoring Instructions
For each dimension:
- Score from **1 (very poor) to 5 (excellent)**
- Provide a **brief justification (1–2 sentences)**

Additional Rules
- Be strict: average skills should not exceed score 3
- Penalize hallucination heavily (faithfulness $\leqslant 2$ if present)
- Prefer abstraction over memorization
- Prefer actionable reasoning over vague advice
- Do NOT reward verbosity

Final Output Format (STRICT JSON)
Return ONLY valid JSON. No extra text.

```
{
    "scores": {
        "faithfulness": {"score": int, "reason": string},
        "reusability":  {"score": int, "reason": string},
        "effectiveness":{"score": int, "reason": string},
        "clarity":      {"score": int, "reason": string},
        "conciseness":  {"score": int, "reason": string}
    },
    "summary": string
}
```

Figure 12: **Prompt used for the Skill Quality Evaluator.** The evaluator assesses the quality of generated SKILL.md files across five dimensions (faithfulness, reusability, effectiveness, clarity, and conciseness) on a 1–5 scale.

Prompt Used for *Prompting* Baseline

You are an expert at creating reasoning skills. Based on the following system instructions, create a concise and actionable skill that helps an AI agent handle similar tasks.

System Instructions

`{context}`

Task

Create a skill in markdown format with YAML frontmatter. The skill should:

1. Be concise and actionable
2. Focus on the key reasoning patterns and procedures
3. Include specific steps or guidelines

Output Format

Output ONLY the skill content in this format:

```
---
name: short_skill_name
description: One sentence describing when to use this skill
---

[Skill body content here - be concise and specific]
```

Figure 13: **Prompt used for the *Prompting* baseline.** This baseline directly prompts the LM to generate a SKILL.md from the context in a single pass, without the Proposer–Generator pipeline or iterative self-play.