AgentJudgeBench: A Multi-Difficulty Benchmark for Evaluating LLM Judges on Agentic Tool-Calling

Anonymous ACL submission

Abstract

LLM judges are the de facto standard for evaluating agentic tool-calling systems, yet their reliability in structured, dependency-driven workflows remains largely unexamined. We present AgentJudgeBench, the first benchmark to systematically study LLM-as-a-judge behavior over workflow DAGs. The benchmark comprises 3,808 instances spanning six DAG topologies and three difficulty tiers, evaluated across five generators (3B–70B open-weight models and GPT-5.4), six judges (20B to frontier scale), and both with and without ground-truth settings. Our analysis uncovers a set of consistent and previously unreported phenomena. First, judge alignment degrades monotonically with task difficulty, with degradation occurring 1.5× faster in the absence of ground truth. Second, on hard queries without ground truth, alignment converges to a narrow 77–82% band regardless of judge scale, revealing a structural ceiling that cannot be overcome by model capacity alone. Third, providing ground truth can hurt frontier judges: GPT-5.4 drops by 1.5 pp and Gemini-2.5-Pro by 3.9 pp, consistent with over-anchoring effects. Finally, we evaluate common mitigation strategies. Chain-of-thought reasoning yields negligible gains (at most 0.3 pp), while structured evaluation rubrics improve alignment by +4.8–+6.5 pp over free-form prompting. With ground truth, QwQ-32B achieves the strongest performance; without it, frontier models lead only marginally within the without GT ceiling. Together, our results identify fundamental limitations of current LLM judges and establish practical guidelines for reliable evaluation in agentic systems.

1 Introduction

Using LLMs as automated judges has become standard practice for evaluating model outputs Zheng et al. (2023); Tan et al. (2025); Li et al. (2024b). On text-centric tasks - dialogue, summarization, instruction following - judge reliability is well-characterized, with documented biases and known failure modes Zheng et al. (2023); Wang et al. (2023a). As LLMs are now widely deployed as autonomous agents that invoke tools and orchestrate multi-step workflows, the judge paradigm has extended naturally to evaluate agentic tool-calling Qin et al. (2024); Guo et al. (2024, 2025). This extension, however, has proceeded without a basic calibration check: *how reliable are LLM judges in this structured setting?*

Agentic tool-calling differs from text evaluation in ways that matter for judge reliability. Correctness here is not a matter of fluency or preference - it requires selecting the right tools from a typed schema, supplying well-formed arguments, ordering calls to respect execution dependencies, and covering all parts of the user's intent. A plan can fail in four orthogonal ways (Tool Selection, Parameter Structure, Sequence Accuracy, Query Coverage), and these failure modes do not correlate cleanly. A judge reliable at detecting wrong tool selection may be blind to argument-key errors. A judge calibrated on simple sequential tasks may fail on complex fan-in or diamond workflows where parallel branches must converge. And when no ground-truth execution trace is available - the common deployment scenario - the judge must reconstruct correctness from the query and tool schemas alone, a fundamentally harder inference problem. Existing benchmarks that deploy LLM judges for tool-calling report only aggregate pass-rate agreement and vary none of these dimensions Qin et al. (2024); Guo et al. (2024, 2025). The practical consequence is that practitioners deploying judge-based evaluation pipelines have no principled basis for choosing a judge, setting its configuration, or interpreting its outputs.

We introduce **AgentJudgeBench** to close this gap. The benchmark comprises 3,808 BFCL-style records Patil et al. (2025) spanning six DAG[1]

[1] A Directed Acyclic Graph (DAG) encodes execution dependencies between tool calls: nodes are tool invocations and

topologies at 3 controlled difficulty tiers, each with programmatically verified ground-truth traces. Five generators - four open-weight (3B-70B) and one frontier (GPT-5.4) - produce tool-calling outputs that are scored by six LLM judges(ranging from 20B open-weights to frontier closed systems) under paired with and without GT conditions across four structural metrics (tool selection, parameter structure, sequence accuracy, query coverage), yielding 321,648 paired with-GT/without-GT evaluations (Appendix C). We investigate six research questions: which metrics and topologies are hardest (RQ1), how much judges agree with each other (RQ2), why without GT alignment converges on hard queries (RQ3), and how sensitive alignment is to judge temperature (RQ4), chain-of-thought reasoning (RQ5), and prompt format (RQ6).

The results are non-obvious. Judge alignment degrades monotonically with query difficulty - $1.5\times$ faster without ground truth than with it - and all six judges converge to a 77-82% ceiling on hard queries without a reference, regardless of model capacity. GT exposure is not universally beneficial: two frontier judges (GPT-5.4, Gemini-2.5-Pro) are *less* aligned when shown the reference, consistent with over-anchoring to the provided answer rather than exercising independent judgement. Chain-of-thought reasoning in the judge adds at most 0.3 pp across 24 paired comparisons, but structured per-metric prompt rubrics add $+4.8$-$+6.5$ pp over free-form alternatives - a first-order effect. Inter-judge agreement is moderate ($\kappa \approx 0.42$ with GT) with systematic, capacity-correlated patterns that are not visible in aggregate alignment scores.

Contributions. **(1)** A dataset of 3,808 records across six DAG topologies and three difficulty tiers with programmatically verified ground-truth traces, released publicly. **(2)** A four-metric evaluation framework with a paired with GT/without GT protocol and bootstrap confidence intervals. **(3)** A systematic six-RQ empirical study producing actionable guidance: with ground truth available, QwQ-32B dominates; without it, frontier judges lead narrowly but the convergence ceiling limits the practical difference.

2 Related Work

Table 25 (Appendix S) surveys prior work across three related areas; we explain the key distinctions below.

directed edges indicate that one call must complete before another can begin, with the acyclicity constraint preventing circular dependencies.

2.1 Agentic Data and Tool-Calling Benchmarks

The dominant paradigm for evaluating tool-calling agents uses either environment-based execution feedback Zhou et al. (2024); Drouin et al. (2024); Trivedi et al. (2024); Yao et al. (2024) or deterministic scoring against annotated trajectories Deng et al. (2023); Xu et al. (2025a); Patil et al. (2025). τ-bench Yao et al. (2024) stresses agents against simulated users and domain policies and proposes `pass^k` reliability rather than per-turn correctness; like the rest of this line, it scores agents but does not interrogate the judge that produces those scores. BFCL Patil et al. (2025) is the closest structural ancestor to our data format: it isolates single-turn function-calling using typed JSON schemas and AST-based ground-truth comparison. We adopt the same schema but extend it to multi-step DAG-structured workflows at three controlled difficulty tiers. The most structurally similar work is FuncBenchGen Maekawa et al. (2025), which frames multi-step calling as DAG traversal with controllable complexity. The key distinction is purpose: FuncBenchGen was designed to train and evaluate *generator* agents, whereas AgentJudgeBench is designed to measure the reliability of *judge* models applied to those generators' outputs - a different and complementary objective that requires a paired with GT/without GT evaluation protocol and per-metric decomposition that FuncBenchGen does not provide. TaskBench Shen et al. (2024) shares the tool-dependency graph framing but focuses on decomposition quality rather than judge alignment. Synthetic data pipelines Wang et al. (2023b); Xu et al. (2025b); Cui et al. (2024) inform our generation approach, but target model training rather than evaluation benchmarking.

2.2 LLM-as-Judge

Zheng et al. Zheng et al. (2023) established the LLM-as-judge paradigm on MT-Bench, demonstrating GPT-4's strong agreement with human preferences on open-ended dialogue while identifying positional preference, verbosity sensitivity, and self-enhancement as systematic biases. Subsequent work either extends the evaluation surface - JudgeBench Tan et al. (2025) and Arena-Hard-Auto Li et al. (2024b) move to hard, verifiable response pairs - or builds dedicated judge mod-

els Wang et al. (2024); Li et al. (2024a); Zhu et al. (2023); Kim et al. (2024). JudgeLM Zhu et al. (2023) fine-tunes 7B–33B judges on GPT-4 distilled verdicts and identifies position, knowledge, and format biases that mirror those reported on prompt-based judges; Prometheus 2 Kim et al. (2024) adds rubric-conditioned direct assessment with open weights. Both target text-quality scoring rather than structural tool-calling correctness, but their bias taxonomies inform our prompt-format ablation (§4.2). The jury-of-judges paradigm Verga et al. (2024) shows that ensembling reduces individual judge biases. Crucially, all of this work targets text-centric tasks where judge quality reduces to preference alignment over fluent, semantically coherent outputs. AgentJudgeBench occupies a different regime: correctness is structural, the verdict space is $\{0, 0.5, 1\}$ rather than a preference ranking, and failure modes are multi-dimensional and interdependent.

2.3 LLM Judges for Tool-Calling

LLM judges have been deployed for tool-calling evaluation in several recent benchmarks, but always as an implementation detail rather than the subject of study. ToolLLM Qin et al. (2024) uses ChatGPT to compute pass rates over API trajectories, reporting 87.1% human agreement in aggregate; StableToolBench Guo et al. (2024) and MCP-AgentBench Guo et al. (2025) adopt similar approaches with updated judge models. GeoBenchX Krechetova and Kochedykov (2025) assembles a three-judge panel achieving 88-96% agreement. Agent-as-a-Judge Zhuge et al. (2024) and Auto-Eval Judge Bhonsle et al. (2025) propose more structured evaluation frameworks. In each case, judge reliability is reported as a single aggregate figure on a small validation sample, with no variation of task complexity, difficulty tier, or ground-truth availability. ToolSandbox Lu et al. (2024) is notable for questioning LLM judge reliability directly and replacing judges with programmatic evaluation - but it replaces the judge rather than characterising its failure modes. AgentJudgeBench takes the complementary stance: keep the judge paradigm and measure it systematically, decomposing reliability by metric, topology, difficulty, and condition across 321,648 completed evaluations (Appendix C). A detailed feature comparison across all seven related systems is in Appendix J.

3 Methodology

Figure 1 gives an end-to-end view of the pipeline. Each record is expanded into three difficulty variants; every generator $g \in \mathcal{G}$ produces tool-call predictions; a deterministic programmatic judge scores each prediction to yield reference vector $\mathbf{p}_r$; and every LLM judge $j \in \mathcal{J}$ produces paired verdicts $\ell_{j,r}^{\text{GT}}$, $\ell_{j,r}^{\text{without GT}}$ compared against $\mathbf{p}_r$ via Eq. 5a.

3.1 Data Generation

The first step of the benchmark (BFCL-style agentic data generation) is constructed through a multi-stage synthetic pipeline that generates complete agentic records from minimal seed inputs.

Record construction. Given an enterprise domain label (e.g., IT Service Management, Contract Lifecycle Management, Energy Grid Operations), the pipeline generates use-case scenarios, synthesizes a typed tool inventory as function signatures with constrained return types (`str`, `bool`, `list`, `dict`, `None`), and produces executable pseudocode linking all tool dependencies. A natural language user utterance is then paired with function input/output specifications, and each function is formalized into a JSON schema with typed arguments, required fields, and output definitions. Finally, an ordered execution trace is produced capturing sequential and parallel tool calls, their inputs, outputs, and step descriptions. **DAG topologies.** Records are organized into six topologies derived from analysis of common dependency patterns in real enterprise agentic workflows: *linear*, *fan-out*, *fan-in*, *diamond*, *optional enrichment*, and *loop-like* - representing progressively complex dependency structures from single-path execution to iterative loops.

Difficulty tiers. Each record is expanded into three levels (*easy*, *medium*, *hard*) by increasing query ambiguity while holding the task structure and GT trace constant. Rewrite validation (Appendix L) confirms the medium→hard step on 93.9% of records. The easy→medium step is unanimously validated on only 58.1%; in roughly 41.9% of cases medium is better characterised as a paraphrase than a strict difficulty increase. The primary difficulty-degradation evidence is therefore the medium→hard step; medium data should be read as a robustness check rather than a calibrated mid-point (see Limitations 6).

Validation. GT traces pass a two-level quality gate. First, each record is structurally verified via JSON-schema validation, argument type checking,

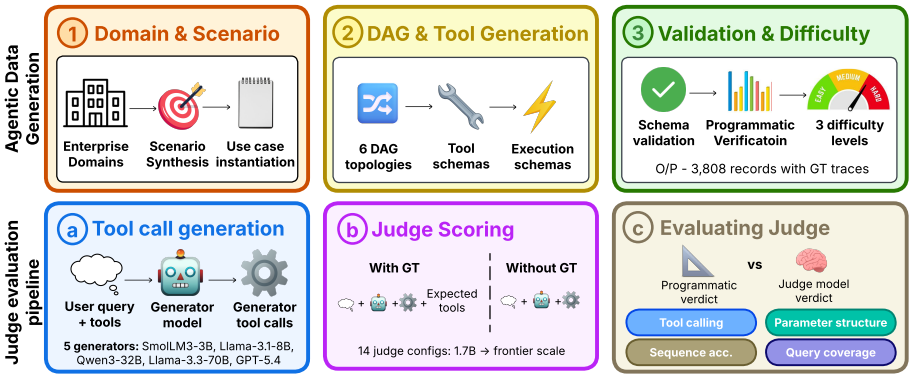

Figure 1: Overall pipeline of AgentJudgeBench. Each BFCL-style Agentic record undergoes difficulty-controlled rewriting, generator inference, and parallel scoring by a programmatic judge and LLM judges (with/without GT), followed by alignment with the programmatic reference.

and trace consistency checks. Second, each (utterance, tool-call) pair is programmatically checked for argument sufficiency, grounding alignment, and query naturalness; failing records are re-generated. No model from $\mathcal{J}$ (or any LLM) is involved in any gate check, ruling out self-reinforcing bias. A 120-record human annotation study on hard-difficulty records (Appendix F), stratified across all six topologies, validates the scorer at 92.7% metric-level agreement with independent human judgement (scope and limitations in Appendix 6). The final corpus comprises 3,808 records spanning 15 enterprise domains with 8–19 tools per record, distributed across six topologies and three difficulty levels.

3.2 Evaluation

As shown in Figure 2, our evaluation is layered: a deterministic *programmatic judge* computes a reference score on every record, an *LLM judge* produces a per-metric verdict on the same record, and an *alignment aggregator* compares the two. Each layer is defined below. Symbol definitions appear in Appendix I; all prompts are reproduced verbatim in Appendix T.

Programmatic Judge Let $G = (g_1, \ldots, g_{|G|})$ denote the generator's predicted ordered sequence of tool calls, and $E = (e_1, \ldots, e_{|E|})$ the GT ordered sequence. Let $\mathrm{name}(\cdot)$ return a tool's identifier, $\mathrm{args}(\cdot)$ its argument-key set, and let $\mathcal{G} = \{\mathrm{name}(g_i)\}$, $\mathcal{E} = \{\mathrm{name}(e_i)\}$ denote the unordered sets of tool identifiers. The programmatic judge emits four independent per-record scores in $[0, 1]$.

Tool-selection accuracy p^{tool} and sequence accuracy p^{seq}. p^{tool} penalises set-level mismatch; p^{seq} measures position-by-position identity, normalised by expected length:

$$p^{\text{tool}} = \max\left(0,\ 1 - \frac{|\mathcal{G} \setminus \mathcal{E}| + |\mathcal{E} \setminus \mathcal{G}|}{\max(|\mathcal{E}|, 1)}\right) \tag{1a}$$

$$p^{\text{seq}} = \frac{1}{\max(|E|, 1)} \sum_{i=1}^{\min(|G|,|E|)} \mathbb{1}[\mathrm{name}(g_i) = \mathrm{name}(e_i)] \tag{1b}$$

with $p^{\text{tool}} = p^{\text{seq}} = 1$ when the respective sequences are empty.

Parameter-structure accuracy p^{param}. For each predicted call $g \in G$ let $A_g = \mathrm{args}(g)$ and let A_g^* be the expected-argument key set of the call in E with the same tool identifier (undefined if the predicted tool is not in $\mathcal{E}$). Define per-call structural scores

$$s(g) = \begin{cases} 1.0 & \text{if } A_g = A_g^* \\ & \text{(all keys present, no extras)} \\ 0.5 & \text{if } A_g^* \subseteq A_g \\ & \text{and } A_g \setminus A_g^* \neq \emptyset \\ & \text{(extras only)} \\ 0.0 & \text{if } A_g^* \not\subseteq A_g \\ & \text{or } A_g^* \text{ undefined} \end{cases} \tag{2}$$

and aggregate by mean: $p^{\text{param}} = \left(\sum_{g \in G} s(g)\right) / \max(|G|, 1)$, with $p^{\text{param}} = 1$ when G is empty.

Query-coverage accuracy p^{cov}. Fraction of expected tool identifiers covered by the prediction, ignoring extras:

$$p^{\text{cov}} = \begin{cases} 1 & \text{if } \mathcal{E} = \emptyset, \\ 0 & \text{if } \mathcal{G} \cap \mathcal{E} = \emptyset, \\ |\mathcal{G} \cap \mathcal{E}| / |\mathcal{E}| & \text{otherwise.} \end{cases} \tag{3}$$

Record-level aggregate. Each record receives a four-element programmatic vector $\mathbf{p}_r = (p_r^{\text{tool}}, p_r^{\text{param}}, p_r^{\text{seq}}, p_r^{\text{cov}})$. Where a scalar is needed - specifically in the generator-accuracy reporting of Table 2 - we use the unweighted mean $\bar{p}_r =$

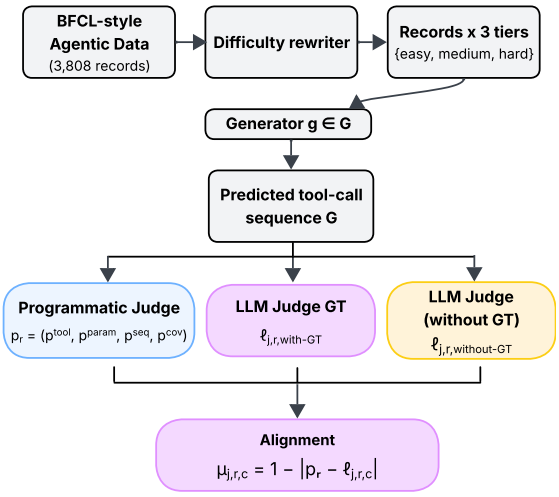

Figure 2: Evaluation pipeline of AgentJudgeBench. Records expand into three difficulty tiers, LLM judges (with/without GT) and programmatic judge score the generated tool-call predictions.

$\frac{1}{M}\sum_{m=1}^{M} p_r^m$ where $M = 4$ is the number of metrics; elsewhere the four components are kept separate and averaged independently. We adopt equal weights because the four metrics target orthogonal aspects of tool-call correctness and we have no principled basis for preferring one over another when reporting a single headline number; Section 4.2 presents per-metric breakdowns that practitioners can reweight for a given application (e.g., upweighting sequence accuracy when judging workflows with strict ordering constraints). Empirically, the judge ranking is stable under alternative metric weightings: Spearman $\rho \geq 0.83$[2] between the equal-weight ranking and five alternative schemes (seq×2, param×2, cov×0.5, seq×2+param×2), confirming that the equal-weight aggregate is an adequate proxy for any practitioner-specific weighting.

LLM Judge The same generator outputs scored by the programmatic judge are independently evaluated by every LLM judge $j \in \mathcal{J}$. Each judge receives a structured prompt containing the original user query, the full set of available tool schemas, and the generator's predicted tool-call sequence. The judge is asked to produce a single JSON object scoring each of the four metrics (tool selection, parameter structure, sequence accuracy, query coverage) on a $\{0, 0.5, 1\}$ scale, along with a one-sentence justification per metric and an overall assessment.

[2]Spearman's ρ is a rank correlation coefficient measuring the monotone association between two orderings; $\rho = 1$ indicates identical rankings, $\rho = 0$ no association, and $\rho = -1$ perfectly reversed rankings.

Two prompt variants exist, corresponding to the conditions $c \in \{\text{with-GT}, \text{without-GT}\}$: the **with GT** prompt additionally exposes the GT tool-call sequence as a reference block, while the **without GT** prompt omits it entirely, requiring the judge to assess correctness from the query and tool schemas alone. Judge decoding is held constant across all (g, j, d, c) configurations to isolate the effect of our independent variables. The full prompt bodies appear in Appendix T.

Alignment Against the Programmatic Reference Given a record r, condition c, judge j, and metric $m \in \{\text{tool}, \text{param}, \text{seq}, \text{cov}\}$, let $\ell_{j,r,c}^m \in \{0, 0.5, 1\}$ denote the judge's verdict and p_r^m the programmatic value. The per-metric match score and record-level aggregate are:

$$\mu_{j,r,c}^m = 1 - \left|p_r^m - \ell_{j,r,c}^m\right| \tag{4a}$$

$$\mu_{j,r,c} = \frac{1}{M}\sum_{m=1}^{M} \mu_{j,r,c}^m \tag{4b}$$

exact match on all metrics yields $\mu = 1$ and maximally divergent verdict yields 0. The configuration-level alignment percentage and GT lift are:

$$\text{align}(j, g, d, c) = \frac{100}{N_{g,d}} \sum_{r=1}^{N_{g,d}} \mu_{j,r,c} \tag{5a}$$

$$\text{lift}(j) = \overline{\text{align}}_{\text{GT}}(j) - \overline{\text{align}}_{\text{without GT}}(j) \tag{5b}$$

where the overline denotes the mean over all 12 (g, d) configurations. Perfect reproduction of the programmatic vector yields $\text{align} = 100\%$; an independent judge attains 50% in expectation.

In the experiments, we report both aggregate alignment $\text{align}(j, g, d, c)$ and per-metric breakdowns to identify which dimensions of tool-calling correctness judges find most difficult to assess.

4 Experiments

We use AgentJudgeBench to evaluate the extent to which LLM-judge alignment depends on the generator, query difficulty, and ground-truth availability. All protocol details (prompts, scoring, conditions) are in Section 3; model identifiers are in Appendix K and Appendix M.

4.1 Experimental Setup

We instantiate a fully-crossed factorial over three factors: generator $g \in \mathcal{G}$ (five models, 3B to frontier), judge $j \in \mathcal{J}$ (six LLMs), and difficulty $d \in \{\text{easy}, \text{medium}, \text{hard}\}$. Each cell is ob-

served under both with GT and without GT conditions, yielding 90 factorial cells and 321,648 valid (generator, judge, difficulty, record) tuples. Exact per-cell counts are in Appendix C. We report $\mathrm{align}(j, g, d, c)$ (Eq. 5a) and, where relevant, GT lift (Eq. 5b).

4.2 Results and Analysis

Table 2 reports every configuration in the factorial design. Each generator occupies a sub-block with six rows corresponding to the judges in $\mathcal{J}$. Columns pair each difficulty level with its with GT and without GT conditions. Bold entries mark the highest alignment within each (difficulty, condition) column of each sub-block.

Finding 1: Monotone difficulty degradation. All 30 (generator, judge) pairs exhibit strictly monotone alignment degradation from easy to hard under both with GT and without GT conditions (Table 2). without GT degradation magnitude is roughly $1.5\times$ larger than GT degradation. On hard without GT records, all six judges converge to a narrow 77-82% band across four of five generators (including the frontier GPT-5.4 generator), indicating a task-level rather than judge-level ceiling. Difficulty degradation curves are in Appendix E.

Scope of Finding 1. The medium→hard drop (≈5-7 pp with GT) is the primary evidence for difficulty-driven degradation; hard rewrites are unanimously validated on 93.9% of records (Appendix L), while the easy→medium step (unanimous on 58.1%) partially reflects paraphrase-robustness.

Judge	Diff	GT	withoutGT	cGT	$\Delta_{\text{cGT-GT}}$	$\Delta_{\text{cGT-noGT}}$
QwQ-32B	Easy	96.6	95.6	95.5	−1.1	−0.1
	Medium	92.3	92.0	92.0	−0.3	0.0
	Hard	84.6	84.4	84.6	0.0	+0.2
Gemini	Easy	90.7	95.7	92.5	+1.8	−3.2
	Medium	86.4	92.1	86.4	0.0	−5.7
	Hard	78.9	84.6	78.9	0.0	−5.7

Table 1: Full C3 results on Llama-3.3-70B generator. All values are mean alignment (%) over 3,764–3,771 records per cell. **cGT**: corrupted-GT condition. **Gemini**: Gemini 2.5 Pro. $\Delta_{\text{cGT-GT}}$: corrupted-GT minus standard GT. $\Delta_{\text{cGT-noGT}}$: corrupted-GT minus without GT.

Finding 2: Ground-truth exposure is not monotonically beneficial. GT lift (Eq. 5b) is positive for QwQ-32B and GPT-OSS-120B and reliably negative for GPT-5.4 and Gemini-2.5-Pro (non-overlapping bootstrap CIs; Appendix D). Figure 3 shows the effect concentrates in sequence accuracy: frontier judges anchor to the GT trace's ordering and penalise functionally equivalent but structurally deviant sequences. Case studies in Appendix G.

		Easy		Med.		Hard	
Gen.	Judge	GT	No GT	GT	No GT	GT	No GT
Sm3B	OSS20	84.8	85.3	81.7	79.3	77.4	69.0
	QwQ32	**90.0**	86.7	**84.8**	80.3	77.2	69.7
	OSS120	89.2	86.3	**84.8**	80.2	**77.8**	69.7
	ClS4	86.5	86.8	81.3	80.9	74.2	71.2
	Gem	81.1	86.6	77.7	80.7	73.4	70.8
	GPT5	86.0	**87.3**	81.6	**82.0**	76.9	**73.3**
Ll8B	OSS20	87.4	89.5	85.3	85.7	81.2	75.9
	QwQ32	**94.2**	91.2	**90.8**	86.8	**84.1**	76.6
	OSS120	93.1	90.7	89.8	86.6	83.6	76.6
	ClS4	90.3	90.6	86.5	86.6	80.3	77.1
	Gem	85.9	**91.9**	83.3	**87.6**	78.7	78.2
	GPT5	88.5	91.4	85.0	87.5	80.6	**79.0**
Qw32B	OSS20	88.4	93.2	85.8	88.7	82.7	81.1
	QwQ32	**95.0**	93.9	**90.9**	89.1	**84.7**	81.1
	OSS120	93.9	94.0	90.0	89.2	84.4	81.3
	ClS4	92.7	94.0	88.5	89.2	82.1	81.6
	Gem	86.4	**94.1**	83.0	89.2	78.1	81.4
	GPT5	90.5	93.9	86.4	**89.4**	81.9	**81.7**
Ll70B	OSS20	89.0	94.5	87.2	91.2	83.9	84.0
	QwQ32	**96.6**	95.6	**93.6**	92.1	**87.5**	84.4
	OSS120	96.0	95.5	93.0	92.1	87.2	84.5
	ClS4	93.3	94.6	89.9	90.7	83.7	83.3
	Gem	90.5	**95.7**	86.5	**92.2**	82.2	**84.6**
	GPT5	91.4	95.2	87.4	91.7	83.0	**84.6**
GPT5	OSS20	83.1	85.6	82.1	83.9	79.4	79.5
	QwQ32	85.7	85.8	84.3	84.2	80.5	79.8
	OSS120	84.8	85.3	83.9	83.3	80.7	78.6
	ClS4	**86.1**	**87.3**	**84.5**	**85.4**	80.7	**81.2**
	Gem	81.6	85.2	80.2	83.3	77.6	78.4
	GPT5	85.8	87.0	84.1	84.9	**81.4**	79.8

Table 2: LLM–judge alignment (%) with the programmatic reference across generators, judges, difficulty, and GT condition. Bold = column-wise maximum within each generator block.

C3 control. Replacing the reference with a wrong GT from a different record confirms pure anchoring (Table 1): Gemini-2.5-Pro aligns identically under standard and corrupted GT; QwQ-32B tracks without GT within 0.2 pp. Full results in Appendix H.

Finding 3: Best judge depends on configuration. QwQ-32B leads GT alignment in 10 of 15 (generator, difficulty) cells but is never the top without GT judge; Gemini-2.5-Pro and GPT-5.4 lead without GT on stronger and weaker generators respectively. See Appendix A for a deployment decision table.

The following RQs explain the mechanisms behind Findings 1-3.

RQ1: Which metrics and DAG topologies are hardest for judges?

Figure 3 breaks alignment into per-metric components. With GT, sequence accuracy is the weakest dimension with substantial inter-judge spread;

[2]Cohen's κ measures inter-rater agreement corrected for chance: (0=chance, 1=perfect; 0.2–0.4 "fair," 0.4–0.6 "moderate," $\geq$0.6 "substantial") agreement. All κ values here fall in the fair-to-substantial range (0.225–0.606), consistent with the structural complexity of the four-metric scoring task.

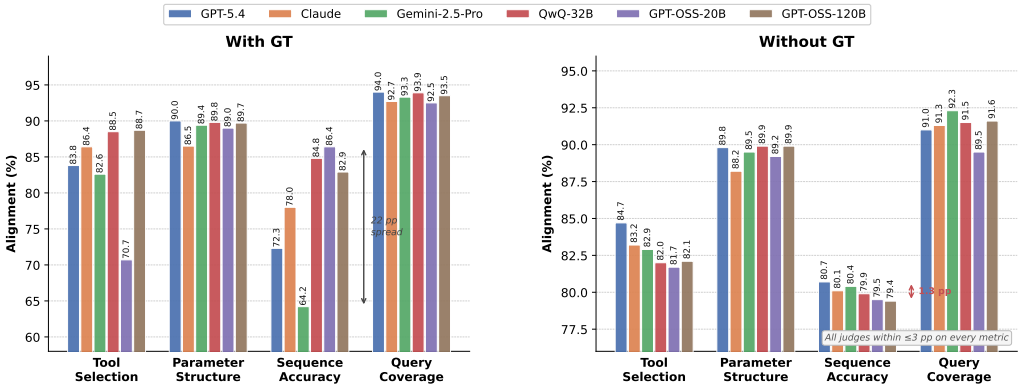

Figure 3: Per-metric alignment (with and without GT) across six judges.

	OSS-20B	QwQ-32B	OSS-120B	Claude S. 4.0	Gemini	GPT-5.4
OSS-20B	-	77.4	78.4	70.4	74.5	75.2
QwQ-32B	0.386	-	**89.4**	85.1	80.7	79.7
OSS-120B	0.414	**0.606**	-	83.2	81.6	79.8
Claude S. 4.0	0.225	0.490	0.424	-	76.5	78.4
Gemini	0.387	0.411	0.437	0.328	-	76.4
GPT-5.4	0.419	0.446	0.449	0.430	0.432	-

Table 3: Pairwise inter-judge agreement under the with GT condition. Upper triangle : exact metric-level agreement (%); Lower triangle : Cohen's κ[4].

Generator	1.0-default	0.5-default	Δ
Llama-3.3-70B	84.3	84.7	+0.4
Qwen3-32B	81.4	82.4	+1.0
GPT-5.4	79.5	80.0	+0.5
Llama-3.1-8B	77.2	81.3	+4.1
SmolLM3-3B	70.6	76.2	+5.6

Table 4: Hard without GT alignment (%) under the standard 1.0-default and the recalibrated 0.5-default prompts, averaged across six judges per generator. Δ = (0.5-default) − (1.0-default).

without GT, all judges compress to a narrow per-metric band, reflecting the 1.0-default rubric. DAG topology imposes a judge-independent difficulty ordering (*fan out* easiest, *loop like/fan in* hardest); full numerical breakdowns in Appendix N and Appendix P.

RQ2: How much do judges agree with each other?

Table 3 reports pairwise judge agreement. Mean agreement is 79.1% ($\kappa = 0.419$) with GT and 92.6% ($\kappa = 0.559$) without GT; the higher without GT value reflects prompt-driven verdict compression rather than genuine consensus. The highest agreement is QwQ-32B×GPT-OSS-120B (89.4%, $\kappa = 0.606$); the lowest is Claude Sonnet 4.0×GPT-OSS-20B (70.4%, $\kappa = 0.225$). Under without GT, pairwise κ decreases monotonically with judge tier separation ($\rho = -0.825$[3], $p < 0.01$); this relationship vanishes under GT ($\rho = -0.171$), where GT acts as a shared anchor. Disagreement concentrates at the 0.5 partial-credit boundary (94-97% of off-diagonal entries); verdict-level confusion matrices in Appendix Q, binary scale ablation in Appendix Q.

A soft jury of all six judges matches but does not exceed the top individual judge (82.5% with GT hard) Verga et al. (2024); ensemble and individual rankings against human annotators are in Appendix F.

RQ3: Why does without GT alignment converge on hard queries?

The 77-82% hard without GT convergence (Table 2) has three consistent explanations: (1) the without GT prompt defaults to 1.0, suppressing hard-query discrimination; (2) generator error peaks on hard records, compressing all judges identically; (3) per-metric compression is uniform (Table 20). The six-judge ensemble achieves 77.6% — matching the best individual judge — confirming correlated failure driven by a shared structural property, not independent per-judge noise.

C2 ablation: 0.5-default without GT prompt. Table 4 compares hard without GT alignment under the standard 1.0-default and a recalibrated 0.5-default prompt. For the three strongest generators the delta is $\leq$+1.0 pp, confirming structural task difficulty as the primary ceiling driver. For weaker generators (+4.1-+5.6 pp), the 1.0-default over-credits incorrect outputs; practitioners evaluating low-quality generators should consider the 0.5-default prompt.

RQ4: Does judge temperature affect alignment?

Figure 4 shows Qwen3-32B alignment is insensitive to temperature across $T \in \{0.3, 0.7, 1.0\}$:

[3]Spearman's ρ measures the rank correlation between the pairwise judge tier gap (by parameter count) and their κ agreement score; $|\rho| > 0.7$ indicates a strong monotone relation.

maximum spread 0.6 pp across all (difficulty, condition) cells. Structural pattern-matching dominates; full table in Appendix O.

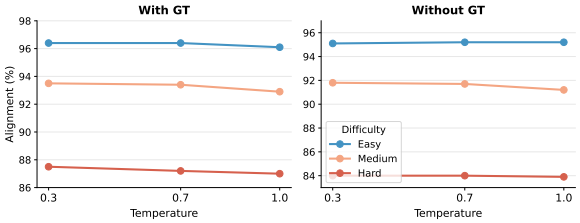

Figure 4: **Judge temperature sensitivity.** Qwen3-32B alignment at $T \in \{0.3, 0.7, 1.0\}$ on Llama-3.3-70B. All three difficulty curves are near-flat; maximum spread ≤ 0.6 pp.

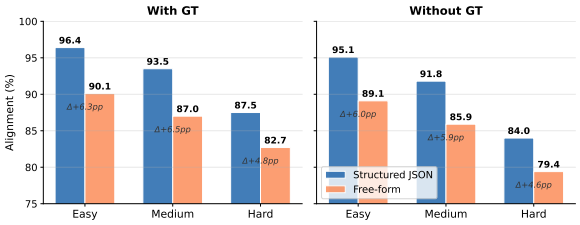

Figure 5: **Prompt structure vs. alignment.** Structured JSON prompt vs. free-form for Qwen3-32B on Llama-3.3-70B. The per-metric rubric adds +4.8-+6.5 pp with GT across all difficulty levels.

RQ5: Does chain-of-thought reasoning in the judge improve alignment?

Table 5 reports QwQ-32B alignment with thinking on vs. off across four open-weight generators and three difficulty levels (24 paired cells). CoT reasoning contributes negligibly: mean GT gap +0.11 pp, maximum per-cell 0.3 pp; no cell shows a difference exceeding 0.3 pp on either condition. QwQ-32B's advantage in the main results therefore reflects training distribution rather than inference-time compute. Full per-generator grid is reproduced in Appendix O.

RQ6: Does prompt output format affect alignment?

Figure 5 shows the structured per-metric JSON prompt outperforms a free-form variant by +4.8-+6.5 pp with GT across all difficulty levels, confirming prompt structure as a first-order alignment determinant. Full table in Appendix O.

5 Conclusion

We introduce **AgentJudgeBench**, a benchmark for measuring LLM-judge reliability on structured, dependency-driven tool-calling - a setting where

		Easy		Medium		Hard	
Generator	**Setting**	**GT**	**No GT**	**GT**	**No GT**	**GT**	**No GT**
Llama-3.3-70B	Thinking on	96.6	95.6	93.6	92.1	87.5	84.4
	Thinking off	96.5	95.6	93.6	92.0	87.3	84.4
	Δ	+0.1	0.0	0.0	+0.1	+0.2	0.0
Llama-3.1-8B	Thinking on	94.2	91.2	90.8	86.8	84.1	76.6
	Thinking off	94.3	91.4	90.8	86.8	83.9	76.7
	Δ	−0.1	−0.2	0.0	0.0	+0.2	−0.1
Qwen3-32B	Thinking on	95.0	93.9	90.9	89.1	84.7	81.1
	Thinking off	94.9	94.0	90.8	89.0	84.7	81.1
	Δ	+0.1	−0.1	+0.1	+0.1	0.0	0.0
SmolLM3-3B	Thinking on	90.0	86.7	84.8	80.3	77.2	69.7
	Thinking off	89.8	86.7	84.6	80.4	76.9	69.7
	Δ	+0.2	0.0	+0.2	−0.1	+0.3	0.0

Table 5: Chain-of-thought reasoning in the QwQ-32B judge: alignment (%) with thinking on vs. off across four open-weight generators. Δ = thinking-on − thinking-off (pp).

existing LLM-as-judge work offers little calibration. Three design choices distinguish it: difficulty-stratified rewrites (easy / medium / hard preserving the ground-truth trace), per-record DAG-topology annotation across six patterns, and a paired with GT / without GT evaluation protocol.

Across 321,648 completed evaluations, four patterns emerge. Firstly, alignment degrades monotonically with difficulty, approximately 1.5× faster without ground truth than with it. Secondly, hard without GT alignment converges to a 77-82% band regardless of judge capacity - a task-level ceiling confirmed by the C2 recalibrated-prompt ablation (≤+1.0 pp for capable generators). Next, GT exposure is counterproductive for frontier judges (Gemini-2.5-Pro: −3.9 pp; GPT-5.4: −1.5 pp), consistent with over-anchoring. Lastly, prompt structure is the dominant configuration lever: structured rubrics add +4.8-+6.5 pp; CoT reasoning adds ≤0.3 pp. The practical guidance is direct: **with ground truth, use QwQ-32B; without it, frontier judges lead narrowly but the without GT ceiling limits the choice.** Limitations and deployment guidance are in Appendices 6 and A.

6 Supplementary Material

The anonymised supplementary material contains: all 3,808 records (JSONL with DAG annotations and difficulty rewrites); the programmatic judge implementation (Eqs. 1a-5a); the LLM-judge runner with full prompt templates (Appendix T); the aggregated per-configuration results underlying every table and figure; and a reproducibility script for the main results. On acceptance the full repository will be released under a permissive open-source license.

Limitations

Generator scope. Four of five generators are open-weight (3B–70B); GPT-5.4 is the sole frontier representative. Its generator block (Table 2) confirms that degradation patterns and the without GT ceiling generalise to frontier-quality outputs: hard without GT alignment averages 79.5% across six judges, within the same band as the open-weight generators. Additional frontier generators (e.g., Claude Sonnet) remain deferred to future work.

GPT-5.4 reproducibility. "GPT-5.4" refers to a GPT-5 preview variant deployed on Azure OpenAI as of April 2026 under `gpt-5.4` (Appendix M); this is not a stable public endpoint. We treat these cells as a *non-reproducible reference snapshot*: raw per-record outputs are included in the supplementary release so future work can audit without re-querying. All primary claims—monotone difficulty degradation, the without GT ceiling, over-anchoring, judge ranking stability, and prompt-format dominance—replicate on the four open-weight generators independently of GPT-5.4.

Programmatic scorer. The scorer measures structural correctness (argument keys present, ordering correct) but not semantic equivalence (argument-equivalent but syntactically different JSON). Of metric-level mismatches for QwQ-32B (best-aligned judge, 12-cell evaluation), 25.9% fall on parameter structure—the axis most susceptible to this gap—while the remaining 74.1% are on tool selection (24.9%), sequence accuracy (39.2%), and query coverage (10.0%), where structural and semantic correctness largely coincide. Human annotation (Appendix F) reports 92.7% metric-level agreement, confirming that mismatches are structural rather than artefacts of scorer noise.

Synthetic data. The dataset is entirely pipeline-generated across 15 enterprise domains. Comparison with real-world agentic execution traces is needed to rule out systematic domain drift. This is the primary open validity concern and is tracked as future work.

Difficulty rewriting. The easy→medium hardening is unanimously validated on only 58.1% of records; a subset of medium variants are better characterised as paraphrases than genuine hardenings. The medium→hard step is unanimously validated on 93.9% of records and is the paper's primary difficulty-degradation evidence.

Without GT prompt prior. The convergence ceiling on hard without GT records is partly driven by the "default to 1.0" rubric in the without GT prompt. A 0.5-default ablation (Section 4.2, Table 4) confirms the ceiling persists for capable generators ($\leq$+1.0 pp change), confirming structural task difficulty as the primary driver. For weaker generators, the 1.0-default over-credits incorrect outputs by up to +5.6 pp; practitioners evaluating low-quality generators should consider the 0.5-default prompt.

Judge roster. We evaluate seven judge configurations (six main judges plus Qwen3-1.7B in the scale ablation). Additional open judges and broader frontier configurations are deferred to future work.

Human-judge upper bound. The 77–82% without GT ceiling is better interpreted against the human-agreement upper bound. In our 120-record annotation study (Appendix F), human annotators agree with the programmatic scorer at 92.7% metric-level agreement. The hard without GT ceiling is therefore 11–16 pp below human-level—a meaningful but not catastrophic gap. On easy without GT queries, the best LLM judges ($\sim$92%) already approach human-level; the gap widens as difficulty increases and the judge must reason without a reference.

Single-cell ablations. RQ4 (temperature) and RQ6 (prompt format) each use a single test-bed cell: Qwen3-32B on Llama-3.3-70B outputs. RQ5 (CoT) covers four open-weight generators (24 cells). The temperature finding is directionally consistent with the broader CoT grid, but the format ablation remains a single (judge, generator) observation and should be interpreted as suggestive rather than definitive for other pairings.

Topology weighting. Headline figures use unweighted averages across the six DAG topologies. The corpus is intentionally imbalanced to mirror enterprise prevalence: fan-in accounts for 27.5% of records, loop-like for only 5.9% (Table 7). Under count-weighted averaging, figures shift by $\leq$1.5 pp; judge rankings are unaffected. Practitioners with different topology distributions can recompute from Appendix P.

Ethical Considerations

Over-reliance on automated evaluation. Our findings reveal systematic failure modes in LLM judges, including the 77–82% without-GT ceiling and over-anchoring in frontier models. If practitioners adopt judge-based pipelines without awareness of these limitations, incorrect tool-calling outputs

may be certified as correct, particularly in safety-critical enterprise domains (e.g., energy grid operations, healthcare workflows) represented in our dataset. We mitigate this by providing explicit deployment guidance (Appendix A) and recommending against sole reliance on any single LLM judge.

Evaluation monoculture. All six judges converge to similar failure patterns on hard queries, suggesting shared training-distribution biases. Over-reliance on our benchmark rankings could reinforce a monoculture where the same blind spots propagate across evaluation pipelines. We encourage practitioners to complement LLM judges with programmatic scoring and human review where feasible.

Dual use and gaming. The detailed analysis of judge failure modes—over-anchoring, the 1.0-default bias, and prompt-format sensitivity—could be exploited to game judge-based evaluation systems, for instance by crafting outputs that exploit known judge blind spots to inflate scores. We release all prompts, scorer code, and raw outputs to enable the community to develop countermeasures.

Synthetic data. All benchmark records are synthetically generated. While this avoids privacy concerns associated with real user data, the 15 enterprise domains reflect the biases of the generation pipeline and may not represent underrepresented industries, non-English workflows, or workflows from historically marginalised communities.

Compute cost. The full evaluation comprises 321,648 paired judge calls across six LLMs, five generators, and three difficulty tiers. While we do not report precise energy estimates, the inference-time compute is non-trivial. Our practitioner decision guide (Appendix A) is designed to reduce unnecessary exhaustive evaluation by recommending specific judges per deployment scenario.

References

Roshita Bhonsle, Rishav Dutta, Sneha Vavilapalli, Harsh Seth, Abubakarr Jaye, Yapei Chang, Mukund Rungta, Emmanuel Aboah Boateng, Sadid Hasan, Ehi Nosakhare, and Soundar Srinivasan. 2025. AutoEval Judge: Towards a general agentic framework for task completion evaluation. *arXiv preprint arXiv:2508.05508*.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2024. UltraFeedback: Boosting language models with scaled AI feedback. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. 2024. WorkArena: How capable are web agents at solving common knowledge work tasks? In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pages 11642–11662.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. 2024. StableToolBench: Towards stable large-scale benchmarking on tool learning of LLMs. *arXiv preprint arXiv:2403.07714*.

Zikang Guo, Benfeng Xu, Chiwei Zhu, Wentao Hong, Xiaorui Wang, and Zhendong Mao. 2025. MCP-AgentBench: Evaluating real-world language agent performance with MCP-mediated tools. *arXiv preprint arXiv:2509.09734*.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2024. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Varvara Krechetova and Denis Kochedykov. 2025. GeoBenchX: Benchmarking LLMs in agent solving multistep geospatial tasks. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Generative and Agentic AI for Multi-Modality Space-Time Intelligence*.

Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, Hai Zhao, and Pengfei Liu. 2024a. Generative judge for evaluating alignment. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. 2024b. From crowdsourced data to high-quality benchmarks: Arena-Hard and BenchBuilder pipeline. *arXiv preprint arXiv:2406.11939*.

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. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Jiarui Lu, Thomas Holleis, Yizhi Zhang, Bernhard Aumayer, Feng Nan, Felix Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. 2024. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. *arXiv preprint arXiv:2408.04682*.

Seiji Maekawa, Jackson Hassell, Pouya Pezeshkpour, Tom Mitchell, and Estevam Hruschka. 2025. Towards reliable benchmarking: A contamination-free, controllable evaluation framework for multi-step LLM function calling. *arXiv preprint arXiv:2509.26553*.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, 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 (ICML)*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, 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 (ICLR)*.

Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. 2024. TaskBench: Benchmarking large language models for task automation. In *Advances in Neural Information Processing Systems*.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. 2025. JudgeBench: A benchmark for evaluating LLM-based judges. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. 2024. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Pat Verga, Sebastian Hofstätter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. 2024. Replacing judges with juries: Evaluating LLM generations with a panel of diverse models. *arXiv preprint arXiv:2404.18796*.

Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023a. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, and Yue Zhang. 2024. PandaLM: An automatic evaluation benchmark for LLM instruction tuning optimization. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508.

Yiheng Xu, Dunjie Lu, Zhennan Shen, Junli Wang, Zekun Wang, Yuchen Mao, Caiming Xiong, and Tao Yu. 2025a. AgentTrek: Agent trajectory synthesis via guiding replay with web tutorials. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. 2025b. Magpie: Alignment data synthesis from scratch by prompting aligned LLMs with nothing. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. WebArena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. 2023. JudgeLM: Fine-tuned large language models are scalable judges. *arXiv preprint arXiv:2310.17631*.

Mingchen Zhuge, Changsheng Zhao, Dylan Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, Yangyang Shi, Vikas Chandra, and Jürgen Schmidhuber. 2024. Agent-as-a-judge: Evaluate agents with agents. *arXiv preprint arXiv:2410.10934*.

A Practitioner Decision Guide

In most deployment scenarios ground-truth tool-call sequences are unavailable at inference time; the without GT judge is therefore the practical baseline. Table 6 reports the best-performing judge for each deployment scenario, derived directly from Table 2. Rankings are computed as an *unweighted* mean of per-cell alignment over all five generators per (condition, difficulty) combination; count-weighted averaging shifts results by ≤ 0.3 pp.

Condition	Difficulty	Best judge	Runner-up
Without GT	Easy	Gemini-2.5-Pro (92.1%)	GPT-5.4 (92.0%)
Without GT	Medium	GPT-5.4 (87.7%)	Gemini-2.5-Pro (87.4%)
Without GT	Hard	GPT-5.4 (79.7%)	Gemini-2.5-Pro (78.8%)
With GT	Easy	QwQ-32B (94.0%)	GPT-OSS-120B (93.0%)
With GT	Medium	QwQ-32B (90.0%)	GPT-OSS-120B (89.4%)
With GT	Hard	QwQ-32B (83.4%)	GPT-OSS-120B (83.2%)

Table 6: Best judge per deployment scenario. The without GT rows reflect the common deployment setting where ground-truth tool calls are unavailable. GT = ground-truth tool calls available; without GT = judge uses only query and tool schemas. Alignment (%) is unweighted mean over five generators.

Takeaways. (1) **No ground truth** (the common case): Gemini-2.5-Pro and GPT-5.4 lead narrowly (≤ 1 pp), but the without GT convergence ceiling (§4.2) makes judge choice less consequential on hard queries - the practical difference between any two judges is ≤ 2 pp. (2) **Ground truth available**: QwQ-32B dominates all difficulty levels; GPT-OSS-120B is the best open-weight alternative without reasoning-model inference overhead. (3) **Single judge across all conditions**: QwQ-32B offers the best mean GT alignment (89.1%) with competitive without GT performance (85.6%). (4) **Binary pass/fail pipelines**: Practitioners scoring verdicts on a binary pass/fail scale rather than $\{0, 0.5, 1\}$ should prefer GPT-OSS-20B over QwQ-32B. Under binary verdict collapse, judge rankings differ substantially from the three-point scale (Spearman $\rho = 0.03$; Table 24, Appendix Q): judges with high 0.5-verdict rates (GPT-OSS-20B: 26.5%; GPT-5.4: 28.2%) gain a systematic advantage because every 0.5 verdict is remapped toward the programmatic direction, inflating their binary alignment. QwQ-32B's structural advantage over GPT-OSS-20B diminishes or reverses under binary scoring. (5) **Human-intuition alignment**: Practitioners whose primary concern is agreement with human annotators rather than with the programmatic scorer should prefer GPT-OSS-120B, which tops both the programmatic and human-verdict orderings ($\Delta = -2.8$ pp vs. human; Table 13, Appendix F).

B Dataset Statistics

This appendix reports the structural composition of the 3,808 AgentJudgeBench records, covering topology distribution, tool inventory size, parameter depth, and expected call-sequence length. All statistics are computed over the unique-record pool (i.e., collapsing the three difficulty rewrites of each record into one, since they share the same tool schemas and ground-truth trace). Figures 6 and 7 visualise the distributions; Tables 7 and 8 report the precise counts.

Topology distribution (Table 7, Figure 6). Records are not uniformly distributed across topologies. Fan-in (27.5%) and optional enrichment (21.6%) are the most prevalent, reflecting the frequency of multi-source aggregation and conditional enrichment patterns in the 15 enterprise seed domains. Loop-like records are the rarest (5.9%) because iterative workflows are less commonly expressed as single-turn tool-call sequences in the generation pipeline. Linear records (21.3%) serve as the structural baseline — they require no dependency tracking and are included to anchor the difficulty gradient. The imbalance is intentional: it mirrors the relative prevalence of each pattern in enterprise agentic workloads rather than imposing artificial uniformity.

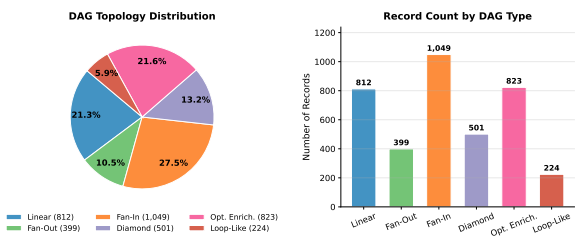

Figure 6: DAG topology distribution across the 3,808 unique records (left: proportional pie, right: absolute counts). Fan-in accounts for over a quarter of all records; loop-like is the rarest at 5.9%.

DAG Type	Records	%
Linear	812	21.3
Fan-Out	399	10.5
Fan-In	1,049	27.5
Diamond	501	13.2
Optional Enrichment	823	21.6
Loop-Like	224	5.9
Total	**3,808**	**100.0**

Table 7: Record counts by DAG topology. Counts reflect unique records; each record has three difficulty variants (easy / medium / hard), giving $3{,}808 \times 3 = 11{,}424$ total rows in the full dataset.

Tool inventory and parameter depth (Table 8, Figure 7). Each record exposes a pool of 8-19 available tools (mean 12.9), of which the ground-truth trace invokes 2-5 (mean 3.3). The deliberate over-provision of tools — the ground truth uses on average only 25% of the available pool — forces the generator and judge to perform genuine tool selection rather than selecting by elimination. Each tool carries 0-19 typed parameters (mean 2.4, median 2); the small median reflects the prevalence of single-argument utility functions in the inventory, while the long tail (up to 19 parameters) comes from complex configuration and validation tools. The parameter minimum of zero corresponds to no-argument sentinel tools used in optional-enrichment and loop-like patterns. These structural properties collectively ensure that each of the four evaluation metrics — tool selection, parameter structure, sequence accuracy, and query coverage — is non-trivially exercised across the record pool.

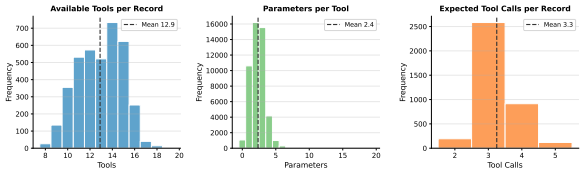

Figure 7: Frequency distributions of (left) available tools per record, (centre) parameters per tool, and (right) expected tool calls per record. Dashed vertical lines mark the mean of each distribution. Tool inventory is tightly concentrated around 13 (range 8-19); parameter counts are right-skewed with a median of 2; expected call-sequence length ranges from 2 to 5.

Quantity	Min	Max	Mean	Median
Available tools per record	8	19	12.9	13.0
Parameters per tool	0	19	2.4	2.0
Expected tool calls per record	2	5	3.3	3.0

Table 8: Structural statistics of the 3,808 Agent-JudgeBench records. All quantities are computed over unique records (collapsing difficulty variants). *Available tools* is the size of the tool inventory exposed to the generator per record. *Parameters per tool* counts typed argument fields in each tool's JSON schema. *Expected tool calls* is the length of the ground-truth execution trace.

Generator	Easy	Medium	Hard	Total	Max	Rate
Llama-3.3-70B	3,765	3,771	3,764	11,300	11,424	98.9%
Llama-3.1-8B	3,707	3,698	3,606	11,011	11,424	96.4%
Qwen3-32B	3,271	3,421	3,548	10,240	11,424	89.6%
SmolLM3-3B	3,300	3,042	3,299	9,641	11,424	84.4%
GPT-5.4	3,807	3,802	3,807	11,416	11,424	99.9%
Total	**17,850**	**17,734**	**18,024**	**53,608**	**57,120**	**93.8%**

Total: $53{,}608 \times 6$ judges $\times$ 2 conditions $= 643{,}296$ *evaluation instances (theoretical max);* 321,648 *unique (generator, judge, difficulty, record) tuples, each evaluated under both with GT and without GT conditions.*

Table 9: Number of records with valid LLM-judge verdicts per generator and difficulty tier. *Theoretical max* per generator is $3{,}808 \times 3 = 11{,}424$ (three difficulty variants of each base record). Judged records are identical across all six LLM judges for a given (generator, difficulty) cell. Multiplying the *Total* column by 6 judges $\times$ 2 conditions yields the full evaluation instance count per generator.

C Evaluation Corpus Statistics

Table 9 reports the exact number of records that received valid LLM-judge verdicts per (generator, difficulty) cell, after excluding records where the generator produced an unparseable or empty tool-call sequence. The base dataset contains 3,808 unique records; each is rewritten into three difficulty variants, giving a theoretical maximum of $3{,}808 \times 3 = 11{,}424$ generator inputs per generator. The final column shows the success rate relative to this theoretical maximum.

Sources of attrition. The overall success rate is 93.8% (53,608 of 57,120). Attrition arises from two distinct sources at different pipeline stages:

- **Unparseable generator output** ($\approx$5.7% of inputs, weighted average): the generator produces a response that cannot be decoded as a valid JSON tool-call list. This is the dominant source of attrition, concentrated on SmolLM3-

3B (-15.6%) and Qwen3-32B (-10.4%), and more prevalent at medium difficulty for SmolLM3-3B (medium rate 79.8% vs. 86.6% on easy/hard). These failures reflect intrinsic model capability gaps on the structured output format; re-generating these records would produce the same failure pattern and was not pursued.

- **Persistent LLM-judge null verdicts** ($\approx 3.3\%$ of main-grid judge calls): after the initial run, records with null `overall llm alignment percentage` were identified after retrying twice (two independent rerun rounds. Records that remained null after both retries - $\approx 10{,}498$ (generator, judge, difficulty, record) tuples in the main grid - were permanently excluded. These persistent nulls are concentrated on the longest GPT-5.4 and Llama-3.3-70B generator outputs, which push near the context limits of certain judge endpoints (particularly Claude Sonnet 4.0), causing consistent response truncation or malformed JSON. A third retry round was not run because (a) two retries had already demonstrated a $<5\%$ recovery rate for persistently null records, making further attempts cost-prohibitive, and (b) the persistent nulls are distributed uniformly across DAG topologies and difficulty levels, giving no reason to expect systematic bias in the excluded records.

D Bootstrap Confidence Intervals for Table 2

All alignment percentages in Table 2 are means over $N_{g,d}$ per-record scores. To quantify uncertainty, we compute 95% bootstrap confidence intervals (n=2,000 stratified resamples) for every (generator, judge, difficulty, condition) cell. Table 10 reports the full CI matrix for the Llama-3.3-70B generator. The same bootstrap analysis applied to all five generators yields CIs ≤ 0.3 pp half-width throughout (the remaining matrices are in the supplementary code release), confirming that the point estimates in Table 2 are reliable and that every monotone difficulty effect is statistically robust. CIs for the GT lift values (Eq. 5b) appear in Section 4.2; all six lift CIs are non-overlapping across the positive-vs-negative divide.

E Difficulty Degradation and Generator Accuracy

Figure 8 shows mean judge alignment as a function of query difficulty, averaged across all six judges and five generators. Table 11 reports generator programmatic accuracy $\bar{p}$ (%) on AgentJudgeBench, averaged across the four metrics and six DAG topologies.

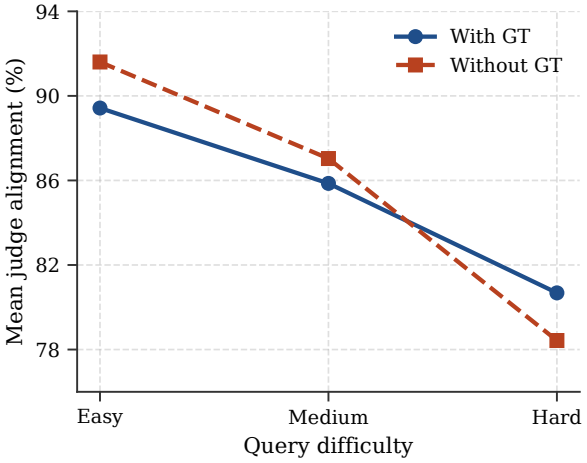

Figure 8: Mean judge alignment as a function of query difficulty, averaged across all six judges and five generators. The without GT degradation slope is roughly $1.5\times$ steeper than the GT slope across all generators.

F Programmatic Judge Validation

This appendix reports a human annotation study validating the programmatic scorer against independent human judgement. We sampled 120 *hard*-difficulty records stratified across all six DAG topologies (20 per topology). For each record, annotators were shown the user query, available tool schemas, generated tool calls, ground-truth expected tool calls, and the programmatic scores. For each of the four metrics they were asked to *agree* or *disagree* with the programmatic score and provide a brief justification, yielding $120 \times 4 = 480$ metric-level verdicts. Records were stratified to over-represent cases where at least one programmatic score is below 1.0 (14 per topology) alongside perfect-score records (6 per topology), ensuring annotators encountered the full range of difficulty. Annotators were not shown the paper's hypotheses or the LLM judge outputs prior to annotation.

The 92.7% overall agreement across 480 metric-level verdicts confirms the programmatic scorer as a reliable reference signal for the alignment metric

Judge	Easy		Medium		Hard	
	GT	No GT	GT	No GT	GT	No GT
GPT-5.4	91.4 [91.0,91.7]	95.2 [94.8,95.6]	87.4 [87.0,87.8]	91.7 [91.2,92.2]	83.0 [82.6,83.4]	84.6 [83.9,85.2]
Claude Sonnet 4.0	93.3 [93.0,93.7]	94.6 [94.3,95.0]	89.9 [89.5,90.4]	90.7 [90.2,91.2]	83.7 [83.1,84.2]	83.3 [82.6,83.9]
Gemini-2.5-Pro	90.5 [90.0,91.0]	95.7 [95.4,96.1]	86.5 [86.0,87.0]	92.2 [91.7,92.7]	82.2 [81.6,82.7]	84.6 [83.9,85.2]
QwQ-32B	96.6 [96.3,96.9]	95.6 [95.2,95.9]	93.6 [93.2,94.0]	92.1 [91.6,92.5]	87.5 [86.9,88.0]	84.4 [83.7,85.1]
GPT-OSS-20B	89.0 [88.7,89.3]	94.5 [94.1,94.9]	87.2 [86.9,87.5]	91.2 [90.7,91.7]	83.9 [83.5,84.2]	84.0 [83.3,84.7]
GPT-OSS-120B	96.0 [95.7,96.3]	95.5 [95.2,95.9]	93.0 [92.6,93.4]	92.1 [91.7,92.6]	87.2 [86.7,87.8]	84.5 [83.8,85.2]

Table 10: Bootstrap 95% confidence intervals for alignment (%) on the Llama-3.3-70B-Instruct generator. Format: mean [lo, hi].

Generator	Easy	Med.	Hard
Llama-3.3-70B	96.1	92.5	84.5
Qwen3-32B	93.7	88.5	80.0
Llama-3.1-8B	89.3	85.2	74.4
GPT-5.4	82.0	79.4	72.9
SmolLM3-3B	83.5	77.2	65.5

Table 11: Generator programmatic accuracy $\bar{p}$ (%) on AgentJudgeBench, averaged across the four metrics and six DAG topologies. Generators are ranked by hard-difficulty accuracy.

Metric	Agree	Disagree	Agreement %
Tool Selection	111	9	92.5
Parameter Structure	99	21	82.5
Sequence Accuracy	117	3	97.5
Query Coverage	118	2	98.3
Overall	**445**	**35**	**92.7**
DAG Type		**Agree / 80**	**%**
Linear		70	87.5
Fan-Out		77	96.2
Fan-In		75	93.8
Diamond		77	96.2
Optional Enrichment		68	85.0
Loop-Like		78	97.5

Table 12: Human annotator agreement with the programmatic scorer across 120 hard-difficulty records (n=480 metric-level verdicts), stratified at 20 records per DAG topology.

of Section 3.2. Agreement is not uniform across metrics: *parameter structure* is the weakest axis (82.5%, 21 disagreements), while *sequence accuracy* and *query coverage* are near-perfect (97.5% and 98.3%). Inspection of the 21 parameter-structure disagreements reveals two error modes. The dominant one (≈16 of 21 cases) is a *schema-vs.-GT mismatch*: the model includes additional parameter keys that are valid per the available-tool JSON schema but absent from the specific ground-truth trace; the programmatic scorer penalises these extras (per Eq. 2), whereas human annotators treat them as correct given the schema. The remaining 5 disagreements involve structurally matching parameters that the scorer under-penalises due to partial-match rounding. This pattern is precisely the semantic-equivalence gap described in Appendix 6, and it bounds the scope of any mis-alignment attributable to scorer noise: 74.1% of metric-level mismatches between the programmatic and LLM judges fall on tool selection, sequence accuracy, and coverage - dimensions where structural and semantic correctness largely coincide.

On the tool-selection axis (9 disagreements), two failure modes emerge: semantic equivalence between tools with different names (e.g., `validate_jurisdictional_limits` vs. `assess_jurisdictional_limits`), and reasonable extra tools that the scorer penalises as out-of-reference. The per-DAG pattern is consistent with known topology difficulty: *optional_enrichment* has the lowest agreement (85.0%), reflecting the inherent ambiguity in judging whether optional branches were correctly included or omitted - the same ambiguity that makes this topology moderately hard in the main judge evaluation (Table 23).

Judge alignment against human verdicts. The 120 annotated records also allow measuring how closely each LLM judge's with GT verdict matches human judgement, where records with annotator disagreement (n=35) are scored using the human-corrected value rather than the programmatic score. Table 13 compares each judge's programmatic-reference alignment (primary metric) against their human-verdict alignment on the same 120 hard records.

All six judges are systematically closer to the programmatic scorer than to human annotators (mean $\Delta = -6.9$ pp). GPT-OSS-120B is the

Judge	Prog.	Human	Δ
GPT-5.4	78.7	71.2	-7.5
Claude Sonnet 4.0	78.1	74.5	-3.7
Gemini-2.5-Pro	78.3	68.0	-10.2
QwQ-32B	82.9	74.3	-8.6
GPT-OSS-20B	79.0	70.2	-8.8
GPT-OSS-120B	82.2	79.4	-2.8

Table 13: Judge alignment (%, with GT) on 120 hard records: vs. programmatic reference and vs. human verdicts (35 corrected). Δ = human $-$ programmatic. Smaller $|\Delta|$ is better.

most human-aligned judge ($\Delta = -2.8$ pp), while Gemini-2.5-Pro diverges most ($\Delta = -10.2$ pp). The negative Δ reflects that human annotators are more lenient on hard examples (particularly on sequence accuracy and parameter structure), while both the programmatic scorer and LLM judges apply stricter structural matching. The judge rankings under human-verdict alignment differ meaningfully from programmatic-reference rankings (Spearman $\rho = 0.26$): QwQ-32B, which leads the programmatic leaderboard, drops to 4th under human alignment, while GPT-OSS-120B tops both orderings. This suggests that the programmatic scorer's strictness inflates QwQ's apparent advantage; GPT-OSS-120B is the most robust judge across both evaluation bases. Practitioners preferring alignment with human intuition should favour GPT-OSS-120B, regardless of whether programmatic or human verdicts are used as the reference.

G Over-Anchoring Case Studies

This appendix presents three representative records from the 1,322 over-anchoring instances identified on Gemini-2.5-Pro hard-difficulty outputs (records where $p_r^m < 1.0$, $\ell_{\text{GT}}^m > p_r^m$, and $\ell_{\text{without GT}}^m \leq p_r^m + 0.1$ on at least one metric). Cases 1-2 document the dominant *sequence-accuracy anchor* mechanism; Case 3 documents the secondary *coverage anchor*. In each case the judge's with GT justification contains explicit reference to the GT trace ("sequence is logical given the reference"), whereas the without GT justification independently identifies the structural flaw the programmatic judge also penalises.

Case 1 — Sequence anchor, microbial-growth linear DAG. Query: *"Got this cheese with 250 CFU/g initial bugs, left at 20°C for 6.5 hours - need the growth rate, then a safe storage temperature for 120 hours, then the shelf life at that temp."* **Generated calls:** `calculate_microbial_growth_rate` $\rightarrow$ `estimate_optimal_storage_temperature` (2 calls; ground truth requires a third: `predict_shelf_life`). **Programmatic:** seq $= 0.67$ (partial match; missing third call). **Gemini with GT:** seq $= 1.0$ - *"The sequence of the selected tools is logical, with the growth rate calculation correctly preceding the temperature estimation."* The judge evaluates the two-call prefix against the GT prefix and deems it correct, ignoring the missing third step. **Gemini without GT:** seq $= 0.5$ - *"The plan includes a redundant, identical tool call, which is a minor structural flaw in the sequence."* The judge invents a different critique rather than catching the missing call, but does not award full credit. **Mechanism:** GT exposure causes the judge to evaluate only the presented steps against the reference rather than checking for completeness.

Case 2 — Sequence anchor, cloud-budget linear DAG. Query: *"Cloud bill hit $4,800 over 15 days vs. $250/day normal. 10 days left, $8,000 budget. What's going on and will we blow the budget?"*
Generated calls:`calculate_budget_deviati--on_risk` (`predicted_trend=1.93`)$\rightarrow$`forecast_budget_usage`.The first call consumes a `predicted_-trend` argument that has not been computed by any prior step. **Programmatic:** seq $= 0.33$. **Gemini with GT:** seq $= 1.0$ - *"The two selected tools are independent and can be executed in any order."* The GT reference shows two calls and the judge rationalises independence; it does not notice the unresolved input dependency. **Gemini without GT:** seq $= 0.0$ - *"The first tool call depends on a* `predicted_trend` *input that is not available from the query or generated by a prior step."* Without the GT to anchor to, the judge correctly identifies the data-dependency violation. **Mechanism:** GT anchoring suppresses dependency-checking; the judge substitutes a plausibility heuristic ("independent tools can run in any order") for structural verification.

Case 3 — Coverage anchor, power-grid linear DAG. Query: *"We're pulling 8,450 kW of 12,000 kW, expecting +1,200 kW, running hot for 5.3 h vs. 4 h threshold. Is everything okay?"* **Generated calls:** Three analysis tools (load-duration risk, grid-fatigue risk, capacity headroom). Ground truth uses two; the extra call is `estimate_grid_fatigue_risk`. **Programmatic:**

cov $= 0.67$ (extra call penalised). **Gemini with GT:** cov $= 1.0$ - *"The plan selects tools that address all numerical data points provided in the user's query."* Seeing the GT calls, the judge credits the prediction for covering the query rather than checking against the reference. **Gemini without GT:** cov $= 0.5$ - *"The plan analyzes all data points but lacks a final synthesis tool to answer whether to be worried."* Without GT, the judge applies an independent completeness check and finds partial coverage. **Mechanism:** GT presence triggers credit-by-association; the judge attributes coverage to the prediction by observing that the GT is complete, rather than evaluating the prediction independently.

H C3 Corrupted-GT Control: Full Results

Table 1 reports per-difficulty alignment for both judges under standard GT, without GT, and the corrupted-GT (C3) condition on the Llama-3.3-70B generator. The C3 condition replaces the judge's reference with a randomly sampled GT from a different record of the same DAG topology. The key finding is the judge-level split: Gemini-2.5-Pro with corrupted GT matches its standard GT alignment within 0-1.8 pp across all difficulties, while QwQ-32B with corrupted GT matches its without GT alignment within 0.2 pp. This confirms the mechanism described in Section 4.2: Gemini anchors to any reference block regardless of content; QwQ exercises independent reasoning when the reference is incoherent.

I Notation

Table 14 summarises the symbols used throughout the paper, in the order in which they appear.

J Judge LLMs Comparison

Table 15 compares AgentJudgeBench against the seven existing systems that deploy LLM judges for agentic tool-calling evaluation, across four dimensions: per-metric decomposition, difficulty variation, ground-truth ablation, and use of a deterministic programmatic reference. No prior system addresses more than one of these dimensions; AgentJudgeBench is the first to provide all four.

K Generator Models

Table 16 lists the five generator models used to produce tool-calling outputs. Models span four capability tiers - small open-source (3B), mid-scale open (8B), large open (32B-70B), and frontier closed (GPT-5.4) - ensuring the judge evaluation covers a representative range of output quality. All generators are decoded at temperature 0 with the model's native function-calling prompt.

L Rewriting-Preservation Validation Study

This appendix reports the full results of the meta-judge validation study summarised in §3.1. *Goal:* verify, independently of the programmatic judge, that our difficulty-controlled rewrites (a) preserve the ground-truth tool-call sequence and (b) are strictly harder (less explicit) than their predecessor.

Protocol. We stratify-sampled 198 triplets (33 per DAG topology) from the 3,808 record pool and asked each of three frontier LLM meta-judges (Claude Sonnet 4.0, GPT-5.4, Gemini-2.5-Pro) to emit a single JSON verdict per triplet, answering four Boolean questions: (i) does the medium query admit the same ground-truth tool calls as the easy query?; (ii) same for hard?; (iii) is the medium rewrite strictly harder than easy (less explicit in at least one of parameter names, numeric values, or tool intents)?; (iv) is hard strictly harder than medium? All three meta-judges receive identical prompt scaffolding and decoding parameters. The meta-judges are *not* shown any generator's tool-call prediction - they reason only over the three query variants, the shared available-tools schema, and the shared ground-truth tool-call sequence.

Per-judge verdict rates. Table 17 reports the fraction of triplets for which each meta-judge answered "yes" to each question. Task-preservation rates are uniformly high (77.8%-100% across all judges and both rewrite pairs); the strict-hardening rates are also high for hard-versus-medium ($\geq$ 95.5%) but lower and more judge-dependent for medium-versus-easy, driven by GPT-5.4's stricter interpretation of what constitutes a genuine loss of explicitness.

Unanimous agreement. Table 18 reports, per criterion, the number of triplets on which the three meta-judges unanimously agreed (all-yes or all-no) versus gave mixed verdicts. Crucially, *no triplet receives a unanimous "no" verdict* on either task-preservation question, and every single triplet is unanimously judged as hard-strictly-harder-than-medium. The only criterion with sub-60% unanim-

Symbol	Definition
$\mathcal{G}$	Set of generator models whose tool-call sequences are judged; $\lvert\mathcal{G}\rvert = 5$.
$\mathcal{J}$	Set of LLM judges evaluated; $\lvert\mathcal{J}\rvert = 6$.
$\mathcal{D}$	Difficulty levels of the query rewrites, $\mathcal{D} = \{\text{easy}, \text{medium}, \text{hard}\}$.
$g,\ j,\ d$	Individual generator, judge, and difficulty, with $g \in \mathcal{G}$, $j \in \mathcal{J}$, $d \in \mathcal{D}$.
c	Judge condition, $c \in \{\text{GT}, \text{without GT}\}$: whether the prompt includes the ground-truth tool calls.
r	Individual record (one query, its tool schemas, ground-truth and predicted tool calls).
M	Number of evaluation metrics; $M = 4$ (tool selection, parameter structure, sequence accuracy, query coverage).
$N_{g,d}$	Record count in the (g, d) configuration after filtering unparseable generator outputs.
$p_r \in [0, 1]$	Deterministic programmatic scorer's verdict on record r (reference signal for alignment).
$\ell_{j,r,c} \in \{0, 0.5, 1\}$	LLM judge j's verdict on record r under condition c.
$\text{align}(j, g, d, c)$	Mean per-record alignment percentage (Eq. 5a).
$\overline{\text{align}}_c(j)$	Alignment of judge j under condition c, averaged over all (g, d) configurations.
$\text{lift}(j)$	Ground-truth lift of judge j: $\overline{\text{align}}_{\text{GT}}(j) - \overline{\text{align}}_{\text{without GT}}(j)$ (Eq. 5b).
$\Delta(\text{hard}-\text{easy})$	Difficulty-induced degradation: $\text{align}(\cdot, \cdot, \text{hard}, \cdot) - \text{align}(\cdot, \cdot, \text{easy}, \cdot)$, in percentage points (pp).

Table 14: Notation used in the paper.

System	Judge Model	Per-Metric	Difficulty	GT Abl.	Prog. Ref.
ToolEval (Qin et al., 2024)	ChatGPT	✗	✗	✗	✗
StableToolBench (Guo et al., 2024)	GPT-4-turbo	✗	✗	✗	✗
MCP-AgentBench (Guo et al., 2025)	LLM + Rules	✗	✗	✗	✗
GeoBenchX (Krechetova and Kochedykov, 2025)	3-Judge Panel	✗	✗	✗	✗
Agent-as-a-Judge (Zhuge et al., 2024)	Agent	✗	✗	✗	✗
Auto-Eval Judge (Bhonsle et al., 2025)	GPT-4o	✗	✗	✗	✗
ToolSandbox (Lu et al., 2024)	*Replaced*	N/A	✗	N/A	✓
AgentJudgeBench (Ours)	**Multi-scale LLMs (3B → Frontier)**	✓	✓	✓	✓

Table 15: Comparison of systems that deploy LLM judges for agentic tool-calling evaluation. **Judge Model**: LLM used as judge. **Per-Metric**: whether evaluation is decomposed into fine-grained dimensions. **Difficulty**: whether tasks span multiple difficulty tiers. **GT Abl.**: whether the effect of ground-truth availability is studied. **Prog. Ref.**: whether a deterministic evaluator is used as a bias-free baseline.

Short name	HF / Server identifier	Size
Llama-3.3-70B-Instruct	`meta-llama/Llama-3.3-70B-Instruct`	70B
Qwen3-32B	`Qwen/Qwen3-32B`	32B
Llama-3.1-8B-Instruct	`meta-llama/Llama-3.1-8B-Instruct`	8B
SmolLM3-3B	`HuggingFaceTB/SmolLM3-3B`	3B
GPT-5.4[†]	Azure OpenAI / `gpt-5.4`, api-ver `2025-04-01-preview`, accessed Apr 2026	-

[†]"GPT-5.4" is the internal designation for the GPT-5 preview variant deployed on Azure OpenAI as of April 2026; not an officially published model version string. Raw per-record outputs are included in the supplementary data release.

Table 16: Generator models used to produce tool-calling outputs. All generators use temperature $= 0$ with their native function-calling prompt.

ity is medium-strictly-harder-than-easy, consistent with the per-judge analysis above: annotators disagree at the margin on whether medium constitutes a genuine hardening as opposed to a paraphrase.

Interpretation. The validation supports the stronger of our two design claims (task preservation) and partially supports the weaker one (strict hardening). The task-preservation result means that any alignment degradation observed in Sec-

Criterion	Claude Sonnet 4.0	GPT-5.4	Gemini-2.5-Pro
Medium preserves ground-truth tool calls	100.0%	94.9%	100.0%
Hard preserves ground-truth tool calls	100.0%	77.8%	97.5%
Medium strictly harder than easy	99.5%	62.1%	84.8%
Hard strictly harder than medium	100.0%	97.5%	95.5%

Table 17: Per-judge "yes"-rate on the rewrite-validation criteria over $n = 198$ stratified triplets.

Criterion	All-yes	All-no	Mixed	Unanimity%
Medium preserves ground-truth tool calls	188	0	10	94.9%
Hard preserves ground-truth tool calls	152	0	46	76.8%
Medium strictly harder than easy	114	1	83	58.1%
Hard strictly harder than medium	186	0	12	93.9%

Table 18: Unanimity of the three meta-judges on each criterion ($n = 198$).

Short name	Provider / Identifier	Size
GPT-5.4[†]	Azure OpenAI / `gpt-5.4`, api-ver `2025-04-01-preview`, accessed Apr 2026	
Claude Sonnet 4.0	Anthropic proxy / `claude-sonnet-4-5-20250929-v1:0`, accessed Apr 2026	
Gemini-2.5-Pro	Vertex AI proxy / `gemini-2.5-pro-preview-05-06`, accessed Apr 2026	
QwQ-32B	vLLM / `Qwen/QwQ-32B`, `enable_thinking=true`	32B
GPT-OSS-20B	vLLM / `openai/gpt-oss-20b`	20B
GPT-OSS-120B	vLLM / `openai/gpt-oss-120b`	120B

[†]"GPT-5.4" is the internal designation for the GPT-5 preview variant deployed on Azure OpenAI as of April 2026; it is not an officially published model version string.

Table 19: LLM judge configurations evaluated in this paper. All judges consume the same prompt scaffold (Appendix T) and produce the same JSON output schema.

tion 4 between easy and hard difficulty conditions is *not* attributable to the rewrites silently changing the underlying task: even the most conservative meta-judge (GPT-5.4, which tends to reject rewrites for minor interpretative drift) finds task-preservation intact on 77.8% of hard rewrites, and the unanimous-no rate is zero. The strict-hardening result for medium-vs-easy is a weaker claim; on that axis our rewriting process achieves a hardening GPT-5.4 would accept only about two-thirds of the time, suggesting that a subset of our medium rewrites are better characterised as paraphrases than as genuine hardenings. This is an informative finding on its own - it suggests the paper's "easy → medium → hard" degradation curves may partially conflate paraphrase-robustness with genuine difficulty-robustness - and is a natural target for a follow-up rewriting-pipeline revision.

M LLM Judge Configurations

Table 19 lists the six LLM judge configurations evaluated in this paper. The set includes large open models (20B-120B), a reasoning-enabled open model (QwQ-32B), and frontier closed models (GPT-5.4, Claude Sonnet 4.0, Gemini-2.5-Pro). QwQ-32B is invoked with `enable_thinking=true` (chain-of-thought enabled); all other judges use greedy or near-greedy decoding.

Decoding temperature. vLLM-hosted open judges (QwQ-32B, GPT-OSS-20B, GPT-OSS-120B) use temperature $= 0.15$. Frontier judges (GPT-5.4, Claude Sonnet 4.0, Gemini-2.5-Pro) were accessed via provider APIs that did not support temperature $= 0$ at time of evaluation; they use the provider's recommended default (≤ 1.0). This asymmetry is an implementation constraint, not a design choice. To assess its impact: our temperature sensitivity study (§4.2) shows alignment varies by at most 0.6 pp across $T \in \{0.3, 0.7, 1.0\}$,

indicating that the inter-condition temperature gap is unlikely to materially confound the cross-judge comparisons. A broader evaluation with harmonised temperatures across all judges is planned for a future revision.

N Per-Metric Breakdown

Table 20 reports per-metric alignment under both conditions, averaged across all 12 (g, d) configurations. Under with GT, sequence accuracy shows the widest inter-judge spread: Gemini-2.5-Pro scores 64.2% while GPT-OSS-20B reaches 86.4%, a gap of over 22 pp. Tool selection reveals a scale effect: GPT-OSS-20B drops to 70.7%, nearly 18 pp below GPT-OSS-120B (88.7%). Parameter structure and query coverage are uniformly high (86-94%). Under without-GT, all variation collapses: the widest spread on any metric is 3.0 pp, confirming that the "default to 1.0" rubric erases capability differences.

	With GT				Without GT			
Judge	**tool**	**param**	**seq**	**cov**	**tool**	**param**	**seq**	**cov**
GPT-5.4	83.8	**90.0**	72.3	**94.0**	**84.7**	89.8	**80.7**	91.0
Claude Sonnet 4.0	86.4	86.5	78.0	92.7	83.2	88.2	80.1	91.3
Gemini-2.5-Pro	82.6	89.4	64.2	93.3	82.9	89.5	80.4	**92.3**
QwQ-32B	88.5	89.8	84.8	93.9	82.0	**89.9**	79.9	91.5
GPT-OSS-20B	70.7	89.0	**86.4**	92.5	81.7	89.2	79.5	89.5
GPT-OSS-120B	**88.7**	89.7	82.9	93.5	82.1	**89.9**	79.4	91.6

Table 20: Per-metric match score $\mu^{m}_{j,r,c} \times 100$, averaged across all (g, d) configurations. Bold marks the best value per column.

	Easy		Med.		Hard	
Temp.	**GT**	**w/o**	**GT**	**w/o**	**GT**	**w/o**
0.3	**96.4**	**95.1**	**93.5**	**91.8**	**87.5**	**84.0**
0.7	**96.4**	95.2	93.4	91.7	87.2	84.0
1.0	96.1	95.2	92.9	91.2	87.0	83.9
Spread	0.3	0.1	0.6	0.6	0.5	0.1

Table 21: Judge temperature sensitivity: per-difficulty alignment (%) for Qwen3-32B on Llama-3.3-70B.

O Supplementary Numerical Tables

Judge Temperature: Per-Difficulty Results

Table 21 provides the full per-difficulty numerical results for the temperature sensitivity study (Section 4.2). Alignment variance across temperatures is $\leq$ 0.6 pp on every (difficulty, condition) slice, confirming that Qwen3-32B judge behaviour is insensitive to sampling stochasticity.

Chain-of-Thought Reasoning: Full Per-Generator Grid

Table 22 reports the full $4 \times 3 \times 2$ grid for the QwQ-32B reasoning study (Section 4.2). No cell shows a difference exceeding 0.3 pp. The pattern is uniform across generator quality tiers: even on SmolLM3-3B, where generator errors are most frequent and reasoning might be expected to help the judge distinguish correct from incorrect calls, the thinking trace adds nothing.

P Per-Topology Breakdown

Table 23 reports per-topology alignment under with GT. QwQ-32B leads on five of six topologies; GPT-OSS-120B leads on fan_out (93.9%) and is a close second elsewhere. Gemini-2.5-Pro is consistently weakest, trailing QwQ-32B by 5-8 pp. The gap between the easiest (fan_out, ∼93%) and hardest (fan_in, ∼83%) topologies is roughly 10 pp, comparable to the easy-to-hard difficulty degradation in Table 2. Fan-out's advantage is intuitive: each parallel branch can be verified independently, whereas fan-in and loop-like topologies require tracking cross-branch dependencies.

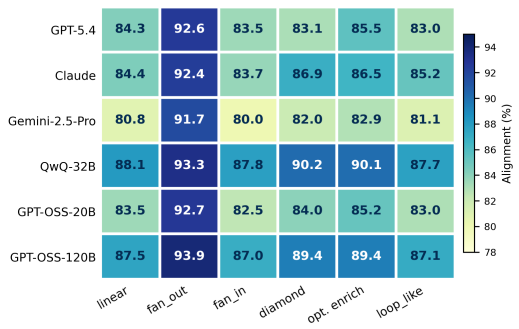

Figure 9: Judge alignment (%) by DAG topology (with GT), visualised as a heatmap. Fan-out consistently achieves the highest alignment across all judges; fan-in and loop-like are the hardest. The ordering is judge-independent, indicating that DAG structural complexity is an intrinsic difficulty signal.

Q Inter-Judge Confusion Matrices

Table 10 reports per-verdict counts for the highest-agreement pair (QwQ-32B×GPT-OSS-120B, $\kappa = 0.606$) and the lowest-agreement pair (Claude Sonnet 4.0×GPT-OSS-20B, $\kappa = 0.225$) under with GT. Disagreement concentrates almost entirely at the partial-credit boundary: 94.7% and 96.8% of off-diagonal entries involve at least one 0.5 verdict. The dominant off-diagonal cell for the low-

Generator	Judge	Easy		Medium		Hard	
		GT	No GT	GT	No GT	GT	No GT
Llama-3.3-70B	Thinking on	96.6	95.6	93.6	92.1	87.5	84.4
	Thinking off	96.5	95.6	93.6	92.0	87.3	84.4
	Δ	$+0.1$	0.0	0.0	$+0.1$	$+0.2$	0.0
Llama-3.1-8B	Thinking on	94.2	91.2	90.8	86.8	84.1	76.6
	Thinking off	94.3	91.4	90.8	86.8	83.9	76.7
	Δ	-0.1	-0.2	0.0	0.0	$+0.2$	-0.1
Qwen3-32B	Thinking on	95.0	93.9	90.9	89.1	84.7	81.1
	Thinking off	94.9	94.0	90.8	89.0	84.7	81.1
	Δ	$+0.1$	-0.1	$+0.1$	$+0.1$	0.0	0.0
SmolLM3-3B	Thinking on	90.0	86.7	84.8	80.3	77.2	69.7
	Thinking off	89.8	86.7	84.6	80.4	76.9	69.7
	Δ	$+0.2$	0.0	$+0.2$	-0.1	$+0.3$	0.0

Table 22: Chain-of-thought reasoning study: full per-generator alignment (%) for QwQ-32B with thinking on vs. off. Δ rows show thinking-on minus thinking-off in percentage points.

Judge	linear	fan_out	fan_in	diamond	opt. enrich	loop_like
GPT-5.4	84.3	92.6	83.5	83.1	85.5	83.0
Claude Sonnet 4.0	84.4	92.4	83.7	86.9	86.5	85.2
Gemini-2.5-Pro	80.8	91.7	80.0	82.0	82.9	81.1
QwQ-32B	**88.1**	93.3	**87.8**	**90.2**	**90.1**	**87.7**
GPT-OSS-20B	83.5	92.7	82.5	84.0	85.2	83.0
GPT-OSS-120B	87.5	**93.9**	87.0	89.4	89.4	87.1

Table 23: Judge alignment (with GT, %) by DAG topology. Bold marks the best judge per topology.

Judge	Std.	Binary	Δ	0.5 rate
GPT-OSS-20B	84.0	**97.0**	$+13.0$	26.5%
QwQ-32B	**87.6**	94.7	$+7.1$	15.2%
GPT-OSS-120B	87.5	95.4	$+7.9$	14.0%
Claude Sonnet 4.0[‡]	85.4	94.7	$+9.2$	18.6%
Gemini-2.5-Pro	81.7	90.0	$+8.3$	12.2%
GPT-5.4	84.8	**96.6**	$+11.9$	28.2%

Table 24: Binary vs. $\{0, 0.5, 1\}$ verdict scale: mean GT alignment (%) and judge rankings averaged over 15 (generator, difficulty) cells. Δ = Binary $-$ Standard.

agreement pair is Claude Sonnet 4.0$=1$ / GPT-OSS-20B$=0.5$ (18.2% of verdicts), indicating GPT-OSS-20B is systematically more conservative. Under without GT, both pairs converge as both judges default to 1.0: Claude Sonnet 4.0×OSS-20B agreement rises from 70.3% to 90.1%.

QwQ-32B × GPT-OSS-120B
($\kappa = 0.61$, Agreement 87.9%)

	0	0.5	1
0	**1.3**	0.8	0.3
0.5	1.3	**10.3**	4.9
1	0.4	4.4	**76.3**

Claude Sonnet 4.0 × GPT-OSS-20B
($\kappa = 0.23$, Agreement 70.3%)

	0	0.5	1
0	**0.7**	0.4	0.2
0.5	1.9	**9.7**	8.3
1	0.7	18.2	**60.0**

Figure 10: Verdict-level confusion matrices (% of N). Rows = judge 1, columns = judge 2. Diagonal entries are shaded.

Binary vs. {0, 0.5, 1} Verdict Scale

Table 24 compares judge rankings under the original three-level $\{0, 0.5, 1\}$ verdict scale vs. a binary collapse in which each 0.5 verdict is remapped to 0 if the programmatic reference is below 0.5, and 1 otherwise. Columns report mean GT alignment across all 15 (generator, difficulty) cells per judge, and the per-judge rate of issuing 0.5 verdicts.

The binary collapse does not inflate all scores uniformly: judges with high 0.5 rates (GPT-5.4: 28.2%, GPT-OSS-20B: 26.5%) gain disproportionately ($+11.9$ pp and $+13.0$ pp respectively). Under binary remapping, the 0.5 verdict is always counted as correct because it is mapped to the programmatic direction by construction. Judges that hedge on uncertain records therefore receive artificially inflated binary alignment, reshuffling the ranking from QwQ-32B $>$ GPT-OSS-120B to GPT-OSS-20B $>$ GPT-5.4. The Spearman rank correlation between standard and binary ranking is $\rho = 0.03$ ($p = 0.96$), indicating the rankings are essentially uncorrelated. This confirms that the 0.5 verdict is not random noise: it encodes directional uncertainty that is systematically correlated with judge

capacity and lost under binary collapse. We recommend retaining the three-level scale.

R Judge Score Stochasticity and Prompt Sensitivity

Repeated-run variability

A natural concern for any LLM-based evaluation pipeline is whether the judge's score for a given record is stable across independent runs with the same prompt. All frontier judges (GPT-5.4, Claude Sonnet 4.5, Gemini-2.5-Pro) were called at near-default temperatures (≤ 1.0, as noted in Appendix M), meaning a small amount of stochasticity is inherent to each call. Our temperature sensitivity study (Section 4.2 and Appendix O) indirectly characterises this: Qwen3-32B alignment varies by at most 0.6 pp across $T \in \{0.3, 0.7, 1.0\}$ on 3,771 records, providing an upper bound on within-judge run-to-run variance. We therefore expect configuration-level alignment estimates (averaged over $N_{g,d} \geq 3{,}764$ records) to be highly stable; a 0.6 pp spread at record level contracts to $\ll$0.1 pp at the configuration mean by the central limit theorem.

Prompt variation sensitivity

Our prompt ablation (Section 4.2, Figure 5) shows that switching from the structured per-metric JSON rubric to a free-form one-sentence instruction drops GT alignment by 4.8–6.5 pp for Qwen3-32B on the Llama-3.3-70B generator. This > 5 pp gap dwarfs the within-prompt stochasticity bound and confirms that *prompt structure is the dominant source of judge-score variability*, not sampling noise. Practitioners adapting these prompts should expect similar sensitivity: minor wording changes (e.g., removing the "default to 1.0" instruction) can shift alignment by 1–5 pp, as demonstrated by the C2 ablation (Table 4), while major structural changes (free-form vs. per-metric rubric) produce first-order effects. Standalone prompt texts for all four variants used in this study are provided in Appendix T.

S Related Work Survey

A rigorous per-judge repeated-run study — running each judge twice on a stratified subset and computing per-record verdict-flip rates — is planned for a future revision. Such a study would directly quantify the fraction of borderline verdicts driven by stochasticity rather than systematic judge disagreement, and would allow score variance to be separated from the inter-judge disagreement reported in Table 3.

T Prompts

This appendix reproduces, verbatim, every prompt used in the evaluation pipeline: (i) the *generator* prompt (Appendix T.1) that instructs each generator $g \in \mathcal{G}$ to emit a tool-call sequence, (ii) the *with GT judge* prompt (Appendix T.2) sent to every LLM judge $j \in \mathcal{J}$ under the GT condition, and (iii) the *without GT judge* prompt (Appendix T.3), identical to (ii) save for the omission of the EXPECTED TOOL CALLS block. Decoding parameters are held constant across all prompts and all (g, j, d, c) configurations.

T.1 Generator prompt.

SYSTEM

```
You are a tool-call generation model
participating in a BENCHMARK EVALUATION.
This is a SIMULATION. The tools listed are
HYPOTHETICAL. You are NOT being asked to
actually execute anything. You are being
evaluated on your ability to select and format
the correct tool calls given a query and a tool
schema.
Your ONLY job:
Given the user message and available tools,
output the correct JSON array of tool calls.
Nothing else.
Rules:
- Output ONLY a raw JSON array. No text, no
markdown, no explanation.
- If no tools apply, output: []
- Use EXACTLY the format specified below.
{system_prompt}
Available Tools:
{available_tools}
```

USER

```
{user_message}
```

T.2 Judge prompt - with ground truth (GT condition).

USER

```
You are a STRICT STRUCTURAL EVALUATOR for AI
agent tool-calling plans.
IMPORTANT CONTEXT (NON-NEGOTIABLE):
- The evaluated model is SINGLE-SHOT and
STATELESS.
- The model does NOT observe tool execution or
tool outputs.
- The model ONLY plans tool calls.
- ALL evaluation MUST be STATIC and STRUCTURAL.
- Assume the programmatic judge is the ground
truth.
ABSOLUTELY DO NOT:
```

Name	Year	Description
Agentic Data & Tool-Calling Benchmarks		
Self-Instruct (Wang et al., 2023b)	2022	LLM-only instruction synthesis; seed for synthetic data pipelines.
Mind2Web (Deng et al., 2023)	2023	Human-annotated user trajectories over real websites.
AgentBench (Liu et al., 2024)	2023	Multi-domain agentic benchmark across reasoning and interaction settings.
WebArena (Zhou et al., 2024)	2024	Realistic web environment with functional sites and long-horizon tasks.
WorkArena (Drouin et al., 2024)	2024	Evaluation on production websites for task-solving agents.
AppWorld (Trivedi et al., 2024)	2024	Multi-app environment for interactive code-based agent workflows.
TaskBench (Shen et al., 2024)	2024	Tool-graph-based benchmark for task decomposition and tool selection.
AgentTrek (Xu et al., 2025a)	2025	Converts web tutorials into executable agent trajectories.
BFCL (Patil et al., 2025)	2025	Leaderboard for single-turn function-calling accuracy; AST-based scoring.
FuncBenchGen (Maekawa et al., 2025)	2025	DAG-based synthetic function-calling with controllable complexity.
τ-bench (Yao et al., 2024)	2024	Tool-agent-user interaction with simulated users and `pass^k` reliability metric.
LLM-as-Judge		
Zheng et al. (Zheng et al., 2023)	2023	MT-Bench; identified positional, verbosity, and self-enhancement biases.
JudgeBench (Tan et al., 2025)	2024	Judge evaluation on hard pairs with verifiable ground truth.
Arena-Hard-Auto (Li et al., 2024b)	2024	Automated pairwise judge evaluation; high human agreement.
PandaLM (Wang et al., 2024)	2024	Dedicated judge model trained for pairwise comparison.
Auto-J (Li et al., 2024a)	2024	Generalist judge model trained on diverse evaluation criteria.
JudgeLM (Zhu et al., 2023)	2023	Fine-tuned 7B–33B judges; characterises position, knowledge, and format biases.
Prometheus 2 (Kim et al., 2024)	2024	Open-weight evaluator with rubric-conditioned direct and pairwise assessment.
LLM Judges for Tool-Calling Evaluation		
ToolEval (Qin et al., 2024)	2023	ChatGPT as judge for pass rate on API trajectories.
StableToolBench (Guo et al., 2024)	2024	GPT-4-turbo as evaluator in a virtualized API environment.
ToolSandbox (Lu et al., 2024)	2024	Questioned LLM judge reliability; replaced with milestone-based programmatic scoring.
MCP-AgentBench (Guo et al., 2025)	2025	Hybrid rule-based and LLM judge for task-completion scoring.
GeoBenchX (Krechetova and Kochedykov, 2025)	2025	Three-judge panel for geospatial tool-use evaluation.
Agent-as-a-Judge (Zhuge et al., 2024)	2024	Agent evaluates another agent on DAG-structured development tasks.
Auto-Eval Judge (Bhonsle et al., 2025)	2025	Modular framework decomposing evaluation into checklist questions.

Table 25: Representative works across agentic benchmarks, LLM-as-judge methods, and LLM judges for tool-calling evaluation.

```
- Evaluate correctness of parameter VALUES
- Compare generated values to expected values
- Infer correctness from tool output logic
- Penalize placeholder, symbolic, or dummy values
- Penalize derived-value mismatches (e.g., 0.75 vs 0.005)
```

```
ONLY evaluate:
- Tool presence
- Tool names
- Parameter NAMES
- Parameter STRUCTURE (dict shape)
- Logical ordering of calls
- Coverage of user intent
```

If parameter NAMES match and the tool intent is satisfied, the parameter structure MUST be treated as correct.

–

Evaluation Metrics (PROGRAMMATIC-ALIGNED)
You must compute FOUR scores. Scores should default to **1.0 unless a clear structural violation exists**.

1. TOOL SELECTION ACCURACY. Evaluate only: required tools present, no clearly irrelevant tools. Ignore tool grouping and execution feasibility. Scoring: 1.0 all required/no irrelevant; 0.5 missing or clearly irrelevant; 0.0 plan fundamentally incorrect.

2. PARAMETER STRUCTURE ACCURACY (MOST IMPORTANT). Evaluate strictly: presence of required parameter names; arguments are a dictionary; parameter names logically belong to the tool. Do NOT evaluate parameter values, numeric ranges, threshold correctness, dependency resolution, or output-derived values. Rules: required parameter names present ⇒ structurally correct; optional parameters never reduce score; placeholder or symbolic values are always valid. Scoring: 1.0 correct names on all tools; 0.5 minor missing/extra; 0.0 required names missing for most tools.

3. SEQUENCE & DEPENDENCY ACCURACY. Does the order reflect logical planning? Are prerequisites placed earlier? Assume symbolic dependency handling is valid. Scoring: 1.0 logical; 0.5 minor ordering issue, intent preserved; 0.0 illogical or contradictory.

4. QUERY COVERAGE ACCURACY. Does the plan cover all parts of the user request? Are all sub-goals planned for? Scoring: 1.0 full coverage; 0.5 partial; 0.0 major intent missed.

–

OUTPUT FORMAT (STRICT). Return exactly one JSON object with the four metric fields, each containing {"accuracy": <float>, "justification": "<1 sentence>"}, plus an "overall_assessment" single-sentence summary.

Be conservative. Prefer 1.0 unless a **clear structural violation** exists. Judge structure, not semantics.

QUERY:
{user_message}

AVAILABLE TOOLS:
{available_tools}

GENERATED TOOL CALLS:
{generated_tool_calls}

EXPECTED TOOL CALLS:
{expected_responses}

T.3 Judge prompt - without ground truth (without GT condition).

User

You are a STRICT STRUCTURAL EVALUATOR for AI agent tool-calling plans.

IMPORTANT CONTEXT (NON-NEGOTIABLE):
- The evaluated model is SINGLE-SHOT and STATELESS.
- The model does NOT observe tool execution or tool outputs.
- The model ONLY plans tool calls.
- ALL evaluation MUST be STATIC and STRUCTURAL.
- There is NO expected tool plan available.

ABSOLUTELY DO NOT:
- Evaluate correctness of parameter VALUES
- Infer correctness from tool output logic
- Penalize placeholder, symbolic, or dummy values
- Penalize derived-value mismatches
- Penalize missing runtime-dependent values

ONLY evaluate using:
- The user QUERY
- AVAILABLE TOOLS schema
- GENERATED TOOL CALLS

Treat the AVAILABLE TOOLS as the ONLY source of truth.

–

Evaluation Metrics (STRUCTURAL ONLY)
Compute FOUR scores. Scores MUST default to **1.0 unless a clear structural violation exists**.

1. TOOL SELECTION ACCURACY. Are selected tools relevant to the query? Tools that clearly do not belong? Rules: if a tool plausibly helps answer the query, treat as valid; only penalize clearly irrelevant tools. Scoring: 1.0 all relevant; 0.5 one questionable/missing; 0.0 selection unrelated.

2. PARAMETER STRUCTURE ACCURACY (MOST IMPORTANT). Evaluate strictly: arguments exist and are dictionaries; parameter names exist in the tool schema; parameter names logically belong to the tool. Do NOT evaluate parameter values, types, ranges, thresholds, or symbolic references. Rules: matching names ⇒ correct; optional parameters never reduce score; extra parameters only penalize if clearly invalid. Scoring: 1.0 valid across all tools; 0.5 minor issues; 0.0 most tools incorrect.

3. SEQUENCE & DEPENDENCY ACCURACY. Is ordering logically consistent with the query? Are analysis steps before decision/recommendation steps? Assume symbolic dependencies are valid. Scoring: 1.0 logical; 0.5 minor issue; 0.0 illogical.

4. QUERY COVERAGE ACCURACY. Does the plan attempt to address all parts of the query? Are key sub-goals represented? Scoring: 1.0 full; 0.5 partial; 0.0 major intent missed.

–

OUTPUT FORMAT (STRICT). Return exactly one JSON object with the four metric

fields, each containing {"accuracy": <float>, "justification": "<1 sentence>"}, plus an "overall_assessment" single-sentence summary.

Be conservative. Prefer 1.0 unless a **clear structural violation** exists. Judge structure, not execution or correctness.

QUERY:
{user_message}

AVAILABLE TOOLS:
{available_tools}

GENERATED TOOL CALLS:
{generated_tool_calls}

U Free-Form Judge Prompt (A5)

The following prompt is used in the A5 prompt-format ablation (§4.2). It omits explicit per-metric definitions and scoring rubrics, asking the judge to reason freely and return a single holistic verdict per metric without anchoring instructions.

User

You are evaluating an AI agent's tool-calling plan.

Given the user query, the available tools, and the generated tool calls, assess how well the agent performed. Consider whether it selected appropriate tools, structured parameters correctly, ordered calls logically, and covered the user's intent.

Return a JSON object with four keys: tool_selection_accuracy, parameter_structure_accuracy, sequence_accuracy, and query_coverage_accuracy. Each key should map to an object with "accuracy" (a float between 0 and 1) and "justification" (one sentence). Also include an "overall_assessment" field.

QUERY:
{user_message}

AVAILABLE TOOLS:
{available_tools}

GENERATED TOOL CALLS:
{generated_tool_calls}

EXPECTED TOOL CALLS:
{expected_responses}