Analyzing Agent Skills: A Four-Dimensional Framework for Structuring and Evaluating Skills

Anonymous ACL submission

Abstract

As agents evolve from single-tool assistants into modular, multi-component systems, Skills have become a central vehicle for constructing and distributing capabilities, and their importance is increasingly salient. However, the academic community has yet to establish a analytical framework for systematically analyzing and evaluating Skills. This paper first examines the position and function of Skills in agent systems from a mechanistic perspective, identifying two core mechanisms: information gain and behavioral constraint. Building on this analysis, we propose **MCRI**, an analytical framework for characterizing Skills along four dimensions. We further demonstrate its utility through two types of empirical tasks. Finally, we discuss the framework's scope of applicability and potential directions for future extension.

1 Introduction

AI agents are transforming the way people work at an extraordinary pace and have already become a mainstream component of contemporary workflows. Taking software engineering as an example, 56% of engineers use AI agents in at least half of their software engineering work (**?**). These agents are gradually evolving from single auxiliary modules into composite systems composed of multiple reusable capability modules. For instance, on October 16, 2025, Anthropic introduced Agent Skills (Anthropic, 2025) and subsequently packaged several of its built-in capabilities as Skills. Major vendors soon followed (OpenAI, 2026; Google Developers, 2026; Cursor, 2026), building modular capability ecosystems around MCP, Skill Hubs, and Skill installation interfaces. At the time of writing, LobeHub's "Skills Marketplace for Every Agent" has collected more than 290,000 Skills (LobeHub, 2026).

These developments indicate that Skills are moving beyond auxiliary configurations in engineering practice and are becoming an important type of component in the capability architecture of agent systems. Although skills are becoming increasingly important in the practice of intelligent agent systems, the academic and engineering communities currently lack a structured perspective for systematically analyzing, organizing, and evaluating skills. Why are Skills effective? What kinds of Skills are more likely to become popular? Traditional large language model evaluations usually take the model itself as the object of evaluation, focusing on its knowledge, reasoning, coding, mathematical, or instruction-following abilities. Agent evaluations, by contrast, tend to emphasize end-to-end task completion rates, tool-use capabilities, and planning and execution abilities. Software testing focuses on whether a program satisfies explicit specifications under given inputs. However, a Skill is not fully equivalent to any of these objects. It is neither model parameters, nor an ordinary prompt, nor a deterministic traditional software program. Instead, it is better understood as a reusable capability structure injected into an agent system at inference time, and therefore requires a new perspective for analysis and evaluation.

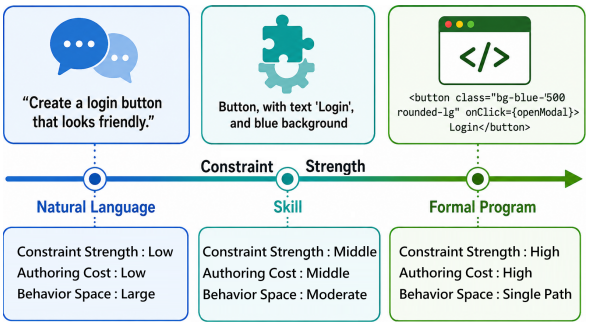

Figure 1: Constraint strength spectrum from natural language to Skills and formal programs.

To address this question, this paper proposes an analytical framework for characterizing skills—the *MCRI Framework*. As shown in Figure 1, we re-

gard a Skill as an intermediate structure between programming languages and natural language. We provide a formal definition of Skills and characterize their influence on large language model behavior from the perspective of constraints. Specifically, we argue that the practical function of a Skill is mainly reflected in two aspects. First, a Skill introduces information external to the model, including task context, domain knowledge, tool descriptions, examples, and external resources. Second, a Skill reduces the instability of model behavior through task decomposition, procedural specifications, interface constraints, boundary conditions, and exception-handling mechanisms. In other words, a high-quality Skill does not merely provide more text to the model. Rather, it enables the model to complete a specific family of tasks in a more stable, controllable, and reusable manner by providing effective information gain and appropriate behavioral constraints.

To answer this question, this paper proposes the MCRI Framework. As shown in the constraint-spectrum figure, we consider Skill as an intermediate structure between programming languages and natural languages, providing a formal definition of Skill and characterizing its impact on the behavior of large language models from the perspective of "constraints." Specifically, we believe that the actual function of Skill is mainly reflected in two aspects: First, Skill introduces external information to the model, including task context, domain knowledge, tool descriptions, examples, and external resources; second, Skill reduces the instability of model behavior through task decomposition, process specification, interface constraints, boundary conditions, and exception handling mechanisms. In other words, a high-quality Skill does not simply provide the model with more text, but rather enables the model to complete tasks more stably, controllably, and reusably within a specific task family through effective information gain and appropriate behavioral constraints.

Based on this assumption, we explored the usability of this framework. We constructed a dataset using over 60,000 skills and their associated information from the OpenClaw Skill Hub and conducted a two-tiered exploration. First, we validated the framework's effectiveness in predicting more popular and well-maintained skills in the real-world ecosystem; second, we set up several downstream tasks to examine the correlation between the theoretical scores derived by the framework and task performance.

2 Related Work

2.1 Large Language Models and Agents Perspective

As artificial intelligence models, large language models are often defined and used as probabilistic models that function as task-specific function bodies. Their evaluation mainly relies on static benchmarks, which are used to measure the intrinsic knowledge and logical reasoning abilities encoded in model parameters (Mohammadi et al., 2025). Agents, by contrast, are often defined as systems capable of perceiving the environment, making decisions, and taking actions, thereby interacting with the environment in practice. Their evaluation typically focuses on end-to-end task completion rates, multi-turn interaction capabilities, and behavioral consistency in diverse tasks and complex scenarios (Yehudai et al., 2025), as exemplified by AgentBench (Liu et al., 2024) and AgentBoard (Ma et al., 2024).

However, both paradigms differ substantially from Skills. A large language model is more like a "brain," an agent is more like a "human," and a Skill is more like a "tool" held by that human. From an evaluation perspective, traditional LLM evaluation focuses on the intrinsic capabilities of the model itself. Although agent evaluation involves external components, Skills remain pluggable add-ons, and evaluation results depend heavily on the architecture, planning strategy, and tool orchestration mechanisms of the host agent. Therefore, whether in terms of usage, evaluation, or analysis of effectiveness, Skills should be treated as a distinct object of study.

2.2 Prompt Engineering Perspective

Because the physical core of a Skill consists of a set of logical instructions written in natural language and structured specifications, Skills are often compared with prompts. There has already been substantial research on prompts, such as the prompt robustness evaluation framework PromptRobust (Zhu et al., 2023). However, prompts and Skills still differ in many respects. First, prompts generally lack reusability and are usually designed for a specific task instance, whereas Skills enable a model to acquire a stable and callable capability for a class of tasks. In addition, a Skill is usually a more stable and structured capability configuration, in-

cluding tool invocation methods, file-processing rules, API-calling rules, and other specifications. These functions are more tightly integrated with agents, while ordinary prompts typically do not include such tool-use mechanisms.

Therefore, it is inappropriate to simply regard Skills as more advanced prompts. Skills are a distinctive subset of prompts with significant differences and should have their own concepts, perspectives, and evaluation methods. From an evaluation perspective, most prompt evaluation frameworks assume that the input is a single, flat textual prompt. They neither involve tool invocation and external interface specifications nor handle multi-file structures, preconditions, and exception branches, which are precisely the key features that distinguish Skills from ordinary prompts. Consequently, theories and evaluation methods developed for prompts cannot be directly transferred to Skills.

2.3 Software Engineering Perspective

From a software perspective, Skills play a role in complex agent systems that resembles third-party libraries in traditional software. Product quality models such as ISO/IEC 25010 have long served as international benchmarks for defining, evaluating, and quantifying software product quality (EmergentMind, 2025). However, their dimensions, functions, and usage scenarios differ greatly from those of Skills. In addition, quality evaluation in software engineering is more like evaluating how humans use tools rather than evaluating the tools themselves. Therefore, understanding Skills solely from the perspective of software engineering remains incomplete.

3 How do Skills Work?

3.1 Where Does a Skill Sit Between Prompts and Code?

To understand the essential characteristics of Skills, we examine them from a broader perspective. We observe that the instructions humans use to guide computational systems in completing tasks can be arranged along a spectrum according to their degree of behavioral constraint. As shown in Figure 1, the two ends of this spectrum are natural language conversation and formal programs, while Skills occupy an intermediate position between them.

Natural language conversation. Humans often express their needs in everyday language. For example, a request such as "help me organize this data" imposes very weak constraints: it specifies neither how the data should be organized, nor the output format, nor the execution steps. Other humans or models must infer the user's intent, such as the intended focus, choose an execution strategy, and determine the form of the output. This gives the model substantial flexibility, but also leads to high behavioral uncertainty: the same instruction may result in markedly different outputs across different inference runs.

Programming languages. A Python function, by contrast, explicitly specifies input types, each operation, every branch condition, and the output format. Its degree of constraint is extremely high: the behavior space is ideally compressed into a single deterministic path. A program does not rely on the executor's "understanding" or "inference"; it only needs to be executed step by step. The cost of this determinism, however, is high authoring effort and limited generalizability: a program cannot handle cases outside its specification, nor can it leverage the executor's background knowledge and reasoning ability.

Skills. The distinctive feature of Skills lies in their moderate degree of constraint. A Skill constrains the behaviors that humans want to make determinate, while leaving other parts to the model's autonomous reasoning. Skills do not attempt to eliminate all uncertainty in model behavior. Instead, they selectively compress the regions of the behavior space that humans do not want, while preserving the freedom necessary for the model to exercise its reasoning ability. In this sense, a Skill is somewhat analogous to a requirements document in the software development workflow: it specifies what the system should do and under what conditions it should be completed in what manner, but it does not prescribe how every line of code should be written.

3.2 How Skills Work: Constraint and Information Injection

Building on the above positioning, we further analyze why Skills can improve an agent's performance on particular families of tasks. We identify two complementary core mechanisms.

3.2.1 Information Gain

The parameters of a large language model encode knowledge covered by its training data. However, in any specific task scenario, the model is almost certain to face information gaps. These gaps may

involve the parameter format of an internal API, the directory structure and coding conventions of the current project, or the latest version changes of a particular library. Such information cannot be compensated for simply by stronger reasoning ability, because it does not exist in the model's parameters in the first place.

For example, a Python script attached to a PPTX generation Skill may contain concrete usage patterns of the `python-pptx` library and code for manipulating presentation templates. Even the strongest models cannot reliably derive such information from parametric knowledge alone. We refer to this mechanism as **information gain**. Its core function is to expand the model's effective knowledge scope from parametric knowledge alone to the union of parametric knowledge and the information injected by the Skill, thereby ensuring that the correct behavioral path first enters the model's space of consideration.

3.2.2 Behavioral Constraint

Even when a model has all the information required to complete a task, it does not necessarily choose the optimal behavioral path in every inference run. The generation process of large language models is probabilistic: given the same input, the model may select different execution strategies, different output formats, or even different ways of decomposing the problem across different sampling trajectories. Many of these paths may be "reasonable," but only some are better suited to the specific requirements of the current task scenario.

For example, suppose a user asks the model to analyze a CSV file. Even if the model is fully capable of completing this task, in the absence of constraints, it might first handle missing values and then create visualizations in one run, jump directly to statistical testing in another, and begin by outputting a descriptive summary before asking the user for clarification in a third. None of these behaviors is necessarily "wrong," but such inconsistency is unacceptable for a reusable tool.

Therefore, if a Skill requires the large language model to first perform data preprocessing, then conduct data analysis, and finally output a report in a specified format, it is essentially compressing the model's legitimate behavior space and increasing the probability of the desired behavior within that space. We refer to this mechanism as **behavioral constraint**. Its core function is to improve the consistency and reliability with which the model

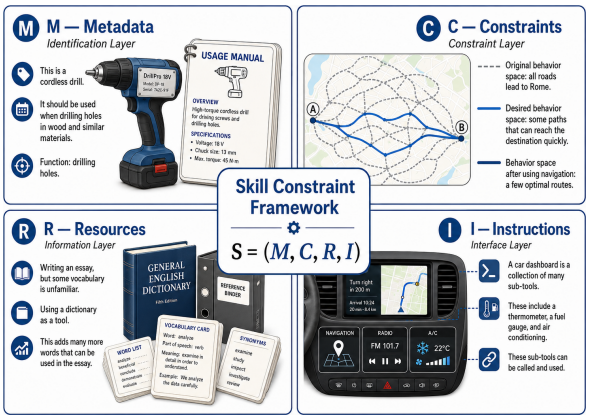

Figure 2: Example of the MCRI Framework.

selects the target behavioral path across repeated invocations, assuming that the model already possesses the capability to complete the task.

3.3 MCRI Framework: A Skill Analytical Framework

Based on the two mechanisms identified in Section 3.2, we represent MCRI Framework as a quadruple.

$$S = (M,\ C,\ R,\ I) \tag{1}$$

M **(Metadata) — Identification Layer.** Describes the task definition, functional positioning, applicable scenarios, triggering conditions, and input/output specifications of the Skill. It declares to the host agent both "what this Skill is for" and "under what circumstances it should be picked up."

C **(Constraints) — Constraint Layer.** Corresponds to the behavioral-constraint mechanism described in Section 3.2. This is the part of the Skill that explicitly compresses the model's space of admissible behaviors, including prohibitive rules, preconditions, invariants, and handling strategies.

R **(Resources) — Information Layer.** Corresponds to the information-gain mechanism described in Section 3.2. This is the part of the Skill that injects information beyond the model's parameters, including template files, configuration scripts, example data, reference documents, and error traces accumulated by others during execution.

I **(Instructions) — Interface Layer.** Specifies the internal scripts, external tools, and sub-Skills that the Skill may invoke during execution, as well as the concrete modes of interaction with its environment, including the form of invocation interfaces, the sequencing of steps, and the logic of sub-task decomposition.

The focus of this framework is to provide an analytical perspective on Skills, rather than to classify a Skill file into a specific dimension. Figure **??** presents a simple example illustrating the role played by each dimension. For instance, for a person who urgently needs to get to school, the original behavior space may contain infinitely many possible routes, whereas the desired behavior space consists only of a few routes that can reach the school quickly. Once a navigation tool is used, the behavior space is compressed into a small set of optimal routes.

4 How well do Skills work?

4.1 From Mechanism to Application

If the proposed framework captures functional properties of Skills, scores derived from it should correlate with independent external signals. We therefore conduct two validation studies.

Ecological-signal correlation test. This task selects externally observable community signals of a Skill, denoted as f, such as stars, downloads, and installs, and examines whether framework-based LLM scores can predict this indicator.

Downstream-task correlation test. This task invokes each Skill in a controlled downstream task environment to complete representative tasks and measures its actual performance, thereby examining the correlation between theoretical scores and task performance.

4.2 MCRI-Eval

We operationalize the MCRI Framework into an evaluation heuristic, termed **MCRI-Eval**. Each layer of the framework is mapped to an evaluation dimension that assesses how effectively the corresponding layer fulfills its prescribed function. Specifically, Metadata is mapped to task-specification clarity, Constraints to behavioral-constraint sufficiency, Resources to information-gain effectiveness, and Instructions to instruction-and-interface specification. The scoring prompt includes the definition of the MCRI Framework and a lightweight scoring guide for each dimension; the full prompt is provided in the supplementary materials.

4.3 Community-Interaction Signal Identification Test

4.3.1 Task Definition

Given a skill S_i, the LLM outputs a four-dimensional score vector $\mathbf{L}_i \in R^{1\times 4}$ according to the MCRI-Eval. On the training set, we use the community metric f_i as the target and perform linear fitting on $\mathbf{L}_i$ to obtain the aggregated LLM scoring function g_i. Finally, we apply this function to the test set and evaluate its correlation and ranking accuracy by comparing the within-group rankings of $\{f_i\}$ and $\{g_i\}$.

Correlation identification. Task T1 calculates the Spearman correlation coefficients and significance for four dimensions and several community indicators.

Extreme-category identification. Task T2 uses the bottom 15% as the low signal group and the top 15% as the high signal group to calculate the classification results.

4.3.2 Experimental Setup

Dataset. We collected 63,812 public Skills from the OpenClaw Skill Hub as our experimental dataset, together with the content of each Skill and a set of platform-level observational signals. Raw download and installation counts may be affected by the time since a Skill was released and by platform exposure. Moreover, the dataset exhibits a clear long-tailed distribution, with most community signals being zero. To mitigate the interference of these factors in the correlation analysis, we computed the average daily downloads for all samples and conducted the analysis only on the top-ranked Skills.

We randomly split all samples into training and test sets at a 1:1 ratio, using a fixed random seed (`random_state=42`) to ensure reproducibility. The training set was used for fitting, while the test set was reserved exclusively for final evaluation.

Scoring Models. We used models from the Mimo series (`mimo-v2.5` and `mimo-v2.5-pro`) and the GPT series (`gpt-5.4-nano` and `gpt-5-mini`) as scoring models. Each Skill was scored along four dimensions according to the proposed evaluation heuristic. The GPT series refers to models developed by OpenAI, while the Mimo series refers to models developed by Xiaomi. The Mimo models have achieved strong performance on evaluations such as GDPval-AA.

4.3.3 Experimental Results

This section addresses the core research question: can LLM-based four-dimensional scoring of Skill text alone identify the Skills that are more popular in the real ecosystem? We examine the predictive ability of the four scoring models at three different filtering thresholds (top-3% / top-1% / top-0.5%), in order to observe how the strength of correlation varies with the level of head-sample concentration.

Overall trend of correlation across thresholds. Table 1 reports, for each community signal at each threshold, the Spearman correlation coefficient of the best-performing model among the four scoring models. As the filtering threshold tightens, the correlation increases monotonically across all signals, suggesting that the variation among tail samples contains noise and that community indicators are difficult to use as a quality proxy when tail samples are included.

Cross-model consistency. Table 2 further reports the correlations of the four scoring models on each signal at the top-1% threshold. At this threshold, all indicators reach significance level *** across all models, and the correlation differences among models fall within the range of 0.06–0.10; no case is observed in which the conclusion is reversed by the choice of scoring model. This indicates that the observed correlations stem primarily from identifiable differences in the Skill text itself along the four dimensions of the framework, rather than from the scoring bias of any particular model.

Dimension-level ablation. To examine the contribution of each of the four dimensions, we removed one dimension at a time, refit g, and compared the resulting correlation changes on the top-1% samples. The results are shown in Table 3. The contributions of different dimensions exhibit clear functional differentiation across signals: after removing `information_gain`, the correlations with the stars signal became non-significant for 3 out of the 4 scoring models, with ρ decreasing from 0.13–0.21 to 0.04–0.06. This indicates that `information_gain` has an almost exclusive predictive contribution to the stars signal, which captures user attention.

After removing `interface_completeness`, the correlations with installsAllTime dropped substantially, suggesting that this dimension makes an exclusive contribution to predicting actual installation behavior. Removing `constraint_sufficiency` led to small but consistent decreases for both signals, indicating that this dimension provides a stable but moderate marginal contribution. In contrast, removing `clarity` caused only limited declines, suggesting that its discriminative power may be weaker than that of the other dimensions.

This differentiated pattern echoes the two mechanisms proposed in this paper. `information_gain` corresponds to the "information-gain" mechanism: Skills that provide domain knowledge and external information are more likely to attract user attention. `interface_completeness` corresponds to the "behavioral-constraint and execution-reliability" mechanism: Skills with complete interfaces that can be reliably invoked by the host agent are more likely to be installed in practice. These two mechanisms are externally validated by two independent community signals.

Extreme-category identification. We further examine whether the scores can distinguish the most popular subset within the head-Skill set. Within the top-1% samples, we use the top 15% and bottom 15% of each community signal as thresholds to partition the high- and low-signal groups, and use the four-dimensional scores via the fitted function g to perform a binary classification prediction. As shown in Tables 4, the ranking of identification performance across the three signals is consistent with the ranking of correlation strength in Tables 1 and 2.

Conclusion. The above results show that the four-dimensional scores based on the MCRI-eval exhibit a significant positive correlation with Skill popularity across multiple independent community signals, multiple scoring models, and multiple filtering thresholds; furthermore, the correlation between information such as the length and number of files based on text metadata and popularity metrics is weak or negative, indicating that MCRI-eval extracts information beyond the text metadata.

4.4 Downstream-Task Correlation Test

Given a set of Skills $U = \{S_i\}_{i=1}^{N}$, for each Skill $S_i \in U$, the LLM produces four-dimensional scores $\mathbf{L}_i \in R^{1\times 4}$ according to MCRI-Eval, and the average score g_i is computed. In parallel, for each Skill S_i, we randomly sample 50 subtasks from its corresponding benchmark evaluation set, execute the Skill, and compute the average downstream task score h_i. We then compute the within-group rankings based on $\{g_i\}$ and $\{h_i\}$ respectively, and compare the consistency between the

F	top-3% (N=1823)	sig	top-1% (N=606)	sig	top-0.5% (N=305)	sig
stars best	0.166 (gpt-nano)	***	0.213 (gpt-nano)	***	0.366 (gpt-nano)	***
downloads best	0.132 (gpt-mini)	***	0.211 (gpt-nano)	***	0.324 (mimo-pro)	***
installsAllTime best	0.200 (gpt-nano)	***	0.329 (gpt-nano)	***	0.416 (gpt-nano)	***

Table 1: Strongest correlation (Spearman ρ) at each threshold; the corresponding scoring model is shown in parentheses.

F	mimo-pro	sig	mimo-v2.5	sig	gpt-nano	sig	gpt5-mini	sig
stars	+0.203	***	+0.152	**	+0.213	***	+0.134	*
downloads	+0.165	**	+0.183	**	+0.211	***	+0.082	ns
comments	+0.156	**	+0.152	**	+0.270	***	+0.208	***
installsAllTime	+0.280	***	+0.267	***	+0.329	***	+0.226	***

Table 2: Correlation (Spearman ρ) of the four scoring models at the top-1% threshold.

Table 3: Ablation results and Baseline calibrators.

(a) stars

Deleted	mimo-pro	mimo-v2.5	gpt-nano	gpt5-mini
Full-4	$+0.203^{***}$	$+0.152^{**}$	$+0.213^{***}$	$+0.134^{*}$
-clarity	$+0.199^{***}$	$+0.169^{**}$	$+0.182^{**}$	$+0.124^{*}$
-info_gain	$+0.036^{ns}$	$+0.057^{ns}$	$+0.172^{**}$	$+0.039^{ns}$
-constraint	$+0.169^{**}$	$+0.140^{*}$	$+0.172^{**}$	$+0.096^{ns}$
-interface	$+0.192^{***}$	$+0.098^{ns}$	$+0.162^{**}$	$+0.127^{*}$

(b) installsAllTime

Deleted	mimo-pro	mimo-v2.5	gpt-nano	gpt5-mini
Full-4	$+0.280^{***}$	$+0.267^{***}$	$+0.329^{***}$	$+0.226^{***}$
-clarity	$+0.277^{***}$	$+0.268^{***}$	$+0.288^{***}$	$+0.234^{***}$
-info_gain	$+0.274^{***}$	$+0.265^{***}$	$+0.334^{***}$	$+0.181^{**}$
-constraint	$+0.254^{***}$	$+0.234^{***}$	$+0.316^{***}$	$+0.198^{***}$
-interface	$+0.155^{**}$	$+0.087^{ns}$	$+0.165^{**}$	$+0.112^{ns}$

(c) Baseline calibrators

Feature	stars	downloads	comments	installsAllTime
textChars	$+0.011^{ns}$	-0.071^{ns}	-0.002^{ns}	$\mathbf{-0.253}^{***}$
n_files	$+0.062^{ns}$	-0.061^{ns}	-0.119^{**}	-0.029^{ns}

Table 4: Extreme-category identification performance at the top-1% threshold.

(a) installsAllTime

Model	AUC	Accuracy	F1
mimo-pro	0.758	57.5%	0.644
mimo-v2.5	0.719	62.9%	0.581
gpt-nano	0.819	76.2%	0.747
gpt5-mini	0.716	68.6%	0.593
Avg.	0.753	66.9%	0.641

(b) downloads

Model	AUC	Accuracy	F1
mimo-pro	0.734	64.6%	0.623
mimo-v2.5	0.746	67.1%	0.658
gpt-nano	0.756	69.5%	0.706
gpt5-mini	0.676	64.6%	0.603
Avg.	0.728	66.4%	0.648

(c) stars

Model	AUC	Accuracy	F1
mimo-pro	0.653	60.2%	0.545
mimo-v2.5	0.662	59.3%	0.540
gpt-nano	0.700	67.3%	0.648
gpt5-mini	0.589	54.9%	0.427
Avg.	0.651	60.4%	0.540

two rankings. **Ranking Correlation Test.** Task T1 computes the full-sample ranking correlation between $\{h_i\}$ and $\{g_i\}$, reporting Spearman's ρ and the corresponding p-value.

Extreme Category Identification. Task T2 treats the bottom 15% as the low-signal group and the top 15% as the high-signal group, and evaluates the resulting classification performance.

4.4.1 Experimental Setup

Task Sets. We select several commonly used Agent benchmarks as task sets, including SimpleQA, Tau-Bench, and SWE-Bench Verified. These tasks cover code generation, general knowledge question answering, business API invocation, and other scenarios. However, since Skills vary substantially in their intended purposes and are difficult to map precisely to a specific benchmark, the number of matched tasks is limited. We finally obtain SimpleQA ($N = 43$), Tau-Bench ($N = 13$), and SWE-Bench Verified ($N = 9$). Since the latter two contain too few samples, they are not reported in the main text.

Dataset. We continue to use the OpenClaw Skill Hub dataset described above. We first filter the top 3% of samples by average daily downloads ($N = 1823$). Then, we use an LLM to classify all samples according to the declared purpose of each Skill, mapping each Skill to an applicable benchmark or marking it as unmatched.

4.4.2 Experimental Results

This section examines the core research question: can four-dimensional LLM-based textual scores of

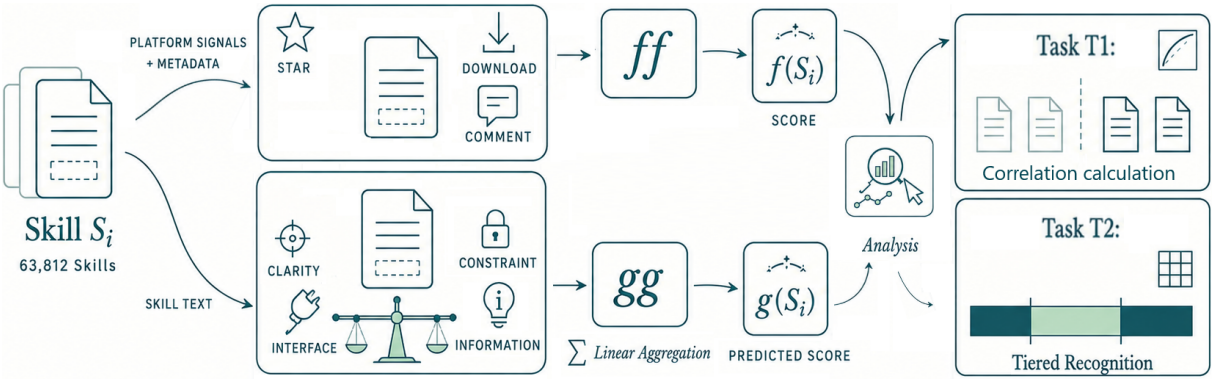

Figure 3: Community signal correlation experiment.

Evaluator	ρ	p	50/50 Auc	50/50 Acc	15/15 Auc	15/15 Acc
AVG Score	+0.543	0.0002	0.826	0.738	0.980	0.857
mimo-pro	+0.434	0.0036	0.723	0.615	0.959	0.857
gpt-nano	+0.498	0.0007	0.822	0.677	0.918	0.857
gpt5-mini	+0.395	0.0087	0.761	0.600	0.980	0.857
mimo-v2.5	+0.299	0.0512	0.749	0.615	0.592	0.429

Table 5: Downstream task relevance results on SimpleQA.

Deleted	$\Delta\rho$	p	sig
Full-4	–	0.0002	***
-clarity	−0.020	0.0003	***
-info_gain	−0.015	0.0003	***
-constraint	−0.038	0.0005	***
-interface	−0.114	0.0041	**

Table 6: Ablation results on SimpleQA using AVG LLM Score as the evaluator.

Skills identify Skills that achieve higher success rates and better performance on downstream tasks?

Main Results. The MCRI-Eval aggregate score shows a significant positive correlation with downstream task performance. Specifically, the Spearman correlation coefficient between AVG Score and downstream performance reaches $\rho = 0.543$, with a significance level of $p = 0.0002$. This suggests that text-based scoring under MCRI-Eval can, to some extent, predict the practical utility of Skills in downstream tasks. In addition, using the average score across four models as the ranking metric performs better than using any single model alone. A possible explanation is that different models exhibit certain distributional biases, while averaging their scores can reduce such biases.

Ablation Results. Table 6 reports the ablation results on SimpleQA. Removing any dimension decreased the correlation, indicating that all four MCRI dimensions provide information for predicting downstream performance.

5 Discussion and Conclusion

This paper centers on Skills as an emerging construct and proposes MCRI, an analytical framework for characterizing their properties, whose utility we demonstrate in two settings: community-level indicators and downstream task performance.

Why there is no "ground truth" of quality. Any study of "quality" faces a fundamental problem: quality itself is definition-dependent, and there is no measurement target that exists independently of such a definition and can be read off directly. Consider a "high-quality" apple: ranking by aroma, sweetness, or crispness yields different orderings, and the weight assigned to each dimension varies across consumers and use cases. The same holds for Skills—a given Skill may exhibit different facets of "quality" under different tasks, models, and user preferences. Consequently, there is no a priori quality value that MCRI can be fit to directly; what we can do is examine whether a given indicator exhibits the expected associations with reasonable external signals.

Positioning the two proxy indicators. It is worth emphasizing that the two indicators studied in this paper are distinct proxies for quality. Community indicators primarily reflect ecosystem-level external feedback and can be regarded as a proxy for popularity; downstream task performance, by contrast, is closer to a Skill's actual utility and therefore closer to the criterion of interest for quality assessment, though it remains subject to task–Skill alignment, sample size, and evaluation methodology.

Overall, we do not claim that MCRI fully defines or exhaustively evaluates Skill quality; rather, it offers a mechanism-driven, operationalizable, and extensible analytical lens.

6 Limitations

This work has several limitations. First, in terms of object coverage, our experiments are primarily based on public Skills from the OpenClaw Skill Hub. Their distribution may differ from those in other Skill ecosystems, such as vendor-provided built-in Skills, closed-source enterprise Skills, or private Skills used within specific organizations. Therefore, the applicability of the MCRI Framework to other ecosystems and usage scenarios requires further examination.

Second, in terms of criterion selection, this paper uses stars, downloads, installs, and downstream task scores as proxy signals for Skill quality. Although these indicators are theoretically related to the practical utility of Skills, they are not equivalent to Skill quality itself. They may also be affected by factors such as release time, platform exposure, user base, task design, and scoring procedures. Therefore, our results should be understood as correlational evidence between the framework and external proxy signals, rather than direct proof of a ground-truth notion of Skill quality.

Third, in terms of downstream validation, Skills serve highly diverse purposes and are difficult to systematically map to existing benchmarks. Although this paper attempts validation using SimpleQA, tau-bench, SWE-bench Verified, and other tasks, the number of matchable samples remains limited, especially for some benchmarks where the sample size is insufficient to support strong conclusions. Therefore, the downstream experiments should be regarded as preliminary evidence. Future work should construct benchmarks specifically designed for Skills, in order to examine the effectiveness and generalizability of MCRI-Eval across a broader range of task types.

References

Anthropic. 2025. Introducing agent skills. Claude Blog. Accessed 2026-05-14.

Cursor. 2026. Subagents, skills, and image generation. Cursor Changelog. Accessed 2026-05-14.

EmergentMind. 2025. Iso/iec 25010 software quality model applied to llm agents.

Google Developers. 2026. Closing the knowledge gap with agent skills. Google Developers Blog. Accessed 2026-05-14.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, and 3 others. 2024. Agentbench: Evaluating llms as agents. ArXiv:2308.03688.

LobeHub. 2026. Agent skills marketplace: Skills marketplace for every agent. LobeHub. Accessed 2026-05-14. The page reported 297,002 skills at the time of access.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2024. Agentboard: An analytical evaluation board of multi-turn LLM agents. In *Advances in Neural Information Processing Systems (NeurIPS)*. ArXiv:2401.13178.

Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. 2025. Evaluation and benchmarking of llm agents: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM.

OpenAI. 2026. Agent skills – codex. OpenAI Developers. Accessed 2026-05-14.

Asaf Yehudai, Lilach Eden, Alan Li, Guy Uziel, Yilun Zhao, Roy Bar-Haim, Arman Cohan, and Michal Shmueli-Scheuer. 2025. Survey on evaluation of llm-based agents. *arXiv preprint arXiv:2503.16416*.

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Neil Zhenqiang Gong, Yue Zhang, and Xing Xie. 2023. Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts. *arXiv preprint arXiv:2306.04528*.