SAAG: Structured Agent Assessment and Grounding

Ritvik Garimella [1], Vedant Khandelwal [1], Anvi Kohli [1], Amit Sheth [1,2]

[1]Artificial Intelligence Institute, University of South Carolina
[2]Indian AI Research Organization
Correspondence: ritvikg@sc.edu

Abstract

Exact-match evaluation of agent-calling obscures qualitatively different failure modes: a model may select the right function yet hallucinate argument values, or satisfy a schema while choosing a agent for the wrong reason. Existing benchmarks collapse these distinctions into a single binary score, leaving practitioners unable to diagnose *where* agent calls fail. We propose **SAAG** a cascaded diagnostic framework that decomposes agent-calling evaluation into three sequential stages: registry conformance, structural completeness, and argument grounding, each producing interpretable stage-specific diagnostics. These diagnostics additionally enable iterative self-repair: on prediction failure, the stage-specific signal guides targeted correction without leaking ground-truth values. We evaluate this framework on a controlled benchmark derived from Glaive's function-calling dataset across registry sizes of 5, 10, and 15 agents using three local sub-4B-parameter models. Structured feedback consistently improves argument precision and reduces value hallucination relative to single-pass inference and uninformative binary feedback, while end-to-end F1 gains are modest and model-dependent. These results suggest that stage-decomposed diagnostic evaluation is a necessary lens for understanding and improving agent-calling reliability across model families and registry scales.

1 Introduction

Function calling has become the operational interface for LLMs to interact with APIs, search systems, and external software. The central evaluation question is no longer whether a model can emit valid JSON, but whether it can choose the *right* tool for the *right* reason. A model may produce syntactically correct calls yet fail in deployment by selecting a tool whose capability does not match the request, or by supplying arguments that are not grounded in the user's actual query. Such failures are silent but costly: the call passes format checks while remaining operationally wrong.

Recent works: API-Bank, ToolLLM, Gorilla, APIBench, and BFCL, has established strong baselines for tool use and API invocation (Li et al., 2023; Qin et al., 2024; Patil et al., 2024, 2025). Yet the dominant evaluation paradigm still centres on *formal correctness*: whether the model emitted a registered function name, whether calls are executable, or whether execution succeeds end-to-end. These metrics do not answer the more critical question: *did the model select the tool because it correctly understood its capabilities?* Existing evaluations are strong at measuring conformance but weaker at isolating *where and why* a tool call fails. Recent work on tool hallucination confirms this concern: Reliability Alignment distinguishes tool selection hallucination from tool usage hallucination (Xu et al., 2025), and ToolBeHonest shows that even frontier models struggle when tool availability and task solvability are systematically varied (Zhang et al., 2024). Valid tool calls are not sufficient evidence of grounded tool understanding.

A second limitation lies in evaluation granularity. A call can fail because the model hallucinated a function name, omitted a required argument, introduced a spurious parameter, or invented an unsupported argument value. Existing benchmarks collapse these qualitatively different failure modes into a single score, leaving practitioners unable to identify whether a model failed at function selection, schema compliance, or argument grounding. This conflation obscures both diagnosis and targeted improvement.

We propose SAAG (**S**tructured **A**gent **A**ssessment and **G**rounding), a cascaded diagnostic framework that decomposes tool calling into three sequential stages: *registry conformance* (is the function name in the registry?), *structural completeness* (do the arguments satisfy the schema?), and *argument grounding* (are argument values

supported by the query?). Each stage produces interpretable, stage-specific metrics that can be applied to any tool-calling system independently of any correction strategy. As one instantiation of these diagnostics, we additionally implement a cascaded correction loop that returns minimal structured feedback on failure without leaking ground truth. This toolkit acts as a sub-module of the larger multi-agent orchestration framework DYNO (Garimella et al., 2026) with evaluations scheme inspired from conceptualization in C3AN (Sheth et al., 2025).

Scope. Our study focuses on *single-tool selection from a fixed registry*: a controlled yet important step, setting the foundation for multi-agent orchestration. We evaluate three sub-4B models across registry sizes of 5, 10, and 15 agents, comparing three feedback regimes: direct inference, binary-feedback iteration, and full SAAG structured feedback. We do not claim to address multi-agent orchestration, dynamic agent discovery, or real agent execution; our goal is to isolate the diagnostic question: *when agent registries become confusable, can stage-decomposed diagnostic metrics expose where and why agent calls fail, and can structured feedback act on those diagnostics to improve reliability?*

Contributions

1. We introduce SAAG's eight intrinsic grounding metrics that act as a diagnostic measure that characterizes agent-calling failures at a granular level.

2. We demonstrate an application of the diagnostics as a cascaded correction loop across three local small model families.

2 Structured Agent Assessment and Grounding

SAAG decomposes tool-calling evaluation into three sequential quality gates: Registry Conformance (§2.1), Structural Completeness (§2.2), and Argument Grounding (§2.3). The cascade is strict: evaluation halts at the first failing stage, ensuring that downstream metrics are computed only on predictions that satisfy upstream obligations. A structured correction ψ_k is returned to the model at each failure, enabling iterative self-repair without leaking ground-truth values. An important consequence of this design is that intrinsic metrics from later stages are *stage-conditional diagnostics*: they characterise the quality of predictions that already passed earlier gates, and should not be interpreted as unconditional population statistics. We return to this point when reporting results.

2.1 Registry Conformance

The first obligation is that the predicted function name $\hat{y}_{\text{name}}$ corresponds to a registered tool. Let $\mathcal{R}$ denote the set of all function names in the system prompt. We define the *Function Name Exact Match* (FNEM) as:

$$FNEM(\hat{y}_{\text{name}}, \mathcal{R}) = \begin{cases} -1 & \text{if } \hat{y}_{\text{name}} = \varnothing \\ 0 & \text{if } \hat{y}_{\text{name}} \notin \mathcal{R} \\ 1 & \text{if } \hat{y}_{\text{name}} \in \mathcal{R} \end{cases} \quad (1)$$

The three cases distinguish complete parse failure (-1), registry mismatch (0), and conformance (1). When FNEM $= 0$, we compute a *Nearest Candidate Score* (NCS) to quantify drift from the closest valid name:

$$\text{NCS}(\hat{y}_{\text{name}}, \mathcal{R}) = \max_{c \in \mathcal{C}} \Big[0.6\, \text{WRatio}(\hat{y}_{\text{name}}, c) + 0.4\, \text{Jaccard}(\hat{y}_{\text{name}}, c) \Big]. \quad (2)$$

where $\mathcal{C} \subseteq \mathcal{R}$ are candidates sharing at least one trigram with $\hat{y}_{\text{name}}$, WRatio is the weighted partial ratio from RapidFuzz (Bachmann, 2025) (normalised to $[0, 1]$), and Jaccard is computed over whitespace-tokenised lowercased strings. The reason to use 0.6 as weightage for WRatio as it better captures approximate string similarity under minor variations in comparison to Jaccard that provides complementary token-level overlap. **What this catches:** hallucinated or misspelled function names. A high NCS with FNEM $= 0$ indicates a recoverable misspelling; a low NCS indicates fabrication. **What this cannot capture:** whether the *correct* registry name actually corresponds to the tool best suited for the query. If FNEM $= 1$, Stage 1 passes; otherwise the correction ψ_1 identifies whether the name was absent or unrecognised and embeds the nearest registry match to guide revision without revealing ground truth. Fix templates are provided in Appendix D.

2.2 Structural Completeness

Given a conformant function name, the second obligation is that the predicted argument set P satisfies

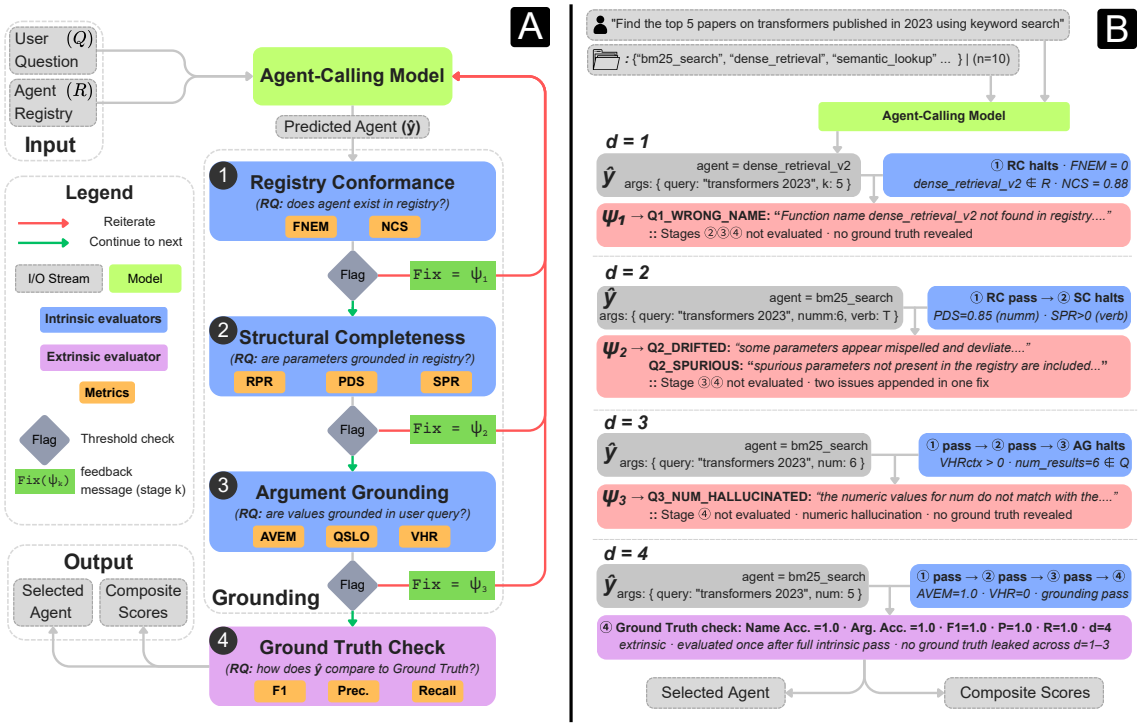

Figure 1: **The SAAG framework.** (A) Overview of the cascaded evaluation pipeline: a tool-calling model's prediction passes sequentially through Registry Conformance (Stage 1), Structural Completeness (Stage 2), and Argument Grounding (Stage 3), each producing interpretable metrics and a targeted correction signal ψ_k on failure. Ground truth (*extrinsic* evaluation) is assessed only after all stages pass. (B) A worked example illustrating four correction iterations ($d = 1$–4): the model first hallucinates a function name (FNEM=0), then produces drifted and spurious parameters, then introduces a numeric hallucination, before arriving at a fully grounded, schema-conformant call at $d = 4$.

the function's schema. Let A_{req} and A_{all} denote the required and admissible parameter sets respectively. We measure structural completeness along three axes.

Required Parameter Recall (RPR) measures coverage of mandatory arguments:

$$\text{RPR} = \frac{|A_{\text{req}} \cap P|}{|A_{\text{req}}|} \tag{3}$$

RPR < 1 is a hard failure: missing required parameters will cause execution to fail regardless of value correctness.

Parameter Drift Score (PDS) captures whether out-of-schema predictions are recoverable misspellings rather than hallucinated arguments. For each parameter $p \in P \setminus A_{\text{all}}$:

$$\text{PDS}(p,\ A_{\text{all}}) = \max_{a \in A_{\text{all}}} \text{WRatio}(p,\ a)\ /\ 100 \tag{4}$$

A parameter is classified as *drifted* if PDS ≥ 0.8 and its nearest match is not already present as an exact prediction, and as *spurious* otherwise. Drifted parameters receive a targeted spelling fix; spurious ones are flagged for removal.

Spurious Parameter Rate (SPR) measures the proportion of predicted arguments with no recoverable schema grounding:

$$\text{SPR} = \frac{\left|\{p \in P \setminus A_{\text{all}} : \text{PDS}(p, A_{\text{all}}) < 0.8\}\right|}{|P|} \tag{5}$$

Stage 2 pass criterion. The stage passes if and only if RPR $= 1$ and SPR $= 0$. Separately, drifted parameters (PDS ≥ 0.8) always trigger a targeted correction regardless of the formal pass/fail outcome, because unresolved drift propagates into downstream grounding failures even when the required parameters are all present. The correction ψ_2 is stratified by failure type: missing required parameters, spurious inclusions, and drifted spellings each produce distinct guidance. **What this catches:** schema violations, missing, misspelled, and fabricated parameters. **What this cannot capture:** whether *correct* parameter names carry appropriate

values.

2.3 Argument Value Grounding

The third obligation is that argument values are faithfully grounded in the user query Q. Unlike the previous stages, grounding is continuous: a model may produce a paraphrase, a near-miss, or an outright hallucination.

Argument Value Exact Match (AVEM) measures the rate at which predicted values appear verbatim in Q:

$$\text{AVEM} = \frac{\left|\{i : \text{ExactMatch}(v_i, Q) = 1\}\right|}{|P|} \quad (6)$$

The check is type-aware: string values are matched case-insensitively at word boundaries; numeric values must match a floating-point token in Q; list values require all elements to satisfy the condition individually. Exact match is the *only* grounding check applied to numeric arguments, there is no partial-credit fallback.

Query-Supported Lexical Overlap (QSLO) measures lexical proximity exclusively for string arguments that fail exact match. Let $\mathcal{S} = \{i : \text{type}(v_i) = \text{string} \wedge \text{ExactMatch}(v_i, Q) = 0\}$. Then:

$$\text{QSLO} = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \text{PartialRatio}(v_i,\, Q) \;/\; 100 \quad (7)$$

where PartialRatio scores the best contiguous alignment of v_i within Q via RapidFuzz (Bachmann, 2025). This is a *lexical* overlap measure, not a semantic similarity score; it captures whether argument values are extractable substrings of the query, not whether they are semantically equivalent. **What this catches:** paraphrased or truncated query spans. **What this cannot capture:** semantically valid values that use entirely different surface forms than the query.

Value Hallucination Rate (VHR) aggregates per-argument hallucination:

$$h(v_i, Q) = \begin{cases} 0, & \text{EM}(v_i, Q) = 1, \\ 1 - \frac{\text{PR}(v_i, Q)}{100}, & \text{str}(v_i), \\ 1, & \text{num}(v_i) \wedge \text{HN}(Q) \\ \text{excluded}, & \text{otherwise.} \end{cases} \quad (8)$$

Here, EM denotes *ExactMatch*, PR denotes *PartialRatio*, and HN denotes *HasNumbers*. We use $\text{str}(\cdot)$ and $\text{num}(\cdot)$ to indicate string and numeric types, respectively.

The final case covers arguments for which no grounding context exists in the query (e.g., a ticker symbol inferred from a company name). These are treated as *assumed* and excluded from the pass/-fail decision, since penalising them would conflate hallucination with legitimate model inference. The pass criterion is defined over the contextual subset $\mathcal{G} = \{i : \text{type}(v_i) = \text{string}\} \cup \{i : \text{type}(v_i) = \text{numeric} \wedge \text{HasNumbers}(Q)\}$:

$$\text{VHR}_{\text{ctx}} = \sum_{i \in \mathcal{G}} h(v_i, Q) = 0 \quad (9)$$

A non-zero VHR_{ctx} produces the correction ψ_3, stratified by failure type: numeric mismatches, hard-drifted strings (PartialRatio < 0.6), and soft-drifted strings each receive distinct correction language.

2.4 Cascaded Correction Loop

The three stages are evaluated sequentially. The cascade halts at the first failing stage k, and its structured fix ψ_k is appended to the conversation as a user turn:

$$\hat{y}^{(t+1)} = \mathcal{M}\Big(H^{(t)} \circ \psi_k\Big) \quad (10)$$

where $H^{(d)}$ is the conversation history at iteration d. The loop repeats up to $D_{\text{max}} = 15$ iterations and terminates early on a full pass. Each fix ψ_k is *templated and deterministic*: given the same failure mode and metric values, the same correction string is produced, ensuring reproducibility. The full set of fix templates is given in Appendix D.

3 Experimental Setup

We evaluate whether cascaded diagnostic feedback improves tool-calling reliability in sub-4B-parameter models when (a) model families change, (b) registry scale increases, and (c) feedback specificity varies.

Dataset construction. We construct a controlled benchmark from the Glaive function-calling dataset (Glaive AI, 2023). From the original corpus, we retain only rows that require exactly one tool call and for which the system prompt can be standardised to expose functions as JSON schema blocks with typed parameters and explicit required fields. Rows with malformed schemas, multi-call sequences, or ambiguous ground truth are excluded.

After filtering, 3,000 queries form the evaluation set. For each query, registry scale is controlled by retaining the ground-truth function and sampling $S - 1$ distractors *uniformly at random* from the remaining function pool to reach $S \in \{5, 10, 15\}$ functions per prompt. Distractors are not curated for semantic confusability; they are random draws, meaning that registry difficulty comes from scale, not from adversarial similarity. The same 3,000 queries are reused across all registry scales so that performance differences reflect registry complexity alone.

Models. We evaluate three models: Llama 3.2:3B (Grattafiori et al., 2024), Ministral 3B (Liu et al., 2026), and Granite 3.1:3B (Mishra et al., 2024). All inferences are run locally via Ollama at temperature $\tau = 0$, with no fine-tuning; every condition is a pure inference-time intervention. This selection spans three distinct families, testing whether SAAG's feedback generalises across architectural and training differences.

Feedback modes. Three conditions are compared (Table 1): **Direct pass**: a single inference pass with no correction loop, providing a zero-feedback baseline of out-of-the-box capability. **Binary feedback**: the correction loop is active, but each stage failure returns a fixed uninformative signal ("No") without diagnostic content. This serves as an *ablation* to isolate the effect of iterative re-prompting from the information content of the feedback; it is not intended as a representative real-world baseline. **Structured feedback (SAAG)**: the full cascaded diagnostic framework from §2, where each failing stage returns a stage-stratified correction ψ_k targeted to the specific failure mode. Direct pass uses a single attempt ($D = 1$). Binary feedback and SAAG both operate under a maximum of $D_{\text{max}} = 15$ iterations, bounding cost fairly across loop-enabled conditions.

Definitions. *Solve rate* is the fraction of the 3,000 queries for which the model produces a prediction that passes all three SAAG stages and matches the ground-truth function name *within the iteration budget*. For direct pass, this is identical to single-pass accuracy; for loop-enabled modes, it captures whether the loop eventually recovers a correct call.

Metric classes. We report two classes of metrics. *Intrinsic metrics* are the eight SAAG diagnostics from §2, aggregated across rows to characterise grounding quality per stage. Because the cascade is strict, intrinsic metrics from later stages are conditioned on passing earlier stages and should be interpreted as stage-conditional diagnostics, not unconditional population statistics. *Extrinsic metrics* (name accuracy, argument precision, recall, F1) are computed against ground truth for all rows and serve as end-to-end validity checks.

Uncertainty. All $\pm$ values in Tables 2–3 are standard deviations across the three registry scales ($S \in \{5, 10, 15\}$), reflecting variability due to registry complexity rather than sampling noise.

Mode	CS	Loop	D_{max}
Direct pass	None	No	1
Binary feedback	Fixed "No"	Yes	15
SAAG	targeted diagnostic	Yes	15

Table 1: Experimental conditions. All three modes are evaluated across three models and three registry scales ($S \in \{5, 10, 15\}$), yielding 27 conditions on 3,000 queries each. Binary feedback is an ablation isolating the effect of iteration from feedback content; it is not a strong external baseline. CS - Correction Signal

4 Results

We report intrinsic grounding diagnostics (Table 3) and extrinsic correctness metrics (Table 2), both averaged across registry scales $S \in \{5, 10, 15\}$. Intrinsic metrics are the primary lens; extrinsic results serve as an end-to-end validity check. Per-scale breakdowns are in Appendix E.

Argument grounding is the dominant failure mode. Across all models and conditions, name accuracy is near ceiling while argument accuracy lags substantially. This persistent gap confirms argument grounding as the primary bottleneck. Conflating these failure modes into a single outcome metric, as standard exact-match benchmarks do, obscures where models most need improvement.

Precision improves consistently; F1 gains are mixed. In agent-call grounding, a hallucinated argument value causes silent downstream failures, whereas a missing argument is at least detectable. Precision is therefore the more meaningful signal in this setting. SAAG improves argument precision for all three models and solve rate improves substantially across all models (Table 2), with the

largest gain for LLaMA ($0.615 \rightarrow 0.802$). End-to-end F1, however, tells a more nuanced story: SAAG improves F1 for LLaMA ($0.527 \rightarrow 0.533$) but slightly decreases it for Ministral and Granite, driven by recall loss. This is consistent with an overcorrection effect in reasoning models: both Ministral 3 and Granite 4 are reasoning-oriented models, and iterative feedback loops can cause them to revise initially correct predictions, a behaviour documented for Ministral 3 (Liu et al., 2026) and similarly observed for Granite 4.

Structured feedback consistently reduces hallucination. The most reliable gain from structured feedback (SAAG) is reduced argument value hallucination. VHR decreases under SAAG for every model: from 0.336 to 0.311 for LLaMA, from 0.277 to 0.166 for Ministral (a 40% reduction), and from 0.278 to 0.259 for Granite. AVEM improves correspondingly, with the largest gain for Ministral ($0.658 \rightarrow 0.786$). Binary feedback, by contrast, occasionally *increases* VHR (Granite: $0.278 \rightarrow 0.288$), suggesting that for reasoning enabled models, correction signals introduce spurious drift rather than fix grounding errors.

SAAG converges faster than binary feedback. Structured feedback converges within 2-3 iterations across all registry scales (Figure 4), reaching a higher ceiling than binary feedback at every depth i.e., consuming less overall time. This confirms that *content* based diagnostic correction drives grounding recovery quicker than pure iterative looping.

5 Related Work

Tool-calling evaluation and benchmarks. Recent benchmarks such as API-Bank (Li et al., 2023), ToolLLM (Qin et al., 2024), Gorilla (Patil et al., 2024), and BFCL (Patil et al., 2025) have established standardized protocols for evaluating tool use in large language models. While effective for measuring overall performance, these benchmarks predominantly focus on format correctness or end-to-end execution success, often conflating distinct failure modes into a single aggregate metric. As a result, they provide limited diagnostic insight into where and why failures occur. ToolBeHonest (Zhang et al., 2024) advances this direction by introducing a structured, multi-level diagnostic framework that decomposes tool use into solvability detection, solution planning, and missing-tool analysis, while also evaluating robustness under hallucination-inducing scenarios such as missing or constrained tools. However, its scope is limited to evaluation. In contrast, SAAG extends this paradigm beyond diagnosis by not only identifying stage-specific failure modes but also leveraging them to drive an inference-time, iterative self-repair process that actively corrects tool-calling errors.

Tool hallucination. Tool hallucination has emerged as a critical challenge in reliable tool use. Reliability Alignment (Xu et al., 2025) distinguishes between *tool selection hallucination* and *tool usage hallucination*, highlighting the need for stage-aware analysis of model behavior. Prior approaches address these issues through additional training signals, such as process-level reward models (Qian et al., 2025) or self-reflection and feedback mechanisms (Masterman et al., 2024). In contrast, SAAG operates entirely at inference time, leveraging deterministic and templated feedback without requiring auxiliary models or retraining, thereby offering a lightweight yet effective alternative.

Iterative correction. Frameworks such as ReAct (Yao et al.) and related reasoning-and-acting paradigms improve tool use through iterative interaction, enabling models to refine their outputs over multiple steps. However, the feedback signals in these approaches are typically coarse and undifferentiated. In this work, we explicitly disentangle the role of iteration from the quality of feedback through a binary feedback ablation, and ground our correction signals in deterministic, rule-based templates derived from intrinsic stage metrics rather than a secondary LLM functioning as judge.

6 Discussion

Summary. SAAG's stage-decomposed metrics reveal what aggregate benchmarks cannot: argument grounding is the dominant failure mode, overcorrection is a distinct and measurable phenomenon, and precision gains do not always propagate to F1. These findings hold independently of any correction strategy with the intrinsic diagnostics characterizing model behaviour whether or not the correction loop is applied. Where the loop (diagnostic feedback) is applied, the strongest signal is that *telling a model what failed* is substantially more effective than *telling it that it failed*, a finding that holds across all three model families and registry complexities. Furthermore, the

Metric	LLaMA 3.2			Ministral			Granite		
	Direct	Binary	SAAG	Direct	Binary	SAAG	Direct	Binary	SAAG
Precision	0.524±.037	0.525±.035	0.540±.033	0.705±.013	0.648±.013	0.661±.013	0.747±.015	0.667±.013	0.717±.049
Recall	0.537±.037	0.524±.033	0.534±.031	0.710±.013	0.648±.013	0.654±.013	0.745±.015	0.663±.013	0.690±.032
F1	0.527±.370	0.521±.033	0.5330±.032	0.705±.013	0.646±.013	0.654±.013	0.744±.015	0.664±.013	0.698±.038
Solve Rate	0.615±.016	0.768±.014	0.802±.014	0.726±.008	0.829±.000	0.825±.005	0.694±.020	0.816±.002	0.771±.048

Table 2: Extrinsic metrics averaged across registry scales $S \in \{5, 10, 15\}$; $\pm$ values are standard deviations across scales. Higher is better for all metrics (↑). SAAG improves precision for all models but F1 gains are mixed: LLaMA improves, while Ministral and Granite show slight decreases due to recall loss from overcorrection. Per-scale breakdowns in Appendix E.

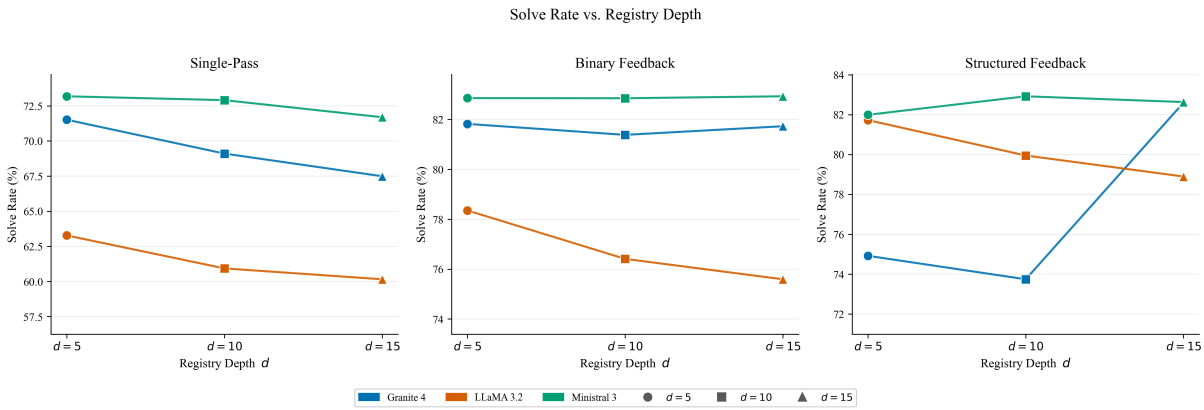

Figure 2: Solve rate (%) across registry depths (d = 5, 10, 15) for three models under Single-Pass, Binary Feedback, and Structured Feedback conditions. Ministral 3 consistently leads across all settings, while LLaMA 3.2 degrades most sharply with increasing depth. Structured Feedback provides the greatest gains and most stable performance as registry size scales.

Stage	Metric	LLaMA 3.2			Ministral			Granite		
		Direct	Binary	SAAG	Direct	Binary	SAAG	Direct	Binary	SAAG
RC	FNEM ↑	0.999±.000	0.9977±.001	1.0000±.000	1.000±.000	1.000±.000	0.999±.001	1.000±.000	0.998±.002	0.998±.002
SC	RPR ↑	0.968±.012	0.965±.004	0.963±.017	0.984±.005	0.937±.002	0.886±.008	0.958±.021	0.976±.012	0.926±.035
	SPR ↓	0.043±.028	0.015±.006	0.028±.039	0.008±.006	0.001±.001	0.002±.001	0.031±.015	0.014±.008	0.020±.007
AG	AVEM ↑	0.584±.036	0.613±.010	0.617±.053	0.658±.004	0.717±.003	0.786±.009	0.648±.003	0.645±.003	0.688±.030
	QSLO ↑	0.339±.019	0.347±.006	0.343 ±.012	0.268±.010	0.331±.011	0.361±.035	0.318±.008	0.296±.006	0.291±.005
	VHR ↓	0.336±.028	0.314±.006	0.311 ±.041	0.277±.006	0.213±.004	0.166±.008	0.278±.005	0.288±.002	0.259±.021

Table 3: Intrinsic grounding diagnostics averaged across registry scales $S \in \{5, 10, 15\}$; $\pm$ values are standard deviations across scales. These are *stage-conditional* metrics: later-stage values are computed only on predictions that passed earlier stages. NCS and PDS are omitted from the main table as they are near-constant across conditions; full values including per-scale breakdowns are in Appendix E.

stage-conditional design further enables actionable diagnosis. For example: under binary feedback, Granite's RPR improves while VHR worsens simultaneously; an indication that a single aggregate metric would simply average out and miss. Per-stage decomposition surfaces such contradictions directly and points to where intervention is needed.

Limitations. Several limitations bound the scope of our findings. Firs, our evaluation is restricted to single-agent selection from a fixed registry; multi-agent sequences, dynamic registries, and agent chaining introduce failure modes that SAAG does not currently address. Second, distractors are sampled uniformly at random rather than for semantic confusability, so our registry difficulty is a lower bound on what real deployments face. Third, the argument grounding stage relies on lexical overlap (PartialRatio) rather than true semantic similarity; values that are semantically correct but lexically distant from the query will be incorrectly flagged. Fourth, the correction loop can overcorrect valid

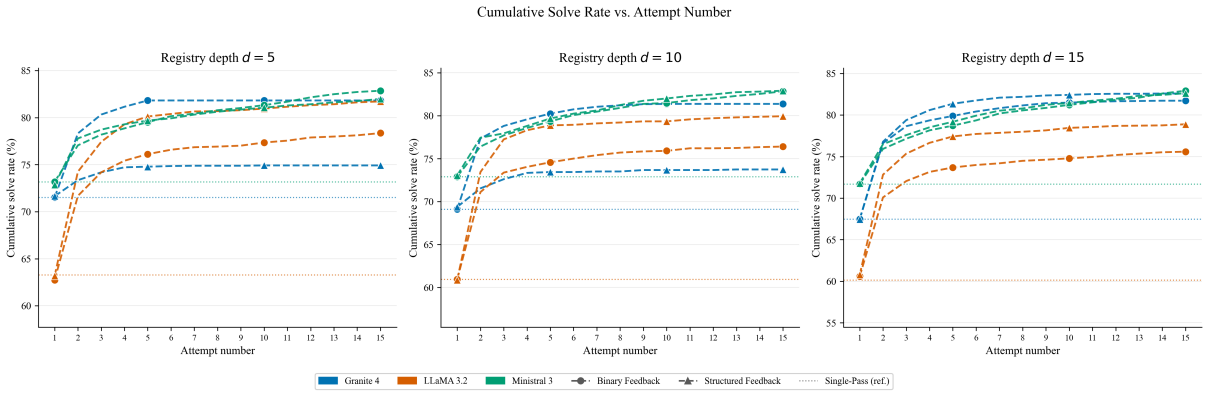

Figure 3: Cumulative solve rate (%) over up to 15 attempts across registry depths (d = 5, 10, 15) for Granite 4, LLaMA 3.2, and Ministral 3 under Binary and Structured Feedback, with Single-Pass as the dotted baseline. All models converge within the first 2–3 attempts, with Structured Feedback improving more with attempts than Binary Feedback across all depths. Lower registry depths converge faster compared to higher depths indicating that more complexity in registry requires more attempts to solve.

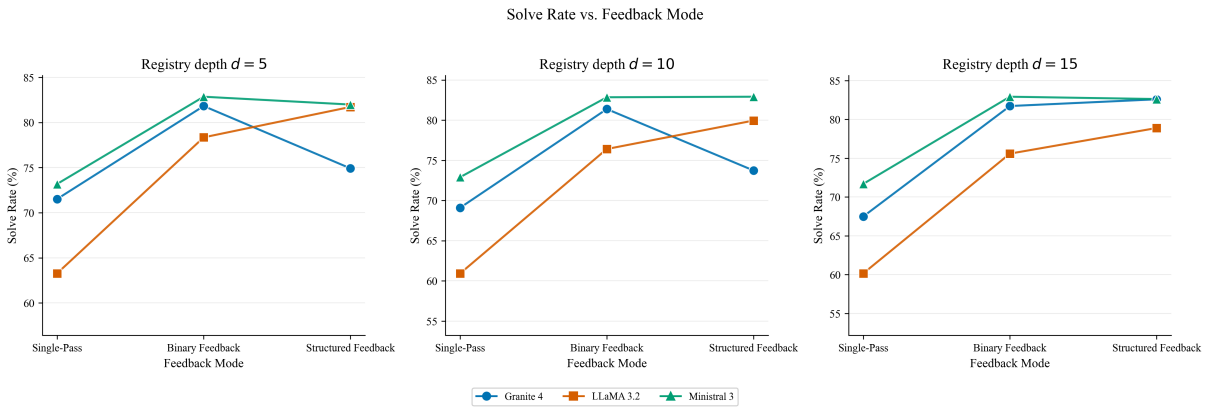

Figure 4: Solve rate (%) across feedback modes (Single-Pass, Binary, Structured) at registry depths d = 5, 10, 15 for Granite 4, LLaMA 3.2, and Ministral 3. All models improve substantially from Single-Pass to Binary Feedback, with LLaMA 3.2 showing the largest gains. Ministral 3 remains stable from Binary to Structured Feedback, while Granite 4 drops notably at d = 5, suggesting it is more sensitive to feedback at shallower depths due to reasoning enabled.

predictions, particularly for agent names, suggesting that tighter thresholds or confidence-gated correction are needed. Also, these overcorrections occur at a higher frequency for reasoning models such as Ministral 3 and Granite 4. Finally, we evaluate only sub-4B-parameter models at inference time with no fine-tuning; whether the diagnostic decomposition transfers to larger models or training-time objectives remains open.

7 Conclusion

We presented SAAG, a diagnostic framework that decomposes agent-calling evaluation into three sequential stages: registry conformance, structural completeness, and argument grounding, with each having interpretable, stage-specific metrics. These metrics constitute the primary contribution: *they expose where agent calls fail i.e., at function selection, schema compliance, or argument grounding and at a granularity* that existing exact-match benchmarks do not provide.Evaluated across three sub-4B model families and three registry scales, the cascaded correction loop consistently improves argument precision and reduces value hallucination, while also surfacing a measurable overcorrection effect in reasoning models, name selection and recall,a limitation that motivates confidence-gated correction as a direction for future work. The central finding is that telling a model *what* failed is substantially more effective than telling it *that* it failed.

8 Acknowledgment

This work was supported in part by NSF grant #2119654, "RII Track 2 FEC: Enabling Factory to Factory (F2F) Networking for Future Manufacturing".

References

Max Bachmann. 2025. rapidfuzz/rapidfuzz: Release 3.13.0.

Ritvik Garimella, Chathurangi Shyalika, Renjith Prasad Kaippilly Mana, and Amit Sheth. 2026. Dyno: Dynamic neurosymbolic orchestrator for multi-agent systems. In *LLM-based Multi-Agent Systems: Towards Responsible, Reliable, and Scalable Agentic Systems*.

Glaive AI. 2023. glaive-function-calling-v2. https://huggingface.co/datasets/glaiveai/glaive-function-calling-v2.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. API-bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116, Singapore. Association for Computational Linguistics.

Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, and 1 others. 2026. Ministral 3. *arXiv preprint arXiv:2601.08584*.

Tula Masterman, Sandi Besen, Mason Sawtell, and Alex Chao. 2024. The landscape of emerging ai agent architectures for reasoning, planning, and tool calling: A survey. *Preprint*, arXiv:2404.11584.

Mayank Mishra, Matt Stallone, Gaoyuan Zhang, Yikang Shen, Aditya Prasad, Adriana Meza Soria, Michele Merler, Parameswaran Selvam, Saptha Surendran, Shivdeep Singh, and 1 others. 2024. Granite code models: A family of open foundation models for code intelligence. *arXiv preprint arXiv:2405.04324*.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 48371–48392. PMLR.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems*, volume 37.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. Toolrl: Reward is all tool learning needs. *Preprint*, arXiv:2504.13958.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations*.

Amit P Sheth, Kaushik Roy, Revathy Venkataramanan, Venkatesan Nadimuthu, and Chathurangi Shyalika. 2025. Composite ai with custom, compact, neurosymbolic models: The emergent enterprise artificial intelligence paradigm. *IEEE Internet Computing*, 29(2):37–49.

Hongshen Xu, Zichen Zhu, Lei Pan, Zihan Wang, Su Zhu, Da Ma, Ruisheng Cao, Lu Chen, and Kai Yu. 2025. Reducing tool hallucination via reliability alignment. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 69992–70006. PMLR.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Yuxiang Zhang, Jing Chen, Junjie Wang, Yaxin Liu, Cheng Yang, Chufan Shi, Xinyu Zhu, Zihao Lin, Hanwen Wan, Yujiu Yang, Tetsuya Sakai, Tian Feng, and Hayato Yamana. 2024. Toolbehonest: A multi-level hallucination diagnostic benchmark for tool-augmented large language models. *Preprint*, arXiv:2406.20015.

A Reproducibility Statement

All experiments use the following exact model identifiers via Ollama: `llama3.2:3b`, `ministral:3b`, and `granite4:3b`. Inference is performed at temperature $\tau = 0$ with no system-level sampling modifications. The correction loop uses a maximum of $T_{\max} = 15$ iterations. All fuzzy matching thresholds are fixed: the parameter drift threshold is $\gamma_{\text{pds}} = 0.8$; the hard-drift boundary for Stage 3 correction is PartialRatio < 0.6. All fix templates (ψ_1, ψ_2, ψ_3) are deterministic and templated: given the same failure mode and metric values, the same correction string is produced. The Glaive function-calling dataset was filtered to retain only single-call rows with well-formed JSON schemas; distractors are sampled uniformly at random from the function pool. Experiments were run on a single NVIDIA A100 GPU. Detailed prompt templates and the full dataset construction pipeline will be provided in the supplementary code repository upon acceptance.

B Ethics Statement

This work proposes a diagnostic framework for evaluating and repairing tool calls at inference time. We note several ethical considerations. First, while structured feedback reduces argument hallucination, it can also induce overcorrection—particularly in function name selection—and should not be deployed without monitoring for this failure mode. Second, our evaluation is restricted to single-tool selection from a fixed registry using synthetic queries; we do not evaluate real tool execution, multi-agent deployment, or user-facing applications. Claims about deployment readiness should not be inferred from these results. Third, the lexical grounding metric (QSLO) captures surface overlap, not semantic correctness; values that are semantically valid but lexically distant from the query may be incorrectly penalised. Users of this framework should be aware of this limitation when interpreting grounding scores.

C Dataset Construction

We construct a controlled evaluation benchmark from the Glaive function-calling dataset (Glaive AI, 2023). The original corpus is restructured along three axes: (i) we retain only rows that require a tool call; (ii) we standardise the system prompt to expose each function as a standalone JSON schema block with typed parameters and explicit `required` fields; and (iii) we stratify the registry scale to $S \in \{5, 10, 15\}$ functions per row by retaining the ground-truth relevant function and sampling $S-1$ distractors uniformly from the broader function pool. The same 3,000 queries are used across all three registry scales, so any performance difference across S reflects registry complexity alone rather than query distribution shift.

Unlike standard function-calling benchmarks that expose only the relevant tool, our construction forces the model to perform genuine registry selection: the correct function must be identified from among $S-1$ structurally similar distractors whose parameter types and description phrasing overlap with the query. As S grows, the risk of name confusability and argument drift increases. This directly probes the failure modes captured by our SAAG metrics.

Figure 5 shows a representative system prompt at S=5. At S=10 and S=15, five and ten further distractor blocks are appended; the query and ground truth are unchanged. The only place ground truth is used is during the extrinsic metric calculation.

System prompt with agent scale depth as 5

```
SYSTEM: You are a helpful assistant with access to the
following functions. Use them if required -
{"name": "get_definition", "description": "Get the
definition of a word", "parameters": {"type":
"object", "properties": {"word": {"type": "string",
"description": "The word to get the definition of"}},
"required": ["word"]}}
{"name": "generate_qr_code", "description": "Generate
a QR code for a given text", "parameters": {"type":
"object", "properties": {"text": {"type": "string",
"description": "The text to encode"}, "size": {"type":
"integer", "description": "QR code size in pixels"}},
"required": ["text"]}}
{"name": "convert_temperature", "description":
"Convert temperature between units", "parameters":
{"type": "object", "properties": {"temperature":
{"type": "number"}, "from_unit": {"type": "string"},
"to_unit": {"type": "string"}}, "required":
["temperature", "from_unit", "to_unit"]}}
{"name": "calculate_mortgage", "description":
"Calculate monthly mortgage payments", "parameters":
{"type": "object", "properties": {"loan_amount":
{"type": "number"}, "interest_rate": {"type":
"number"}, "loan_term": {"type": "integer"}},
"required": ["loan_amount", "interest_rate",
"loan_term"]}}
{"name": "calculate_tip", "description":
"Calculate the tip amount based on the bill",
"parameters": {"type": "object", "properties":
{"bill_amount": {"type": "number"}, "tip_percentage":
{"type": "number"}}, "required": ["bill_amount",
"tip_percentage"]}}
USER: What does the word 'serendipity' mean?
GROUND TRUTH: {"name": "get_definition", "arguments":
{"word": "serendipity"}}
```

Figure 5: System prompt at S=5 for a representative query. The registry contains one relevant function (`get_definition`) and four distractors. At S=10 and S=15, five and ten further distractor blocks are appended; the query and ground truth remain identical.

D Structured Correction Messages

Table 4 shows representative fix messages ψ_k for each failure mode across the three stages. All fixes are constructed programmatically from the evaluation output; no human authoring is involved. Crucially, no fix reveals the ground-truth function name, parameter set, or argument values, ensuring that corrections are grounded solely in the registry schema and the model's own prediction.

E Per-Scale Results

Per-scale breakdowns of intrinsic grounding metrics and extrinsic ground-truth metrics for all three models across registry scales $S \in \{5, 10, 15\}$; summary tables averaged across scales are reported in the main paper (intrinsic: Table 3; extrinsic: Table 2). Tables 5–16 present full per-scale breakdowns, with each table reporting intrinsic grounding metrics (top block) and extrinsic correctness metrics (bottom block) as mean $\pm$ std over all rows at that registry scale and the respective solve rate.

Stage	Failure Mode	Representative Fix Message (Fix Prompts)
Q1	No name found	*No function name found. Please return only the agent invocation in the exact JSON format specified in the registry.*
Q1	Wrong name	*Function name "{predicted}" not found in registry. Please check the system prompt for the correct function names and respond with only the agent invocation in the exact JSON format specified.*
Q2	Missing required	*Key parameters of the agent from the registry are missing. Please ensure the invocation includes all parameters specified for that agent in the system prompt.*
Q2	Spurious params	*Spurious parameters not present in the registry are included. Please include only the parameters specified for that agent.*
Q2	Drifted params	*Some parameters appear misspelled and deviate slightly from the registry. Please ensure all parameter names exactly match the schema in the system prompt.*
Q3	Numeric mismatch	*The numeric values for {params} do not match what the user stated. Use the exact numbers from the user's request.*
Q3	Hard value drift	*The values for {params} have no grounding in the user's request. Do not invent values. Use only what the user explicitly stated, or omit the argument.*
Q3	Soft value drift	*The values for {params} are loosely related but not exact. Use the precise wording or value from the user's request.*

Table 4: Representative structured correction messages ψ_k per failure mode. Bracketed tokens ({predicted}, {params}) are instantiated programmatically from the evaluation output.

Group	Metric	Direct	Binary	SAAG
INTRINSIC	FNEM ↑	$1.000_{\pm 0.000}$	$0.999_{\pm 0.026}$	$1.000_{\pm 0.000}$
	NCS ↑	$1.000_{\pm 0.000}$	$0.999_{\pm 0.026}$	$1.000_{\pm 0.000}$
	RPR ↑	$0.979_{\pm 0.136}$	$0.970_{\pm 0.147}$	$0.976_{\pm 0.131}$
	PDS ↑	$0.006_{\pm 0.075}$	$0.002_{\pm 0.037}$	$0.001_{\pm 0.033}$
	SPR ↓	$0.048_{\pm 0.208}$	$0.008_{\pm 0.080}$	$0.002_{\pm 0.039}$
	AVEM ↑	$0.567_{\pm 0.438}$	$0.620_{\pm 0.446}$	$0.653_{\pm 0.441}$
	QSLO ↑	$0.325_{\pm 0.233}$	$0.340_{\pm 0.239}$	$0.334_{\pm 0.242}$
	VHR ↓	$0.351_{\pm 0.381}$	$0.312_{\pm 0.401}$	$0.288_{\pm 0.396}$
EXTRINSIC	Precision ↑	$0.561_{\pm 0.451}$	$0.556_{\pm 0.466}$	$0.569_{\pm 0.465}$
	Recall ↑	$0.573_{\pm 0.455}$	$0.552_{\pm 0.465}$	$0.561_{\pm 0.464}$
	F1 ↑	$0.564_{\pm 0.450}$	$0.550_{\pm 0.462}$	$0.561_{\pm 0.461}$
STATS	Solve Rate ↑	0.6328	0.7835	0.8172

Table 5: LLaMA 3.2 at registry scale $S = 5$. The top panel reports intrinsic grounding metrics, while the bottom panel shows extrinsic correctness metrics across feedback conditions. Values are reported as mean $\pm$ std over all rows at this scale; the standard deviation reflects the dispersion of row-level scores within the $[0, 1]$ range. Arrows indicate directionality: higher is better (↑), lower is better (↓).

Group	Metric	Direct	Binary	SAAG
INTRINSIC	FNEM ↑	$0.999_{\pm 0.037}$	$0.997_{\pm 0.082}$	$1.000_{\pm 0.000}$
	NCS ↑	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$
	RPR ↑	$0.955_{\pm 0.198}$	$0.963_{\pm 0.164}$	$0.970_{\pm 0.143}$
	PDS ↑	$0.010_{\pm 0.097}$	$0.005_{\pm 0.066}$	$0.005_{\pm 0.066}$
	SPR ↓	$0.068_{\pm 0.240}$	$0.017_{\pm 0.118}$	$0.009_{\pm 0.090}$
	AVEM ↑	$0.560_{\pm 0.439}$	$0.618_{\pm 0.443}$	$0.646_{\pm 0.437}$
	QSLO ↑	$0.331_{\pm 0.225}$	$0.351_{\pm 0.238}$	$0.358_{\pm 0.241}$
	VHR ↓	$0.354_{\pm 0.380}$	$0.310_{\pm 0.393}$	$0.287_{\pm 0.388}$
EXTRINSIC	Precision ↑	$0.524_{\pm 0.452}$	$0.533_{\pm 0.465}$	$0.547_{\pm 0.466}$
	Recall ↑	$0.538_{\pm 0.457}$	$0.532_{\pm 0.465}$	$0.541_{\pm 0.464}$
	F1 ↑	$0.528_{\pm 0.452}$	$0.528_{\pm 0.461}$	$0.540_{\pm 0.462}$
STATS	Solve Rate ↑	0.6093	0.7641	0.7995

Table 6: LLaMA 3.2 at registry scale $S = 10$. The top panel reports intrinsic grounding metrics, while the bottom panel shows extrinsic correctness metrics across feedback conditions. Values are reported as mean ± std over all rows at this scale; the standard deviation reflects the dispersion of row-level scores within the $[0, 1]$ range. Arrows indicate directionality: higher is better (↑), lower is better (↓).

Group	Metric	Direct	Binary	SAAG
INTRINSIC	FNEM ↑	$1.000_{\pm 0.000}$	$0.997_{\pm 0.082}$	$1.000_{\pm 0.000}$
	NCS ↑	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$
	RPR ↑	$0.969_{\pm 0.149}$	$0.962_{\pm 0.165}$	$0.944_{\pm 0.217}$
	PDS ↑	$0.008_{\pm 0.087}$	$0.009_{\pm 0.090}$	$0.019_{\pm 0.131}$
	SPR ↓	$0.013_{\pm 0.103}$	$0.019_{\pm 0.123}$	$0.073_{\pm 0.246}$
	AVEM ↑	$0.625_{\pm 0.440}$	$0.601_{\pm 0.444}$	$0.552_{\pm 0.437}$
	QSLO ↑	$0.361_{\pm 0.240}$	$0.350_{\pm 0.231}$	$0.336_{\pm 0.232}$
	VHR ↓	$0.303_{\pm 0.392}$	$0.321_{\pm 0.393}$	$0.359_{\pm 0.379}$
EXTRINSIC	Precision ↑	$0.488_{\pm 0.451}$	$0.487_{\pm 0.464}$	$0.503_{\pm 0.465}$
	Recall ↑	$0.499_{\pm 0.456}$	$0.488_{\pm 0.465}$	$0.500_{\pm 0.465}$
	F1 ↑	$0.490_{\pm 0.450}$	$0.485_{\pm 0.461}$	$0.498_{\pm 0.461}$
STATS	Solve Rate ↑	0.6015	0.7559	0.7889

Table 7: LLaMA 3.2 at registry scale $S = 15$. The top panel reports intrinsic grounding metrics, while the bottom panel shows extrinsic correctness metrics across feedback conditions. Values are reported as mean ± std over all rows at this scale; the standard deviation reflects the dispersion of row-level scores within the $[0, 1]$ range. Arrows indicate directionality: higher is better (↑), lower is better (↓).

Group	Metric	Direct	Binary	SAAG
INTRINSIC	FNEM ↑	0.9997 ± 0.0006	0.9977 ± 0.0012	1.0000 ± 0.0000
	NCS ↑	1.0000 ± 0.0000	0.9997 ± 0.0006	1.0000 ± 0.0000
	RPR ↑	0.9677 ± 0.0121	0.9650 ± 0.0044	0.9633 ± 0.0173
	PDS ↑	0.0080 ± 0.0020	0.0053 ± 0.0035	0.0083 ± 0.0095
	SPR ↓	0.0430 ± 0.0283	0.0147 ± 0.0059	0.0280 ± 0.0392
	AVEM ↑	0.5840 ± 0.0359	0.6130 ± 0.0104	0.6170 ± 0.0525
	QSLO ↑	0.3390 ± 0.0187	0.3470 ± 0.0061	0.3427 ± 0.0122
	VHR ↓	0.3360 ± 0.0284	0.3143 ± 0.0059	0.3113 ± 0.0406
EXTRINSIC	Precision ↑	0.5243 ± 0.0366	0.5253 ± 0.0349	0.5397 ± 0.0330
	Recall ↑	0.5367 ± 0.0370	0.5240 ± 0.0333	0.5340 ± 0.0313
	F1 ↑	0.5273 ± 0.0370	0.5210 ± 0.0326	0.5330 ± 0.0322
STATS	Solve Rate ↑	0.6145 ± 0.0160	0.7678 ± 0.0140	0.8019 ± 0.0145

Table 8: LLaMA 3.2 aggregated across registry scales $S = 5, 10, 15$. Values are reported as mean ± sample standard deviation over the three scale-level results.

Group	Metric	Direct	Binary	SAAG
INTRINSIC	FNEM ↑	1.000$_{\pm 0.000}$	1.000$_{\pm 0.000}$	0.999$_{\pm 0.041}$
	NCS ↑	1.000$_{\pm 0.000}$	1.000$_{\pm 0.000}$	1.000$_{\pm 0.011}$
	RPR ↑	0.987$_{\pm 0.112}$	0.934$_{\pm 0.248}$	0.877$_{\pm 0.321}$
	PDS ↑	0.000$_{\pm 0.016}$	0.000$_{\pm 0.016}$	0.000$_{\pm 0.016}$
	SPR ↓	0.003$_{\pm 0.049}$	0.000$_{\pm 0.013}$	0.001$_{\pm 0.032}$
	AVEM ↑	0.662$_{\pm 0.433}$	0.717$_{\pm 0.425}$	0.791$_{\pm 0.377}$
	QSLO ↑	0.279$_{\pm 0.264}$	0.343$_{\pm 0.240}$	0.369$_{\pm 0.323}$
	VHR ↓	0.271$_{\pm 0.388}$	0.215$_{\pm 0.375}$	0.164$_{\pm 0.345}$
EXTRINSIC	Precision ↑	0.718$_{\pm 0.427}$	0.660$_{\pm 0.456}$	0.674$_{\pm 0.448}$
	Recall ↑	0.723$_{\pm 0.428}$	0.660$_{\pm 0.456}$	0.668$_{\pm 0.447}$
	F1 ↑	0.718$_{\pm 0.426}$	0.658$_{\pm 0.454}$	0.668$_{\pm 0.446}$
STATS	Solve Rate ↑	0.7318$_{\pm 0.000}$	0.8286$_{\pm 0.000}$	0.8199$_{\pm 0.000}$

Table 9: Ministral 3 at registry scale $S = 5$. The top panel reports intrinsic grounding metrics, while the bottom panel shows extrinsic correctness metrics across feedback conditions. Values are reported as mean $\pm$ std over all rows at this scale; the standard deviation reflects the dispersion of row-level scores within the $[0, 1]$ range. Arrows indicate directionality: higher is better (↑), lower is better (↓).

Group	Metric	Direct	Binary	SAAG
INTRINSIC	FNEM ↑	1.000$_{\pm 0.000}$	1.000$_{\pm 0.000}$	1.000$_{\pm 0.000}$
	NCS ↑	1.000$_{\pm 0.000}$	1.000$_{\pm 0.000}$	1.000$_{\pm 0.000}$
	RPR ↑	0.986$_{\pm 0.113}$	0.938$_{\pm 0.240}$	0.887$_{\pm 0.309}$
	PDS ↑	0.001$_{\pm 0.023}$	0.002$_{\pm 0.037}$	0.001$_{\pm 0.033}$
	SPR ↓	0.006$_{\pm 0.074}$	0.002$_{\pm 0.040}$	0.002$_{\pm 0.039}$
	AVEM ↑	0.658$_{\pm 0.435}$	0.720$_{\pm 0.421}$	0.792$_{\pm 0.376}$
	QSLO ↑	0.266$_{\pm 0.275}$	0.326$_{\pm 0.253}$	0.392$_{\pm 0.330}$
	VHR ↓	0.277$_{\pm 0.394}$	0.209$_{\pm 0.369}$	0.159$_{\pm 0.338}$
EXTRINSIC	Precision ↑	0.705$_{\pm 0.434}$	0.650$_{\pm 0.461}$	0.659$_{\pm 0.455}$
	Recall ↑	0.709$_{\pm 0.435}$	0.649$_{\pm 0.460}$	0.653$_{\pm 0.454}$
	F1 ↑	0.705$_{\pm 0.433}$	0.648$_{\pm 0.459}$	0.653$_{\pm 0.452}$
STATS	Solve Rate ↑	0.7291$_{\pm 0.000}$	0.8285$_{\pm 0.000}$	0.8292$_{\pm 0.000}$

Table 10: Ministral 3 at registry scale $S = 10$. The top panel reports intrinsic grounding metrics, while the bottom panel shows extrinsic correctness metrics across feedback conditions. Values are reported as mean $\pm$ std over all rows at this scale; the standard deviation reflects the dispersion of row-level scores within the $[0, 1]$ range. Arrows indicate directionality: higher is better (↑), lower is better (↓).

Group	Metric	Direct	Binary	SAAG
INTRINSIC	FNEM ↑	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$
	NCS ↑	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$
	RPR ↑	$0.978_{\pm 0.142}$	$0.938_{\pm 0.238}$	$0.893_{\pm 0.302}$
	PDS ↑	$0.003_{\pm 0.055}$	$0.002_{\pm 0.047}$	$0.002_{\pm 0.038}$
	SPR ↓	$0.014_{\pm 0.112}$	$0.002_{\pm 0.037}$	$0.003_{\pm 0.057}$
	AVEM ↑	$0.654_{\pm 0.437}$	$0.714_{\pm 0.425}$	$0.776_{\pm 0.389}$
	QSLO ↑	$0.259_{\pm 0.264}$	$0.323_{\pm 0.249}$	$0.323_{\pm 0.336}$
	VHR ↓	$0.283_{\pm 0.397}$	$0.216_{\pm 0.374}$	$0.174_{\pm 0.352}$
EXTRINSIC	Precision ↑	$0.693_{\pm 0.437}$	$0.635_{\pm 0.463}$	$0.649_{\pm 0.458}$
	Recall ↑	$0.697_{\pm 0.438}$	$0.634_{\pm 0.464}$	$0.642_{\pm 0.457}$
	F1 ↑	$0.692_{\pm 0.436}$	$0.632_{\pm 0.461}$	$0.642_{\pm 0.455}$
STATS	Solve Rate ↑	$0.7169_{\pm 0.000}$	$0.8293_{\pm 0.000}$	$0.8263_{\pm 0.000}$

Table 11: Ministral 3 at registry scale $S = 15$. The top panel reports intrinsic grounding metrics, while the bottom panel shows extrinsic correctness metrics across feedback conditions. Values are reported as mean $\pm$ std over all rows at this scale; the standard deviation reflects the dispersion of row-level scores within the $[0, 1]$ range. Arrows indicate directionality: higher is better (↑), lower is better (↓).

Group	Metric	Direct	Binary	SAAG
INTRINSIC	FNEM ↑	1.0000 ± 0.0000	1.0000 ± 0.0000	0.9997 ± 0.0006
	NCS ↑	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000
	RPR ↑	0.9837 ± 0.0049	0.9367 ± 0.0023	0.8857 ± 0.0081
	PDS ↑	0.0013 ± 0.0015	0.0013 ± 0.0012	0.0010 ± 0.0010
	SPR ↓	0.0077 ± 0.0057	0.0013 ± 0.0012	0.0020 ± 0.0010
	AVEM ↑	0.6580 ± 0.0040	0.7170 ± 0.0030	0.7863 ± 0.0090
	QSLO ↑	0.2680 ± 0.0101	0.3307 ± 0.0108	0.3613 ± 0.0351
	VHR ↓	0.2770 ± 0.0060	0.2133 ± 0.0038	0.1657 ± 0.0076
EXTRINSIC	Precision ↑	0.7053 ± 0.0125	0.6483 ± 0.0126	0.6607 ± 0.0126
	Recall ↑	0.7097 ± 0.0130	0.6477 ± 0.0131	0.6543 ± 0.0131
	F1 ↑	0.7050 ± 0.0130	0.6460 ± 0.0131	0.6543 ± 0.0131
STATS	Solve Rate ↑	0.7259 ± 0.0079	0.8288 ± 0.0004	0.8251 ± 0.0048

Table 12: Ministral 3 aggregated across registry scales $S = 5, 10, 15$. Values are reported as mean $\pm$ sample standard deviation over the three scale-level results.

Group	Metric	Direct	Binary	SAAG
INTRINSIC	FNEM ↑	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$
	NCS ↑	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$
	RPR ↑	$0.980_{\pm 0.133}$	$0.988_{\pm 0.100}$	$0.918_{\pm 0.222}$
	PDS ↑	$0.008_{\pm 0.084}$	$0.004_{\pm 0.057}$	$0.007_{\pm 0.077}$
	SPR ↓	$0.016_{\pm 0.117}$	$0.005_{\pm 0.067}$	$0.013_{\pm 0.107}$
	AVEM ↑	$0.652_{\pm 0.437}$	$0.646_{\pm 0.457}$	$0.707_{\pm 0.431}$
	QSLO ↑	$0.327_{\pm 0.227}$	$0.303_{\pm 0.235}$	$0.286_{\pm 0.312}$
	VHR ↓	$0.272_{\pm 0.377}$	$0.288_{\pm 0.411}$	$0.245_{\pm 0.399}$
EXTRINSIC	Precision ↑	$0.762_{\pm 0.401}$	$0.676_{\pm 0.454}$	$0.750_{\pm 0.420}$
	Recall ↑	$0.759_{\pm 0.402}$	$0.672_{\pm 0.453}$	$0.713_{\pm 0.418}$
	F1 ↑	$0.758_{\pm 0.401}$	$0.673_{\pm 0.452}$	$0.725_{\pm 0.415}$
STATS	Solve Rate ↑	0.7151	0.8182	0.7492

Table 13: Granite 4 at registry scale $S = 5$. The top panel reports intrinsic grounding metrics, while the bottom panel shows extrinsic correctness metrics across feedback conditions. Values are reported as mean $\pm$ std over all rows at this scale; the standard deviation reflects the dispersion of row-level scores within the $[0, 1]$ range. Arrows indicate directionality: higher is better (↑), lower is better (↓).

Group	Metric	Direct	Binary	SAAG
INTRINSIC	FNEM ↑	$1.000_{\pm 0.000}$	$0.999_{\pm 0.032}$	$1.000_{\pm 0.000}$
	NCS ↑	$1.000_{\pm 0.000}$	$0.999_{\pm 0.032}$	$1.000_{\pm 0.000}$
	RPR ↑	$0.957_{\pm 0.187}$	$0.974_{\pm 0.151}$	$0.896_{\pm 0.254}$
	PDS ↑	$0.015_{\pm 0.117}$	$0.008_{\pm 0.084}$	$0.014_{\pm 0.114}$
	SPR ↓	$0.030_{\pm 0.156}$	$0.017_{\pm 0.116}$	$0.027_{\pm 0.149}$
	AVEM ↑	$0.647_{\pm 0.437}$	$0.648_{\pm 0.452}$	$0.704_{\pm 0.431}$
	QSLO ↑	$0.314_{\pm 0.227}$	$0.293_{\pm 0.231}$	$0.294_{\pm 0.300}$
	VHR ↓	$0.280_{\pm 0.380}$	$0.286_{\pm 0.410}$	$0.250_{\pm 0.400}$
EXTRINSIC	Precision ↑	$0.748_{\pm 0.407}$	$0.673_{\pm 0.455}$	$0.740_{\pm 0.424}$
	Recall ↑	$0.746_{\pm 0.408}$	$0.669_{\pm 0.455}$	$0.704_{\pm 0.422}$
	F1 ↑	$0.745_{\pm 0.406}$	$0.669_{\pm 0.453}$	$0.715_{\pm 0.418}$
STATS	Solve Rate ↑	0.6910	0.8138	0.7374

Table 14: Granite 4 at registry scale $S = 10$. The top panel reports intrinsic grounding metrics, while the bottom panel shows extrinsic correctness metrics across feedback conditions. Values are reported as mean ± std over all rows at this scale; the standard deviation reflects the dispersion of row-level scores within the $[0, 1]$ range. Arrows indicate directionality: higher is better (↑), lower is better (↓).

Group	Metric	Direct	Binary	SAAG
INTRINSIC	FNEM ↑	$1.000_{\pm 0.000}$	$0.996_{\pm 0.063}$	$0.996_{\pm 0.063}$
	NCS ↑	$1.000_{\pm 0.000}$	$0.996_{\pm 0.063}$	$0.996_{\pm 0.063}$
	RPR ↑	$0.938_{\pm 0.227}$	$0.965_{\pm 0.175}$	$0.964_{\pm 0.178}$
	PDS ↑	$0.021_{\pm 0.135}$	$0.012_{\pm 0.102}$	$0.012_{\pm 0.104}$
	SPR ↓	$0.046_{\pm 0.196}$	$0.020_{\pm 0.133}$	$0.019_{\pm 0.128}$
	AVEM ↑	$0.646_{\pm 0.438}$	$0.642_{\pm 0.456}$	$0.653_{\pm 0.453}$
	QSLO ↑	$0.312_{\pm 0.228}$	$0.293_{\pm 0.229}$	$0.294_{\pm 0.224}$
	VHR ↓	$0.281_{\pm 0.381}$	$0.289_{\pm 0.413}$	$0.283_{\pm 0.412}$
EXTRINSIC	Precision ↑	$0.732_{\pm 0.416}$	$0.653_{\pm 0.461}$	$0.661_{\pm 0.460}$
	Recall ↑	$0.730_{\pm 0.417}$	$0.648_{\pm 0.461}$	$0.654_{\pm 0.459}$
	F1 ↑	$0.729_{\pm 0.415}$	$0.649_{\pm 0.460}$	$0.655_{\pm 0.458}$
STATS	Solve Rate ↑	0.6749	0.8173	0.8259

Table 15: Granite 4 at registry scale $S = 15$. The top panel reports intrinsic grounding metrics, while the bottom panel shows extrinsic correctness metrics across feedback conditions. Values are reported as mean ± std over all rows at this scale; the standard deviation reflects the dispersion of row-level scores within the $[0, 1]$ range. Arrows indicate directionality: higher is better (↑), lower is better (↓).

Group	Metric	Direct	Binary	SAAG
INTRINSIC	FNEM ↑	1.0000 ± 0.0000	0.9983 ± 0.0021	0.9987 ± 0.0023
	NCS ↑	1.0000 ± 0.0000	0.9983 ± 0.0021	0.9987 ± 0.0023
	RPR ↑	0.9583 ± 0.0210	0.9757 ± 0.0116	0.9260 ± 0.0347
	PDS ↑	0.0147 ± 0.0065	0.0080 ± 0.0040	0.0110 ± 0.0036
	SPR ↓	0.0307 ± 0.0150	0.0140 ± 0.0079	0.0197 ± 0.0070
	AVEM ↑	0.6483 ± 0.0032	0.6453 ± 0.0031	0.6880 ± 0.0303
	QSLO ↑	0.3177 ± 0.0081	0.2963 ± 0.0058	0.2913 ± 0.0046
	VHR ↓	0.2777 ± 0.0049	0.2877 ± 0.0015	0.2593 ± 0.0206
EXTRINSIC	Precision ↑	0.7473 ± 0.0150	0.6673 ± 0.0125	0.7170 ± 0.0488
	Recall ↑	0.7450 ± 0.0145	0.6630 ± 0.0131	0.6903 ± 0.0318
	F1 ↑	0.7440 ± 0.0145	0.6637 ± 0.0129	0.6983 ± 0.0379
STATS	Solve Rate ↑	0.6937 ± 0.0202	0.8164 ± 0.0023	0.7708 ± 0.0481

Table 16: Granite Aggregate results across $S = 5, 10, 15$. Values are reported as mean ± sample standard deviation computed over the three iterations.