TRACEJUDGEBENCH: A UNIFIED BENCHMARK FOR VALIDATING LLM-AS-A-JUDGE ON AGENT EXECUTION TRACES

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM judges are now a standard way to evaluate LLM-based agents, but the methods and the benchmarks used to validate them do not agree with each other. Each defines its own error taxonomy, annotation scheme, and error distribution, so neither a judge's score nor a benchmark's verdict carries over to another. We present TraceJudgeBench, a meta-analysis that harmonizes ten judge-validation datasets and three unannotated trace corpora into one frame: 846 agent execution traces kept in their native formats, with every label mapped into a shared two-level taxonomy of ten fine-grained and five coarse categories. The frame lets existing and new evaluation methods, each with its own taxonomy, be cross-validated against one another and against the benchmarks they come from. We annotate 231 further traces to fill the gaps public sources leave open: API and system failures, environment failures, and clean executions with no error at all. Applying the frame to a prompted LLM, TRAIL, AEGIS, and a decomposed ensemble judge on one backbone shows what the separate benchmarks hide. Three of the four judges are blind by construction to categories their native schemes cannot name. Every judge reports errors on clean traces: false-error rates run from 27% to 74% for the three established judges and reach 100% for the ensemble run without abstention, and the prompted baseline stays in that range across three backbones. False-error also grows with trace length: on clean labels that two human annotators confirmed trace by trace, the prompted baseline and TRAIL rise 44 and 50 points from short to long clean traces, an association that the data cannot yet separate from task type. A judge validated on short traces carries no guarantee on long ones, and a corpus concentrated at one length cannot show it. The traces, taxonomy mappings, judge-output mapping code, and analysis scripts are included as supplementary material.

1 INTRODUCTION

LLM-based agents (Wang et al., 2023; Guo et al., 2024; Khapre et al., 2026; Qi et al., 2026) are increasingly deployed in environments where they must plan, call tools, and recover from intermediate failures. For such agents the final answer is not enough to evaluate: developers must verify that an agent followed instructions, interpreted tool outputs correctly, and completed tasks without hidden execution errors. This need has driven the use of LLM-as-a-judge methods (Zheng et al., 2023) for trace-level evaluation and debugging.

An LLM judge can be wrong too, so it must be validated. A judge that works on one benchmark may miss other failure modes, confuse error types, or flag valid traces as erroneous. Validation therefore has to test three things: whether a judge detects failures, whether it tells error types apart, and whether it recognizes a clean execution.

Recent work has introduced judge-oriented benchmarks and diagnostic datasets for agent trace evaluation, including AEGIS (Kong et al., 2026), AgentErrorBench (AEB) (Zhu et al., 2025), AFTraj (Zhang et al., 2026a), AgenTracer (Zhang et al., 2026b), AgentRx (Barke et al., 2026), AgentRewardBench (ARB) (Lù et al., 2025), Pumpkin (Zhidkovskaya et al., 2026), TraceElephant (Chen et al., 2026), TRAIL (Deshpande et al., 2025), and Who&When (Zhang et al., 2025). They are useful testbeds

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

for studying agent failures but hard to compare or reuse across studies. Each benchmark typically defines its own trace format, annotation scheme, error taxonomy, and implicit distribution over failure types. A judge evaluated on one dataset can look strong because the dataset overrepresents the types of errors it is designed to detect, while other practically important failures remain underrepresented.

Such benchmarks cannot simply be pooled. Their label spaces are incompatible, and their error distributions follow annotation-design choices rather than real failure priors, so several failure modes that matter in practice (API timeouts, environment setup errors, permission failures) are sparse or absent in every public source, and so are clean executions with no error. Section 2 sets out both problems in detail.

We present **TraceJudgeBench**, a meta-analysis and harmonization of existing judge-validation benchmarks. It preserves native trace representations while aligning source annotations under a shared L2/L1 taxonomy, so that evaluation methods with distinct taxonomies can be validated against each other and against the benchmarks they come from. To construct it, we analyzed existing benchmarks, mapped their annotations, and added traces for underrepresented categories (API and system failures, environment failures, hallucination, and clean no-error cases). We also downsampled overrepresented categories so that frequent error types do not dominate.

Our **contributions** are threefold: (1) we harmonize ten judge-validation datasets and three trace corpora into one frame that preserves native traces and maps their annotations under a unified L2/L1 taxonomy, so that methods with distinct taxonomies can be cross-validated; (2) we cross-validate four judge designs through this frame at both taxonomy levels, separating scheme-defined zeros from empirical failures and reporting per-category bootstrap intervals (Appendix A.8) and a mapping-perturbation study; and (3) we show that clean-trace false-error rises with trace length for two of the three established judges, by 44 and 50 points pooled, an association that a length-concentrated corpus cannot expose and that our data does not yet separate from task type.

2 Related Work: Existing Judge Benchmarks Are Fragmented

Judge-validation benchmarks differ in what is annotated, how errors are localized, and which label space is used. Some benchmarks emphasize trace-level error types, others focus on attribution to agents or steps, and others target agent-environment interactions or tool behavior. Labels with similar names are not always equivalent: a "tool failure" may refer to wrong tool selection, malformed arguments, failed execution, or an external API error. This heterogeneity makes cross-benchmark judge comparison unreliable without an explicit taxonomy-alignment layer. Failure taxonomies for multi-agent systems have been proposed independently (Cemri et al., 2025), but they are not grounded in existing benchmark annotations and do not address the cross-benchmark alignment problem. Meta-evaluation of LLM judges has centered on response pairs and reward models (Tan et al., 2025; Lambert et al., 2025), and agentic judges (Zhuge et al., 2025) are validated on their own task suites rather than across annotated trace corpora.

A second source of fragmentation is uneven error coverage. Mapped into a common taxonomy, public benchmarks have sharply different error priors (Appendix Figure 3), and those priors reflect annotation design, not any shared model of agent failures. Two consequences matter here. `api_system` and `environmental` failures are rare across every source, even though timeouts, malformed provider responses, missing resources and permission errors are common in practice (Shah et al., 2026); and clean no-error traces are scarce outside two sources, which makes false failure reports on valid executions hard to measure at all. TraceJudgeBench addresses both by aligning heterogeneous annotations under a shared taxonomy and adding traces for the underrepresented cases.

3 TraceJudgeBench

TraceJudgeBench makes heterogeneous agent execution traces comparable by preserving native trace formats while harmonizing their error annotations under a shared L2/L1 taxonomy.

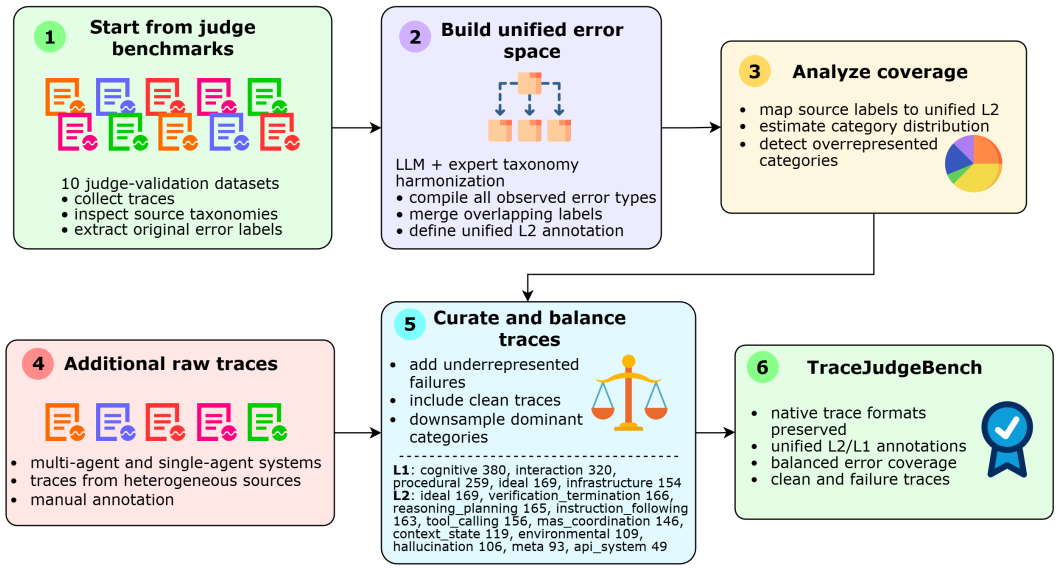


Figure 1: Construction pipeline of TraceJudgeBench. We collect traces and native annotations from existing judge-validation benchmarks, derive a unified L2 taxonomy via semantic clustering of source L3 labels, analyze coverage gaps, add traces for underrepresented failure modes, and curate a balanced benchmark while preserving native trace formats.

3.1 Source Datasets and Native Trace Preservation

TraceJudgeBench is built from heterogeneous judge-validation datasets and newly added traces. We start from existing benchmarks for evaluating agent and multi-agent execution traces, including TRAIL, AEGIS, Who&When, AgenTracer, AgentRx, Pumpkin, TraceElephant, AFTraj, and related sources. These datasets cover ReAct-style (Yao et al., 2023) tool-augmented agents, multi-agent conversation systems (Wu et al., 2024), and custom orchestrators. They differ in trace format, annotation scheme, taxonomy, and labeling granularity.

Figure 1 summarizes the construction process. We preserve each trace in its native format, including messages, tool calls, observations, environment logs, and metadata when available. The original datasets stay intact; TraceJudgeBench makes them comparable through their harmonized error annotations. Trace lengths also remain heterogeneous: most selected traces fall between approximately 5K and 100K tokens, with some exceeding one million tokens (Figure 5 in Appendix).

One property of the selection governs what a text judge can see. The ARB traces come from browser environments, but ARB judges receive either the accessibility tree or the screenshot as the final state, and its method is "compatible with both multimodal and text-only LLMs" (Lù et al., 2025). We use the accessibility-tree setting with visual observations pruned, and we exclude VisualWebArena, the one ARB environment requiring image-based reasoning, keeping WebArena, WorkArena, and AssistantBench. Every trace in TraceJudgeBench is available in full to a text-only judge, as text; the pruned visual observations are not.

3.2 Unified L2 Taxonomy and L1 Abstraction

To compare heterogeneous datasets, we introduce a unified L2 taxonomy and a coarser L1 abstraction (Figure 2). Taxonomy construction followed three steps. First, we compiled all native error labels across source benchmarks, yielding over 70 distinct L3 labels. Second, with an LLM proposing candidate groupings, we semantically clustered labels by their operational definition and the agent behavior they describe, resolving synonyms across sources and surfacing failure modes absent or underspecified in individual benchmarks. This yielded ten L2 categories: `reasoning_planning`, `hallucination`, `instruction_following`, `tool_calling`, `mas_coordination`, `context_state`, `verification_termination`, `environmental`, `api_system`, and `ideal`. Label merging decisions were reviewed by two authors, with disagreements resolved by discussion; the final L3-to-L2 mapping

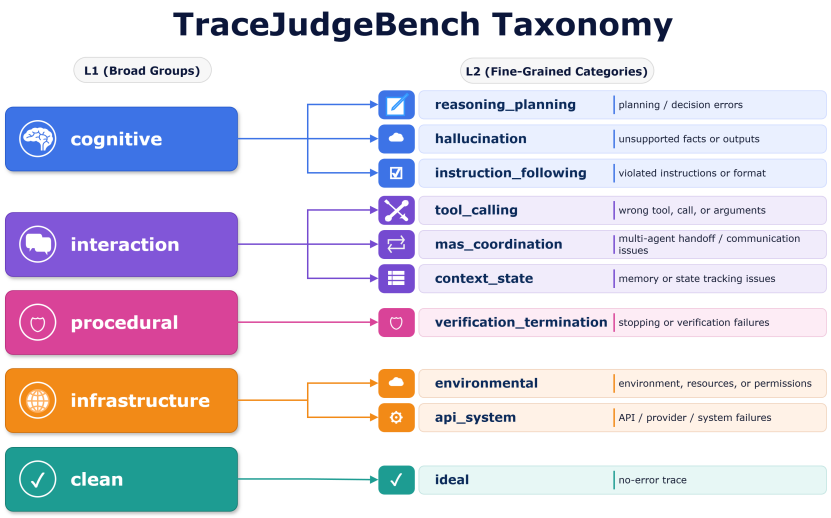


Figure 2: TraceJudgeBench taxonomy. The L2 taxonomy defines fine-grained error categories, while L1 groups them into broader classes for cross-dataset comparison.

was fixed before evaluation and released with the benchmark. Third, L2 categories are grouped into five L1 classes (`cognitive`, `interaction`, `procedural`, `infrastructure`, and `clean`) for cross-dataset comparison where fine-grained distinctions are unreliable. The `ideal` category is reserved for traces with no annotated error, enabling direct measurement of false-error rates.

3.3 Annotation Mapping and Added Traces

Since L2 categories were derived from source L3 labels, mapping each benchmark's annotations is a lookup rather than a projection onto a foreign scheme. For benchmarks with structured categorical labels (AEGIS, AEB, TRAIL, AgentRx, Pumpkin), we apply this lookup directly. Four benchmarks (AFTraj, AgenTracer, TraceElephant, and Who&When) annotate errors as natural-language descriptions in a `mistake_reason` field rather than formal category labels; these benchmarks focus on failure attribution (which agent, which step) rather than error-type classification. For Who&When, the `mistake_reason` is a human-written justification that constitutes the benchmark's ground-truth label. We map these existing descriptions to L2 using a prompted LLM classifier (Gemini 2.5 Flash); the LLM maps existing human-written annotations onto the schema and never annotates a raw trace. The resulting L2 assignments are one step removed from native annotations, a limitation we return to in Appendix A.15.

A single trace may contain multiple errors, so TraceJudgeBench is treated as a multi-label benchmark. Each trace can receive one or more L2 labels; `ideal` is used only when no error is assigned. Labels are deterministically mapped to L1.

Validating the free-text mapping. Two human annotators independently assigned an L2 category to 50 items stratified across the four free-text sources, working from the `mistake_reason` text alone. On the 47 items that carry text, they agree with each other on 83.0% ($\kappa = 0.78$) and the classifier agrees with each of them on 78.7% ($\kappa = 0.72$ and 0.73); on the 39 items where both annotators agree it matches 33. Per-category F1 on mapped and natively labeled traces differs in both directions, by category more than by judge (Appendix A.13). Because every score depends on this lookup, we also re-derive the gold labels under alternative assignments for five contestable L3 labels; no single change moves a per-category F1 by more than 0.033, and re-mapping the free text with a second model family (Claude Sonnet 4.6) changes no per-category winner on the full benchmark. Appendices A.4 and A.5 give the full audit, the slice figures and the perturbation study.

Added traces. Coverage analysis shows that `api_system` and `environmental` are sparse in public benchmarks compared to `reasoning_planning`, and clean `ideal` traces are largely absent outside ARB and Pumpkin. We address these gaps by annotating open-source agent traces and adding

them to TraceJudgeBench. Of the 846 traces, 615 come from the ten error-annotated benchmarks, 577 with their native labels and 38 ARB WorkArena and AssistantBench traces labeled through the review interface described below, 205 are drawn from three public trace corpora that ship no error annotations (`Exgentic/agent-llm-traces`, `nlile/misc-merged-claude-code-traces-v1`, `sammshen/swebench-minimax-traces`), and 26 were collected by us from natural runs to target the two sparsest categories (23 carrying `api_system`, 3 `environmental`).

Three human annotators labeled all 231 added traces and the 38 ARB WorkArena and AssistantBench traces, 269 in all, working on the raw traces in a purpose-built review interface. For each trace an LLM (`minimax-m3`) proposed a candidate label set, which the human annotator could accept, edit, or discard entirely, following the released L2 taxonomy definitions. The model never fixed a label and a trace could be rejected outright, so no contributed label is model output alone. This annotation role is distinct from the schema-mapping classifier described above: the classifier rewrites existing human-written descriptions into L2 and never sees a raw trace, whereas here the model drafts a candidate over a raw trace and a human decides. Each trace received a single adjudicated decision rather than parallel independent ones, so we do not report inter-annotator agreement for this stage; a double-annotated subset is planned for the next release.

Auditing the clean labels. The clean-trace results rest on the `ideal` labels, so we audited all of them. Two human annotators each read every trace originally labeled `ideal`, 200 in total, and recorded whether it contained any error. They reached the same verdict on 176 of the 197 traces both could judge (89.3%, $\kappa = 0.34$; errors are rare, which keeps κ low). We keep a trace as `ideal` only if both annotators judged it error-free, which leaves 169. The other 31 are removed from the benchmark: 21 on which the annotators disagreed, 7 in which both found an error, and 3 that one annotator could not judge from the trace. The removals fall on ARB (21), NLiLe (8) and Pumpkin (2), and 24 of the 31 are long traces; every Exgentic, SWE-bench and WorkArena clean label was confirmed. Neither auditor was among the three annotators who labeled the traces in the review interface. For reference, each flagged an error in about 9% of the originally clean traces (18 and 17 of 197). The audit sheet is released with the benchmark.

3.4 Dataset Statistics and Construction

TraceJudgeBench contains $N = 846$ traces. It is multi-label, so category counts sum to more than N: `ideal` 169, `verification_termination` 166, `reasoning_planning` 165, `instruction_following` 163, `tool_calling` 156, `mas_coordination` 146, `context_state` 119, `environmental` 109, `hallucination` 106, and `api_system` 49. A further 93 traces carry the `meta` roll-up label, a benchmark-specific artifact that we drop before scoring. The traces themselves stay in every denominator; the 30 that carry no other label enter as error traces with no evaluated category. Dropping `meta` is why the `procedural` L1 class has 259 traces at the dataset level but 166 in Table 10.

The final counts are not round because two selection objectives pull against each other: category balance and length balance. We first mapped the L2 distribution across all source benchmarks, then drew a stratified sample of 1,000 traces balancing L2 categories against token-length deciles. That sample still under-covered the sparsest categories, so we added 26 traces carrying `api_system` or `environmental` and 47 Who&When traces labeled only `hallucination`. The sample was also skewed short, so we dropped 196 AFTraj traces from the two lowest token-length deciles, which were trivially short and over-concentrated in `reasoning_planning` and `verification_termination`. Removing the 31 clean traces the audit did not confirm gives $1{,}000 - 196 + 26 + 47 - 31 = 846$.

The stratified sample used two properties of each trace only: its L2 labels and its token-length decile, and the L3-to-L2 mapping was frozen before any judge was run. The one later change, removing clean traces that two human annotators did not both confirm (Section 3.3), rests on those annotators' reading of each trace; no trace was kept, dropped, or weighted according to how any evaluated judge scored on it. TraceJudgeBench is a single evaluation set with no train/test split, intended for validating judges and not for training them.

A balanced core for iteration. The 846 traces total 86M tokens, which every judge pass must read at least once, so a four-judge evaluation is a real barrier to reuse. We therefore also release a 180-trace *balanced core*: 29 traces in every L2 category, all ten token-length deciles covered, at 20.8% of the full token cost. Equalizing category frequencies makes it a different measurement, not a

L2 category	Prompted LLM			TRAIL			AEGIS			Our ensemble			Random judge, F1		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	k=1.5	k=3	k=7.5
reasoning_planning	0.229	**0.735**	**0.349**	**0.272**	0.438	0.336	0.245	0.218	0.231	0.000$^\diamond$	0.000$^\diamond$	0.000$^\diamond$	0.171	0.235	0.311
hallucination	0.136	0.287	0.184	**0.204**	0.200	0.202	0.000$^\diamond$	0.000$^\diamond$	0.000$^\diamond$	0.115	**0.903**	**0.204**	0.136	0.178	0.214
instruction_following	0.245	0.599	0.348	0.255	0.606	**0.359**	**0.275**	0.485	0.351	0.192	**0.969**	0.320	0.166	0.235	0.307
tool_calling	**0.343**	0.431	0.382	0.318	0.651	**0.427**	0.000$^\diamond$	0.000$^\diamond$	0.000$^\diamond$	0.181	**0.962**	0.305	0.166	0.228	0.296
mas_coordination	**0.400**	0.587	**0.476**	0.000$^\diamond$	0.000$^\diamond$	0.000$^\diamond$	0.222	0.493	0.306	0.172	**0.979**	0.293	0.160	0.220	0.279
context_state	0.154	0.342	0.212	0.072	0.043	0.054	**0.166**	0.353	0.226	0.139	**0.966**	**0.243**	0.145	0.190	0.236
verification_termination	0.213	0.245	0.228	0.272	0.346	0.305	**0.295**	0.434	**0.351**	0.199	**0.982**	0.331	0.170	0.238	0.312
environmental	**0.394**	0.272	**0.322**	0.387	0.113	0.175	0.000$^\diamond$	0.000$^\diamond$	0.000$^\diamond$	0.129	**0.972**	0.227	0.138	0.177	0.219
api_system	**0.274**	0.436	**0.337**	0.172	0.511	0.257	0.000$^\diamond$	0.000$^\diamond$	0.000$^\diamond$	0.050	**0.891**	0.094	0.083	0.098	0.109
ideal	**0.898**	0.265	0.409	0.732	0.550	**0.628**	0.526	**0.728**	0.610	0.000	0.000	0.000	0.079	0.000	0.000

Table 1: Per-category L2 performance after mapping each judge's native outputs to L2: precision (P), recall (R) and F1. Judges are scored on the traces with parseable output (AEGIS 846/846; ensemble 840/846; TRAIL 822/846; Prompted LLM 818/846; all 169 `ideal` traces except 166 for the Prompted LLM). $^\diamond$ marks *scheme-defined* zeros: no label in the judge's native vocabulary maps to the category (Table 4; Section 4.1), so no backend change can raise them. The ensemble's `ideal` zero is empirical and moves with the aggregation rule (Appendix A.11). The last three columns are a random judge asserting k labels per trace (Appendix A.9); the judges assert 2.6, 1.9, 1.8 and 7.9.

cheaper estimate of the same one, and at 29 traces per category its estimates are noisy: across repeated draws it does not reliably reproduce the per-category ranking of Table 1. It is meant for development iteration and for checking that a judge exercises every category and length regime; Appendix A.7 scores all four judges on it. The core is a strict subset of the benchmark, so an evaluation begun on it is completed by running the remaining 666 traces. All reported numbers are computed on the full set.

4 Judge Evaluation via Taxonomy Alignment

We evaluate judges through taxonomy alignment. Each judge operates in its native output space, using its own prompts, diagnostic categories, and output schema. Its predictions are then mapped to the unified TraceJudgeBench L2 taxonomy and further to L1. Source-annotation mapping defines the gold labels; judge-output mapping defines comparable predictions, and the two are kept separate.

4.1 Evaluated Judges

We evaluate four judge designs: a prompted LLM baseline, the TRAIL judge, the AEGIS judge, and our ensemble judge. The prompted LLM baseline uses a single trace-level prompt to predict L2 categories directly from the execution trace, so it is the one judge natively aligned with the target scheme, which may favor it on L2. The TRAIL and AEGIS judges are evaluated in their native output formats, preserving the diagnostic schemes used by each method.

The ensemble judge decomposes evaluation into fifteen specialized agent-level and system-level checks: observation alignment, policy alignment, state consistency, task completeness, tool selection, tool parameter extraction, tool performance, prompt quality, MAS complexity, MAS role distribution, MAS task transfer, MAS task completion, API issues, environment setup errors, and a summarizer. The deployed configuration has *no* planning sub-judge, which is why the ensemble cannot express `reasoning_planning` (Section 5.3).

Overlap between the ensemble and a source benchmark. The ensemble reimplements the metric suite of Zhidkovskaya et al. (2026), whose Pumpkin dataset contributes 85 of the 846 traces, so on 10.0% of the benchmark it is scored against labels from the same family of metrics; this does not appear to help it (Appendix A.11). The native judges show larger gaps on their own sources: AEGIS scores 0.501 macro-F1 on its 123 traces against 0.165 on the rest, and TRAIL 0.372 on the 18 of its 20 it covers against 0.269, though the category mix differs.

Harness. Judges receive each trace whole, and a trace exceeding the backbone context window yields no output rather than a truncated one: none of the nine traces above 1M tokens appears in the prompted or TRAIL runs. Reported rates describe long traces that fit, not mutilated inputs. The ensemble is the exception, since its sub-judges consume per-step slices rather than whole traces, which is why it covers the largest traces; the four judges are compared on the same trace set but not on identical inputs. All four run on Gemini 2.5 Flash at temperature 0.

4.2 Output Mapping and Metrics

Each judge produces outputs in its native format, which we normalize and map to one or more L2 categories. If a judge predicts no error, the output is mapped to `ideal`; if it predicts multiple errors, all corresponding L2 categories are retained. L2 predictions are then mapped to L1.

Because traces may have multiple labels, we compute precision, recall, and F1 in a multi-label one-vs-rest setting for each L2 and L1 category. The `ideal` category measures whether a judge recognizes a valid execution instead of assigning it a false error. We report the L2 view throughout and the L1 view in Appendix A.10, where the ranking is not stable across levels: the prompted LLM leads on L2 macro-F1 and TRAIL on L1.

5 Experiments and Results

We evaluate both benchmark coverage and judge behavior. First, we compare L2 error coverage across source datasets and TraceJudgeBench. Second, we evaluate judges after mapping their native outputs to the unified taxonomy, focusing on per-category performance and clean-trace recognition.

5.1 Benchmark Coverage Analysis

Appendix Figure 3 shows the distribution of L2 labels across source benchmarks and TraceJudgeBench after mapping. The same skew appears in the aggregate distribution before our additions, where infrastructure failures are a small fraction of all aligned labels (Figure 4, Appendix).

5.2 Per-Category L2 Judge Performance

Table 1 reports per-category precision, recall, and F1 after mapping each judge's native outputs to the unified L2 taxonomy. The prompted LLM baseline detects many errors but has low precision across most categories, suggesting that a single trace-level prompt tends to over-predict failures. TRAIL improves over the prompted baseline on several categories, especially `tool_calling`, `verification_termination`, and `ideal`, but performs poorly on categories outside its main diagnostic focus, such as `mas_coordination` and `context_state`.

AEGIS shows a different profile: the highest `ideal` recall and the best `verification_termination` F1, competitive results on `instruction_following` and `mas_coordination`, but near-zero recovery on `hallucination`, `tool_calling`, `environmental`, and `api_system` after taxonomy alignment. A paired bootstrap separates the top two judges only on `mas_coordination`, `environmental` and `api_system`, and the prompted baseline's `api_system` lead comes mostly from the traces labeled in the review interface (F1 0.64 against 0.05 on native labels). The pattern holds across judges: each recovers the failure modes its native design emphasizes. A judge chosen on aggregate scores without taxonomy alignment can therefore be blind to entire failure classes, and per-category reporting is what exposes those blind spots.

In the current ensemble configuration, recall is high for many non-`ideal` categories, but precision is low. The ensemble is sensitive to potential failures but not calibrated for category selection or abstention. The random columns of Table 1 calibrate these numbers. TRAIL and AEGIS exceed a random judge of their own volume (the ladder of Appendix A.9, interpolated at each judge's labels per trace) on every category they can express, except TRAIL on `context_state`; the prompted baseline exceeds it everywhere, but by only 0.008 on `verification_termination` and 0.017 on `hallucination`, so those two cells carry no evidence of detection; and the ensemble, at 7.9 labels per trace, tracks the random judge to within 0.02 on all eight categories it can express (Appendix A.9).

5.3 Clean-Trace Recognition and False-Error Patterns

We separately evaluate whether judges can recognize clean executions. For traces annotated as `ideal`, a false-error prediction occurs when the judge assigns any non-`ideal` category to an otherwise clean trace.

No judge recognizes clean traces reliably (Table 1). The prompted LLM baseline reaches only 0.265 recall on `ideal`, corresponding to a 0.735 false-error rate. TRAIL improves to 0.550 recall and, with

Judge	Backbone	False-error	95% CI
(a) Judge designs, shared backbone			
AEGIS	Gemini 2.5 Flash	27.2%	[21.2, 34.2]
TRAIL	Gemini 2.5 Flash	45.0%	[37.2, 52.4]
Prompted LLM	Gemini 2.5 Flash	73.5%	[66.7, 80.1]
(b) Prompted baseline, backbone transfer			
Prompted LLM	Sonnet 4.6	61.5%	[54.4, 68.6]
Prompted LLM	DeepSeek v4 Pro	69.6%	[62.6, 76.5]
(c) By trace length (short: deciles 3–5, n=78; long: 6–9, n=91, or 88 for the Prompted LLM)			
AEGIS	short / long	10.3% / 41.8%	[3.8, 16.7] / [31.9, 51.6]
TRAIL	short / long	17.9% / 68.1%	[10.3, 26.9] / [58.2, 76.9]
Prompted LLM	short / long	50.0% / 94.3%	[38.5, 60.3] / [88.6, 98.9]

Table 2: Clean-trace false-error rate, defined as $1 -$ recall on the `ideal` category. Intervals are trace-level bootstrap percentile intervals over 2,000 resamples. Panel (a) holds the backbone fixed and varies the judge design; (b) holds the judge fixed and varies the backbone; (c) splits the clean traces at the median token-length decile. Table 11 repeats split (c) inside a single source. The ensemble is excluded here and analyzed separately in Appendix A.11, because its aggregation rule determines the rate.

higher precision, the best `ideal` F1 at 0.628; AEGIS has the highest clean recall, 0.728, and 0.610 F1; Appendix A.13 breaks this down by source. The prompted baseline is conservative about clean verdicts rather than blind to them: it calls 26.5% of clean traces clean against 0.8% of error traces, and a judge that said "clean" on the same share of traces at random would reach 94.0% false-error (Appendix A.9). Its 73.5% is a threshold that sits too far toward flagging.

We report the ensemble as a diagnostic probe: its near-zero `ideal` recall shows high recall across failure categories paid for with false errors on every clean trace, and the last paragraph of this section traces that price to the aggregation rule.

Uncertainty and backbone transfer. Every judge runs once per trace at temperature 0. We quantify the uncertainty this leaves over the choice of traces with a trace-level bootstrap of 2,000 resamples, and we repeat the prompted baseline on two further backbones. Temperature 0 does not remove run-to-run variation, which a bootstrap over traces does not measure and which we did not quantify. The 95% intervals of the three established judges do not overlap, so the pooled ordering AEGIS $<$ TRAIL $<$ Prompted LLM survives resampling over traces; it depends on Exgentic (Appendix A.13). Changing the backbone gives 61.5% [54.4, 68.6] on Sonnet 4.6 and 69.6% [62.6, 76.5] on DeepSeek v4 Pro against 73.5% [66.7, 80.1] on Gemini 2.5 Flash; the intervals overlap, so the three backbones are not separated. What the comparison supports is that all three sit far above a usable false-error rate, not any ordering among them. For the prompted design, over-flagging on clean traces persists on all three backbones.

The ensemble's 100% is an aggregation rule, not a cost of ensembling. The configuration in Table 1 flags a category whenever the corresponding sub-judge runs, whatever verdict it returns, with no abstention path, so no clean trace can survive it. Re-aggregating the identical sub-judge outputs on their verdicts, with abstention and nothing retrained, lifts macro-F1 from 0.202 to 0.305 and drops clean-trace false-error from 100% to 58%. On macro-F1 that lifts it past TRAIL and AEGIS but not past the prompted baseline's 0.325, and by a post-hoc adjustment the baselines did not receive. `reasoning_planning` stays at zero under both rules, matching the structural account above. Appendix A.11 gives the ablation and the source-overlap control.

5.4 Clean-Trace False Errors Grow with Trace Length

The rates in Table 2 average over a benchmark that spans three orders of magnitude in trace length, and the average hides most of the structure. Splitting the clean traces at the median token-length decile (deciles 3–5 against 6–9; no clean trace falls in deciles 0–2) separates two regimes, and every judge is worse above the split than below it: the prompted baseline rises from 11.1% false-error in decile 3 (n=9) to 100% in deciles 7 and 9 (n=18 and 15), flagging every clean trace it sees there.

Pooled (Table 2c), every judge degrades on long traces by 31.5 to 50.2 points, with non-overlapping intervals. Pooling alone would not settle it, since the halves differ in composition: 81% of short clean traces come from Exgentic against 40% of long ones. Table 11 therefore repeats the split inside the two sources with clean traces on both sides, and a logistic regression of false error on log trace length with source fixed effects puts the effect on one scale. For the prompted baseline and TRAIL it survives: the odds of a false error rise 21-fold [5.4, 84] and 17-fold [6.7, 42] per tenfold increase in length, and both Exgentic splits are significant (Fisher $p < 10^{-4}$). Inside Exgentic, however, length coincides with task environment: short clean traces are almost all τ^2 retail dialogues and long ones AppWorld and BrowseComp-Plus tasks, and the one environment with both, τ^2 telecom, points the same way on 11 traces (Appendix A.13), so this evidence does not separate length from task type. AEGIS never false-flags an Exgentic clean trace at any length, so most of its pooled rise is composition; its source-adjusted estimate, 11.5 [0.6, 237] on 51 traces, is not established. The ARB/WebArena splits point the same way for all three judges, but at 10 short and 19 long traces none is significant on its own ($p = 0.06$ to 0.24). Discrimination falls with length as well. Scoring each trace by the number of errors a judge flags, the pooled AUROC separating error from clean traces drops between deciles 3–5 and 6–9 from 0.93 to 0.71 for the prompted baseline, 0.92 to 0.78 for TRAIL and 0.91 to 0.75 for AEGIS; without Exgentic it remains clear for AEGIS, points the same way for the prompted baseline, and does not appear for TRAIL (Appendix A.13). These are associations, in line with long-context degradation on other tasks (Liu et al., 2024).

Label noise offers an alternative explanation, and it works against this result. A long trace labeled clean that in fact contains an error raises long-trace false-error for any judge that catches it, so missed errors concentrated in long traces manufacture part of the gap. The audit reduces this risk without removing it. Each annotator found errors the other missed: 28 of the audited traces drew an error from at least one of them and both agreed on only 7, so some errors both missed likely remain among the 169 (about 16 by a capture–recapture estimate, Appendix A.13), and the removals were mostly long traces (24 of 31). Residual noise of that size could account for much of a gap like AEGIS's; it is a less likely account of the 54- and 58-point gaps the prompted baseline and TRAIL show inside Exgentic.

A judge validated on short traces therefore gives no guarantee on long ones, and a corpus concentrated at one length regime cannot expose this. That is why Section 3.4 balances the benchmark across length deciles.

6 Conclusion

We introduced TraceJudgeBench, a harmonization of existing judge-validation benchmarks under one taxonomy that lets evaluation methods with distinct taxonomies be cross-validated against each other and against their sources. Existing judge-validation datasets cover the taxonomy unevenly, with infrastructure failures and clean traces sparse throughout. Judge performance is strongly category-dependent, each judge reflecting the scope of its native diagnostic scheme, so none performs well everywhere. Clean-trace recognition is the most persistent weakness: 27% to 74% false-error across established judges, 100% for an ensemble run without abstention, and 62% to 74% for the prompted baseline across three backbones. The judges discriminate but flag too readily: over all lengths, the prompted baseline calls 26.5% of clean traces clean but only 0.8% of error traces. On long traces the prompted baseline and TRAIL degrade by a further 44 and 50 points, a rise that also appears inside the largest corpus, where it coincides with task environment. Reporting false-error rates and category-level gaps alongside detection (Kartik et al., 2025; Pan et al., 2026) tells a practitioner which judge is safe for which deployment (Rabanser et al., 2026).

Limitations. Appendix A.15 gives six constraints in full; one binds hardest. Every `ideal` label was confirmed by two human annotators, but two readers can miss the same error. Residual noise charges a judge for catching a real error and, concentrated in long traces, inflates the length gap (Section 5.4).

AI use statement

Generative AI tools were used in this work in two distinct roles, and we separate them deliberately because one is part of the method and one is not.

As part of the method, LLMs are research artifacts under study. All four evaluated judges are LLM-based and run on Gemini 2.5 Flash, with the prompted baseline additionally run on Sonnet 4.6 and DeepSeek v4 Pro (Section 4.1, Table 2). When we built the taxonomy, an LLM proposed candidate groupings of the source labels, which two authors reviewed and revised (Section 3.2). A prompted classifier, also on Gemini 2.5 Flash, maps free-text `mistake_reason` annotations from four source benchmarks into the L2 taxonomy (Claude Sonnet 4.6 re-maps the same texts as a robustness check, Appendix A.4), and `minimax-m3` proposed candidate labels that human annotators accepted, edited, or discarded when we annotated the 231 added traces and 38 ARB WorkArena and AssistantBench traces (Section 3.3). All three uses are described in the main text, and none leaves a model decision unreviewed: two authors reviewed every taxonomy grouping, Section 3.3 reports human agreement with the mapping classifier, and none of the 269 labels drafted this way is model output that a human did not adjudicate.

As authoring assistance, we used LLM tools for manuscript editing and for parts of the analysis code that computes the reported tables. The judge harnesses, the ensemble implementation and the data-construction pipeline were written by the authors, with LLM assistance limited to editing. We also used LLM tools to review drafts of the paper and to check its claims against the analysis outputs; the research questions, the study design and the conclusions are the authors'. All AI-assisted work was reviewed by the authors. Every numeric result reported in this paper was recomputed from the released artifacts and checked against the manifest; AI-assisted code was read and verified by the authors before its output entered the paper. We take responsibility for the final content of this work, including all text, claims, and artifacts.

Ethics statement

TraceJudgeBench is assembled from agent-execution traces released with judge-validation benchmarks, from three public trace corpora, and from traces the authors release. Each trace keeps its source's license and terms, recorded per trace in the manifest; because TRAIL and GAIA-derived traces may not be reshared outside a gated or private repository, the dataset is distributed through one. Traces are released unaltered apart from one live API key that we redacted. Four sources state no license, and their traces are included for research use while we seek the authors' permission. We scanned every trace for credentials and personal data (Appendix A.14); the remaining matches are synthetic test accounts, placeholders and public repository metadata already present in the source releases. The work involves no human subjects: all annotators and auditors are authors of this paper, and annotation was carried out on machine-generated execution traces. We release the benchmark for judge validation, not judge training, and we note in Appendix A.15 that its label frequencies are chosen for diagnostic coverage, so they should not be read as estimates of deployment error rates.

Reproducibility statement

The benchmark, the manifest recording every trace's source and labels, the frozen L3-to-L2 mapping, the judge-output mapping code, the mapping-audit sheet and the clean-label audit sheet are released as supplementary material and at anonymized links for review: code and analysis at `https://anonymous.4open.science/r/TraceJudgeBench-A7B1`, and the traces at `https://anonymous-hf.com/a/22czqxc7v6ml/`. Section 3.4 gives the selection procedure and the arithmetic behind $N = 846$; Appendix A.3 gives the complete per-benchmark L3-to-L2 lookup, which reproduces every cell of Table 1; and Appendix A.6 gives the provenance of every clean trace. All judges run at temperature 0, and the mapping classifier runs at temperature 0 with cached outputs, so both are deterministic. Confidence intervals in Table 2 come from a trace-level bootstrap over 2,000 resamples.

References

Shraddha Barke, Arnav Goyal, Alind Khare, Avaljot Singh, Suman Nath, and Chetan Bansal. AgentRx: Diagnosing AI agent failures from execution trajectories. *arXiv preprint arXiv:2602.02475*, 2026.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E.

Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025.

Mengzhuo Chen, Junjie Wang, Fangwen Mu, Yawen Wang, Zhe Liu, Huanxiang Feng, and Qing Wang. Seeing the whole elephant: A benchmark for failure attribution in LLM-based multi-agent systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19888–19905, 2026. doi: 10.18653/v1/2026.acl-long.912.

Darshan Deshpande, Varun Gangal, Hersh Mehta, Jitin Krishnan, Anand Kannappan, and Rebecca Qian. TRAIL: Trace reasoning and agentic issue localization. *arXiv preprint arXiv:2505.08638*, 2025.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*, 2024.

N. V. J. K. Kartik, Garvit Sapra, Rishav Hada, and Nikhil Pareek. AgentCompass: Towards reliable evaluation of agentic workflows in production. *arXiv preprint arXiv:2509.14647*, 2025.

Smita Khapre, Melkamu Abay Mersha, Zainab Olalekan, Hassan Shakil, Samer Iskander, Asser Moustafa, and Jugal Kalita. Agentic AI systems: A systematic survey of multi-agent architectures, cognitive foundations, interaction, explainability, security, and performance evaluation. *Neurocomputing*, 696:134049, 2026. doi: 10.1016/j.neucom.2026.134049.

Fanqi Kong, Ruijie Zhang, Huaxiao Yin, Guibin Zhang, Xiaofei Zhang, Ziang Chen, Zhaowei Zhang, Xiaoyuan Zhang, Song-Chun Zhu, and Xue Feng. AEGIS: Automated error generation and attribution for multi-agent systems. In *International Conference on Learning Representations (ICLR)*, 2026.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. RewardBench: Evaluating reward models for language modeling. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1755–1797, 2025.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

Xing Han Lù, Amirhossein Kazemnejad, Nicholas Meade, Arkil Patel, Dongchan Shin, Alejandra Zambrano, Karolina Stańczak, Peter Shaw, Christopher J. Pal, and Siva Reddy. AgentRewardBench: Evaluating automatic evaluations of web agent trajectories. In *Conference on Language Modeling (COLM)*, 2025.

Melissa Z. Pan, Negar Arabzadeh, Riccardo Cogo, Yuxuan Zhu, Alexander Xiong, Lakshya A. Agrawal, Huanzhi Mao, Emma Shen, Sid Pallerla, Liana Patel, Shu Liu, Tianneng Shi, Xiaoyuan Liu, Jared Davis, Emmanuele Lacavalla, Alessandro Basile, Shuyi Yang, Paul Castro, Daniel Kang, Koushik Sen, Dawn Song, Joseph E. Gonzalez, Ion Stoica, Matei Zaharia, and Marquita Ellis. Characterizing agents in production. In *International Conference on Machine Learning (ICML)*, 2026.

Shihao Qi, Jie Ma, Rui Xing, Wei Guo, Xiao Huang, Zhitao Gao, et al. Beyond individual intelligence: Surveying collaboration, failure attribution, and self-evolution in LLM-based multi-agent systems. *arXiv preprint arXiv:2605.14892*, 2026.

Stephan Rabanser, Sayash Kapoor, Peter Kirgis, Kangheng Liu, Saiteja Utpala, and Arvind Narayanan. Towards a science of AI agent reliability. In *International Conference on Machine Learning (ICML)*, 2026.

Mehil B. Shah, Mohammad Mehdi Morovati, Mohammad Masudur Rahman, and Foutse Khomh. Characterizing faults in agentic AI: A taxonomy of types, symptoms, and root causes. *arXiv preprint arXiv:2603.06847*, 2026.

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

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, et al. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*, 2023.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In *Conference on Language Modeling (COLM)*, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Boxuan Zhang, Jianing Zhu, Zeru Shi, Dongfang Liu, and Ruixiang Tang. AgentForesight: Online auditing for early failure prediction in multi-agent systems. *arXiv preprint arXiv:2605.08715*, 2026a.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. AgenTracer: Who is inducing failure in the LLM agentic systems? In *International Conference on Learning Representations (ICLR)*, 2026b.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? On automated failure attribution of LLM multi-agent systems. In *International Conference on Machine Learning (ICML)*, 2025. Spotlight.

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. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.

Alina Zhidkovskaya, Kirill Rapatskikh, Jerzy Kaminski, Grigorii Barakhsin, Anna V. Kalyuzhnaya, Alexey Druzhinin, Andrey Savchenko, Julia Belikova, Konstantin Polev, and Nikolay O. Nikitin. Towards design of an automated judge for multi-agent systems. In *Trustworthy Agentic AI Workshop at the AAAI Conference on Artificial Intelligence*, 2026.

Kunlun Zhu, Zijia Liu, Bingxuan Li, Muxin Tian, Yingxuan Yang, Jiaxun Zhang, Pengrui Han, Qipeng Xie, Fuyang Cui, Weijia Zhang, Xiaoteng Ma, Xiaodong Yu, Gowtham Ramesh, Jialian Wu, Zicheng Liu, Pan Lu, James Zou, and Jiaxuan You. Where LLM agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*, 2025.

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. Agent-as-a-judge: Evaluate agents with agents. In *International Conference on Machine Learning (ICML)*, 2025.

A Appendix

A.1 Aggregate Source-Benchmark Error Distribution

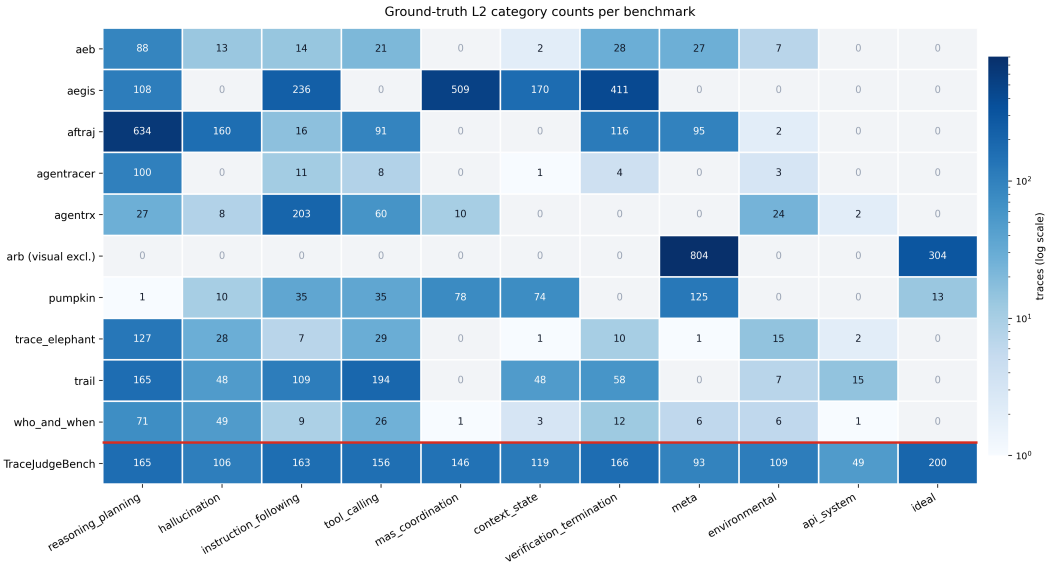


Figure 3: Distribution of harmonized L2 labels across source judge-validation benchmarks and TraceJudgeBench. Cells count label occurrences, not traces, so a multi-label source can exceed its trace count. Source rows cover the annotated portion of each release, which for AEGIS, AgenTracer and AFTraj is a subset of the traces reported in Table 3; the TraceJudgeBench row gives the final $N = 846$ selection after the downsampling and clean-label audit of Section 3.4. Public benchmarks have highly uneven error priors: some categories dominate within individual sources, while infrastructure failures and clean traces are sparse or absent. TraceJudgeBench fills these gaps across the unified L2 taxonomy.

Figure 4 shows the aggregate L1 and L2 label distribution across all source benchmarks before adding TraceJudgeBench-specific traces. The distribution is highly skewed: cognitive and interaction errors dominate, while infrastructure-related failures are rare.

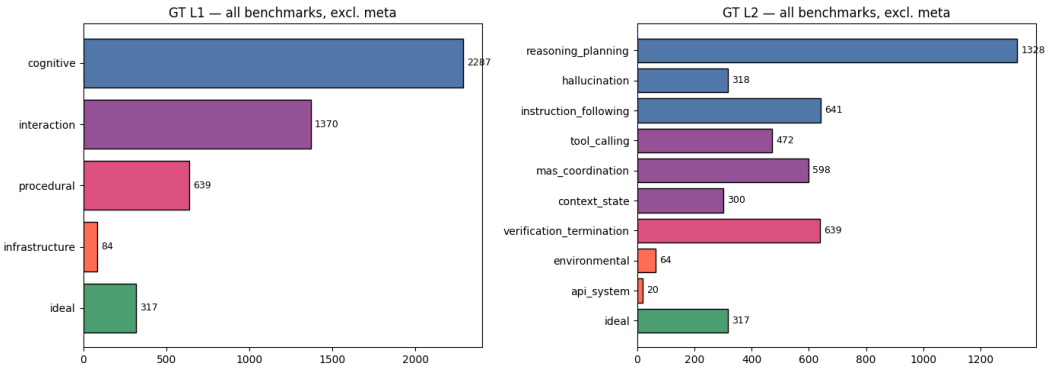


Figure 4: Aggregate L1 and L2 label distribution across source benchmarks before TraceJudgeBench additions, excluding meta labels. The distribution shows strong concentration in cognitive and interaction errors, with sparse coverage of infrastructure-related failures.

Table 3: Benchmark sources *considered* for TraceJudgeBench. The final column gives the number of traces each source contributes to the released benchmark; AgentFail was surveyed but not incorporated. The ten incorporated sources contribute 615 traces in total, 577 with their native labels and 38 ARB WorkArena and AssistantBench traces labeled by us; the remaining 231 traces were added and annotated by us (Section 3.3).

Benchmark	**MAS architecture (trace source)**	**Judge architecture**	**Evaluation objective**	**N-traces (source)**	**In TJB**
Who & When	ReAct-style tool-augmented QA agents; single- and multi-agent reasoning pipelines	Single LLM judge; prompt-based rubric; no SFT; optional self-consistency sampling	Agent-level attribution of failure; step-level correctness and error localization	176	80
TRAIL	Web navigation agents; SWE-style coding agents; environment-interaction agents (browser / API / sandbox)	Single LLM judge; hierarchical prompting; no training; decomposition into sub-criteria	Detection of planning, environment, and tool-use failures in execution traces	148	20
AEGIS	Heterogeneous MAS: QA, math reasoning (GSM8K/MATH), coding (HumanEval), and multi-agent debate systems	Multi-aspect prompt-based LLM judge; rubric decomposition; no SFT	Assessment of (1) agent competence, (2) inter-agent coordination, (3) validation correctness	9533	123
AgentRewardBench	Web agents (WebArena / VisualWebArena), workstation agents (WorkArena), assistant-style task agents	Single LLM reward-style judge; scalar/binary scoring; prompt-only (no training)	Global task completion judgment (success / partial / failure)	1302	85
AgentErrorBench	Goal-driven interactive agents (ALFWorld, WebShop, GAIA subsets)	LLM-based error classifier; taxonomy-driven prompting; no SFT	Fine-grained failure categorization (memory, planning, reflection, action, tool/API, environment)	200	27
AgentRx	Structured API workflows (Tau-bench); incident management (Flash); web/file tasks (Magentic-One)	Dual-stage constraint synthesizer; LLM adjudicator; validation log analysis	Identification of first unrecoverable failure step; classification into 10-category root-cause taxonomy	115	73
TraceElephant	Single-agent scaffolds and multi-agent orchestrators (Captain-Agent, Magentic-One, SWE-Agent)	LLMs with full execution observability; access to inputs and environmental feedback	Developer-facing failure attribution; full execution narrative analysis; decisive failure step identification	220	31
AFTraj	Coding, math reasoning, and general agentic domains	AgentForesight-7B online auditor; coarse-to-fine RL	Online step-by-step auditing; early failure prediction and intervention; decisive error localization	2276	77
AgenTracer	Reasoning and coding agents (MBPP+, KodCode, GSM8K, MATH, structured puzzles)	LLM judge with trajectory-level decomposition and retrieval; prompt-only	Localization of first failure step and causal action attribution	2500	14
AgentFail	General multi-agent systems (synthetic and real-world deployments)	Hierarchical LLM judge; optional ensemble of specialized prompt-based evaluators	System-level failure detection and categorization (agent / system / environment)	307	—
Pumpkin	Multi-agent pipeline traces with heterogeneous reasoning and tool-use patterns	Ensemble of 10 independent LLM judges followed by a final aggregator; prompt-only (no SFT)	Binary success/failure classification of full trajectory	328	85

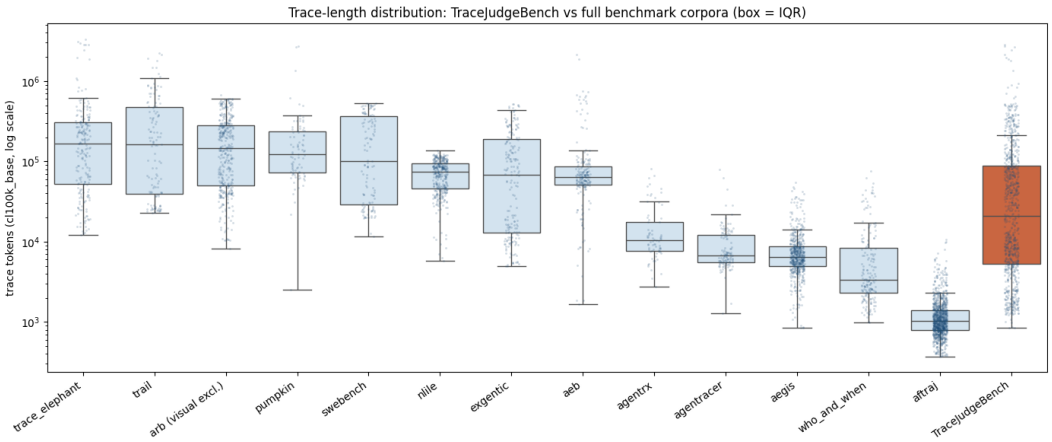


Figure 5: Trace-length distribution across source benchmark corpora and TraceJudgeBench. Boxes show the interquartile range, and points correspond to individual traces. TraceJudgeBench keeps the spread of lengths without being dominated by very short or very long traces.

A.2 Trace Length Statistics

Figure 5 compares trace-length distributions across the source benchmark corpora and the final TraceJudgeBench selection. Trace length is measured in tokens and shown on a logarithmic scale. The source corpora differ widely: some contain mostly short traces, while others include long executions with extensive intermediate reasoning, tool calls, observations, and environment logs.

TraceJudgeBench keeps this spread while limiting the length bias of any single source. Its distribution spans several orders of magnitude, with most traces between approximately 5K and 100K tokens and a small number above one million, a broader length profile than any single source benchmark offers.

A.3 L3-to-L2 Category Mapping

Table 4 lists the complete L3-to-L2 lookup for all natively annotated source benchmarks. Four benchmarks (AFTraj, AgenTracer, TraceElephant, Who&When) provide natural-language `mistake_reason` annotations instead of categorical labels; their L2 was derived by LLM classification of those descriptions, and the L3 field carries an `llm_` prefix as a provenance marker. The `meta` category is a benchmark-specific roll-up excluded from judge evaluation.

Table 4: Per-benchmark L3-to-L2 mapping for natively annotated sources.

Native L3 label	**Mapped L2**
AEGIS	
conversation_reset	mas_coordination
disobey_role_spec	instruction_following
disobey_task_spec	instruction_following
fail_to_ask_clarification	mas_coordination
ignored_agent_input	mas_coordination
incomplete_verification	verification_termination
incorrect_verification	verification_termination
information_withholding	mas_coordination
loss_of_history	context_state
premature_termination	verification_termination
reasoning_action_mismatch	reasoning_planning
step_repetition	context_state
task_derailment	mas_coordination
unaware_of_termination	verification_termination
AgentErrorBench (AEB)	
causal_misattribution	reasoning_planning
constraint_ignorance	instruction_following
environment_error	environmental
hallucination	hallucination
impossible_action	reasoning_planning
inefficient_plan	reasoning_planning

(continued on next page)

Table 4 *(continued)*

Native L3 label	Mapped L2
invalid_action	tool_calling
llm_limit	instruction_following
memory_retrieval_failure	context_state
misalignment	reasoning_planning
outcome_misinterpretation	tool_calling
over_simplification	reasoning_planning
parameter_error	tool_calling
progress_misjudge	verification_termination
step_limit	verification_termination
tool_execution_error	tool_calling
AgentRx	
guardrails_triggered	environmental
instruction_adherence_failure	instruction_following
instruction_plan_adherence_failure	instruction_following
intent_not_supported	tool_calling
intent_plan_misalignment	reasoning_planning
invalid_invocation	tool_calling
invention_of_new_information	hallucination
misinterpretation_of_tool_output	tool_calling
system_failure	api_system
underspecified_user_intent	mas_coordination
Pumpkin	
all_metrics_ideal	ideal
mas_complexity	mas_coordination
mas_planning	reasoning_planning
mas_roles_distribution	mas_coordination
mas_summarizer	meta
mas_task_completion	meta
mas_task_transfer	context_state
observation_alignment	hallucination
state_consistency	context_state
task_completeness	instruction_following
tool_parameter_extraction	tool_calling
tool_selection	tool_calling
TRAIL	
authentication_error	api_system
context_handling_failure	context_state
environment_setup_error	environmental
formatting_error	tool_calling
goal_deviation	reasoning_planning
incorrect_memory_usage	context_state
incorrect_problem_identification	reasoning_planning
instruction_non_compliance	instruction_following
language_only	hallucination
poor_information_retrieval	reasoning_planning
resource_abuse	verification_termination
resource_exhaustion	verification_termination
resource_not_found	api_system
service_error	api_system
task_orchestration	reasoning_planning
task_orchestration_error	reasoning_planning
timeout_issue	api_system
tool_definition_issue	tool_calling
tool_output_misinterpretation	tool_calling
tool_related	tool_calling
tool_selection	tool_calling
tool_selection_error	tool_calling

AFTraj, AgenTracer, TraceElephant, Who&When: these benchmarks provide natural-language `mistake_reason` descriptions rather than categorical labels. L2 was as signed by LLM classification of those descriptions; the L3 field records `llm_<category>` as a provenance marker.

A.4 Free-Text Mapping Audit

The mapped labels are one step removed from native annotations, so we audit them directly. Two human annotators independently assigned an L2 category to 50 items stratified across the four free-text sources. They worked from the `mistake_reason` text alone, which is the only input the classifier sees, and followed the released L2 definitions. Three of the 50 items have no `mistake_reason` text, and all three raters mark them `meta`; on the 47 items with text, the annotators agree with each other on 83.0% ($\kappa = 0.78$), and the classifier agrees with each of them on 78.7% ($\kappa = 0.72$ and 0.73). On the 39 items where both annotators agree, the classifier matches 33 (84.6%). The classifier runs on Gemini 2.5 Flash, the backbone of every evaluated judge, so on the four mapped sources a judge may agree with the gold labels partly because it shares a model with the mapper. To test this, we re-mapped the 186 in-benchmark `mistake_reason` texts with Claude Sonnet 4.6, using the same prompt, and re-scored every judge against the resulting gold labels, which change on 57 traces. The two mappers agree on 69.4% of texts ($\kappa = 0.60$). On the full benchmark no per-category winner changes, the macro-F1 order stays the same, and no per-category F1 moves by more than

0.022. On the mapped traces alone the macro-F1 order also holds, and the winner changes in three categories: `mas_coordination` and `context_state`, which have one positive trace each under the Gemini labels, and `verification_termination`, where AEGIS overtakes the prompted baseline (0.164 against 0.141). A shared-model advantage would lower the Gemini judges' scores under the Sonnet labels, but on the mapped traces the prompted baseline rises (macro-F1 0.157 to 0.172), TRAIL falls (0.114 to 0.099) and AEGIS stays level. Sonnet agrees with the human annotators less often than Gemini does (61.7% against 78.7% on the 47 audit items; 26 against 33 of the 39 on which the annotators agree), so we keep the Gemini labels as gold. Disagreements concentrate on adjacent category boundaries, predominantly `api_system` against `environmental`, and are not spread across the taxonomy. The classifier (Gemini 2.5 Flash) runs at temperature 0 with cached outputs, so the mapping is deterministic and reproducible. We release the audit sheet and the mapping code with the benchmark.

Per-category F1 on mapped and natively labeled traces (Table 14) differs in both directions and by category more than by judge: mapped traces score higher on `reasoning_planning` for the prompted baseline and TRAIL and lower for AEGIS, and lower on `verification_termination` for all three. Composition differs between the slices, so these gaps mix mapping with source; Appendix A.5 tests the lookup itself.

A.5 Mapping Sensitivity

Every number in Table 1 depends on the L3-to-L2 lookup, and several assignments are contestable. We re-derive the gold labels under alternative assignments for five such labels and re-score all three judges. Table 5 reports the largest and mean absolute change in per-category F1 across the thirty judge-category cells, and how many of the ten categories change which judge scores highest.

Table 5: Sensitivity of Table 1 to contestable L3-to-L2 assignments.

Alternative assignment	**max $\lvert\Delta$F1$\rvert$**	**mean $\lvert\Delta$F1$\rvert$**	**winner changes**
resource_exhaustion → api_system	0.009	0.002	0/10
resource_abuse → api_system	0.009	0.002	0/10
llm_limit → verification_termination	0.009	0.002	0/10
observation_alignment → tool_calling	0.033	0.003	0/10
outcome_misinterpretation → hallucination	0.011	0.002	0/10
all five at once	0.028	0.004	0/10

No single reassignment moves any per-category F1 by more than 0.033, applying all five together moves no cell by more than 0.028, and the judge that scores highest changes in none of the ten categories; only three categories have a separated winner (Table 13), so the check is informative mainly for those. The conclusions in Table 1 are therefore properties of the judges, not of our clustering decisions.

A.6 Provenance of Clean (`ideal`) Traces

Table 6 gives the source of every trace labeled `ideal`, together with the procedure that established the label. The clean-trace false-error rate is the paper's headline result, and the evidence behind a clean label differs in strength by source.

A.7 Results on the Balanced Core

The balanced core of Section 3.4 is a first pass: 180 traces, 29 in every L2 category, 20.8% of the token cost, and a strict subset of the benchmark, so a judge evaluated on it is brought to the full 846 by running the remaining 666 traces. Table 7 scores the four judges of Table 1 on it by restricting the same predictions to the core; no judge was re-run. Macro-F1 moves by up to 0.04: the Prompted LLM and TRAIL keep the top two places, but AEGIS and the ensemble swap the last two. Individual cells move by up to 0.21 at 29 traces of support (AEGIS on `ideal`, 0.610 to 0.405), so the core does not resolve categories. Clean-trace false-error rests on 29 clean traces: the point estimates sit 3 to 14 points above the full set and the intervals are roughly two and a half times wider. Equal category

Table 6: Provenance of the 169 traces labeled `ideal`. Every one was also confirmed clean by two human annotators (Section 3.3).

Source	N	Procedure establishing the clean label
Exgentic	99	Review interface: LLM-proposed candidate, accepted, edited, or discarded by one of three annotators (Section 3.3).
ARB/WebArena	29	Native ARB expert trajectory annotation; admitted where `trajectory_success` is `successful`.
ARB/WorkArena	11	Review interface, as above.
SWE-bench	18	Review interface, as above.
Pumpkin	11	Native metric-based: every Pumpkin judge metric reads ideal (`all_metrics_ideal`).
NLiLe	1	Review interface, as above.
Expert or metric provenance	*40*	ARB/WebArena and Pumpkin.
Annotator provenance	*129*	Exgentic, SWE-bench, ARB/WorkArena, NLiLe.
Total	**169**	

frequencies also favor a judge that asserts most categories, which is why the ensemble is the one judge whose macro-F1 rises.

Table 7: The four judges on the 180-trace balanced core, with the full-set value of Table 1 in brackets. Per-category F1; macro-F1 over the ten categories; clean-trace false-error with a 95% trace-level bootstrap interval (2,000 resamples); traces with parseable output.

L2 category	Prompted LLM	TRAIL	AEGIS	Ensemble
`reasoning_planning`	0.351 (0.349)	0.329 (0.336)	0.172 (0.231)	0.000 (0.000)
`hallucination`	0.225 (0.184)	0.178 (0.202)	0.000 (0.000)	0.260 (0.204)
`instruction_following`	0.263 (0.348)	0.269 (0.359)	0.259 (0.351)	0.266 (0.320)
`tool_calling`	0.345 (0.382)	0.386 (0.427)	0.000 (0.000)	0.265 (0.305)
`mas_coordination`	0.375 (0.476)	0.000 (0.000)	0.289 (0.306)	0.284 (0.293)
`context_state`	0.244 (0.212)	0.047 (0.054)	0.279 (0.226)	0.275 (0.243)
`verification_termination`	0.129 (0.228)	0.247 (0.305)	0.270 (0.351)	0.257 (0.331)
`environmental`	0.190 (0.322)	0.270 (0.175)	0.000 (0.000)	0.277 (0.227)
`api_system`	0.489 (0.337)	0.413 (0.257)	0.000 (0.000)	0.241 (0.094)
`ideal`	0.229 (0.409)	0.566 (0.628)	0.405 (0.610)	0.000 (0.000)
Macro-F1	0.284 (0.325)	0.271 (0.274)	0.167 (0.208)	0.213 (0.202)
Clean-trace false-error	86.2% [71.4, 96.9] (73.5%)	48.3% [30.3, 65.5] (45.0%)	41.4% [24.0, 60.0] (27.2%)	100.0% [100.0, 100.0] (100.0%)
Traces covered	171 (818)	174 (822)	180 (846)	177 (840)

A.8 Per-Category Bootstrap Intervals

Table 1 reports point estimates. Table 8 gives the corresponding F1 with 95% trace-level bootstrap intervals (2,000 resamples) for the three established judges, together with the support each category carries.

A.9 Random-Judge Reference

One-vs-rest F1 rewards prediction volume. A judge that asserts a category on every trace has recall 1 and precision equal to the category's frequency without reading the trace, and the four judges assert very different volumes: 2.6, 1.9, 1.8 and 7.9 labels per trace for the Prompted LLM, TRAIL, AEGIS and the ensemble. Table 9 therefore reports a random judge at fixed volume k. For each trace it draws k of the ten L2 categories uniformly without replacement, `ideal` included; a fractional k alternates between $\lfloor k \rfloor$ and $\lceil k \rceil$ so that the mean is exact. It is scored on all 846 traces and averaged over 200 draws. For the error categories precision is the category frequency at every k and recall is $k/10$, so F1 grows with k and is capped by frequency: the rarest category, `api_system`, never exceeds 0.109. Table 1 carries the $k = 1.5$, 3 and 7.5 columns, which bracket the four judges.

A draw that contains `ideal` alongside an error label is scored as flagging the trace, since a judge outputs `ideal` only when it reports no error (Section 4.1); so the random judge recognizes a clean

Table 8: Per-category F1 with 95% bootstrap intervals. Support is the number of traces carrying the category. Zero-width intervals at zero are scheme-defined zeros (◇ in Table 1).

L2 category	Sup.	Prompted LLM	TRAIL	AEGIS
reasoning_planning	165	0.349 [0.30, 0.39]	0.336 [0.28, 0.39]	0.231 [0.17, 0.29]
hallucination	106	0.184 [0.13, 0.24]	0.202 [0.13, 0.28]	0.000 [0.00, 0.00]
instruction_following	163	0.348 [0.30, 0.40]	0.359 [0.30, 0.41]	0.351 [0.29, 0.40]
tool_calling	156	0.382 [0.31, 0.45]	0.427 [0.37, 0.48]	0.000 [0.00, 0.00]
mas_coordination	146	0.476 [0.41, 0.54]	0.000 [0.00, 0.00]	0.306 [0.25, 0.36]
context_state	119	0.212 [0.16, 0.27]	0.054 [0.01, 0.10]	0.226 [0.17, 0.28]
verification_termination	166	0.228 [0.17, 0.28]	0.305 [0.24, 0.36]	0.351 [0.29, 0.41]
environmental	109	0.322 [0.23, 0.41]	0.175 [0.09, 0.26]	0.000 [0.00, 0.00]
api_system	49	0.337 [0.21, 0.45]	0.257 [0.17, 0.34]	0.000 [0.00, 0.00]
ideal	169	0.409 [0.32, 0.49]	0.628 [0.56, 0.69]	0.610 [0.55, 0.66]

Table 9: A random judge asserting k labels per trace, drawn uniformly from the ten L2 categories including ideal: per-category F1, macro-F1 and clean-trace false-error, averaged over 200 draws on all 846 traces.

L2 category	1.5	3	4.5	6	7.5	9
reasoning_planning	0.171	0.235	0.273	0.294	0.311	0.320
hallucination	0.136	0.178	0.196	0.206	0.214	0.221
instruction_following	0.166	0.235	0.271	0.291	0.307	0.318
tool_calling	0.166	0.228	0.260	0.282	0.296	0.307
mas_coordination	0.160	0.220	0.249	0.269	0.279	0.290
context_state	0.145	0.190	0.212	0.228	0.236	0.243
verification_termination	0.170	0.238	0.272	0.295	0.312	0.322
environmental	0.138	0.177	0.199	0.211	0.219	0.226
api_system	0.083	0.098	0.102	0.106	0.109	0.109
ideal	0.079	0.000	0.000	0.000	0.000	0.000
Macro-F1	0.141	0.180	0.203	0.218	0.228	0.235
Clean-trace false-error (%)	95.0	100.0	100.0	100.0	100.0	100.0

trace only when it draws `ideal` alone, and its clean-trace false-error is 95% at $k = 1.5$ and 100% from $k = 3$. At the prompted baseline's 73.5%, false alarms outnumber even perfect error detection once more than 58% of the executions a judge sees are valid. Read against the ladder, the ensemble's macro-F1 of 0.202 at 7.9 labels per trace lies below the random 0.228 at $k = 7.5$: it tracks the random judge to within 0.02 on the eight categories it can express and scores zero on the two it cannot, and its 100% clean-trace false-error equals the random judge's at that volume. The three established judges sit far below the ladder on clean traces. A sharper reference says "clean" on the same share of traces as the judge, chosen without reading them: it scores 94.0% false-error against the prompted baseline's 73.5%, 84.5% against TRAIL's 45.0% and 72.3% against AEGIS's 27.2%, so every established judge discriminates clean from error traces, the prompted baseline with the most conservative threshold. A stricter null for macro-F1 that keeps each judge's own per-trace label count gives the same ordering: lifts over it are $+0.12$, $+0.09$, $+0.03$ and -0.03 for the Prompted LLM, TRAIL, AEGIS and the ensemble.

A.10 L1 Judge Results

The L1 abstraction groups the ten L2 categories into five classes. Table 10 reports judge performance at that level.

Table 10: L1 per-category performance with 95% bootstrap intervals on F1, and macro-F1 at both taxonomy levels. AEGIS scores zero on `infrastructure` because both of its L2 members are scheme-defined zeros. The ranking is not stable across levels: the Prompted LLM leads on L2 macro-F1 and TRAIL on L1.

Judge	L1 class	Sup.	P	R	F1 [95% CI]
Prompted LLM	`cognitive`	371	0.496	0.922	0.645 [0.61, 0.68]
	`interaction`	306	0.452	0.696	0.548 [0.50, 0.59]
	`procedural`	159	0.213	0.245	0.228 [0.17, 0.29]
	`infrastructure`	139	0.556	0.504	0.528 [0.46, 0.59]
	`clean`	166	0.898	0.265	0.409 [0.33, 0.49]
TRAIL	`cognitive`	365	0.527	0.775	0.627 [0.59, 0.66]
	`interaction`	306	0.423	0.490	0.454 [0.41, 0.50]
	`procedural`	159	0.272	0.346	0.305 [0.24, 0.36]
	`infrastructure`	148	0.444	0.480	0.461 [0.39, 0.53]
	`clean`	169	0.732	0.550	0.628 [0.56, 0.69]
AEGIS	`cognitive`	380	0.582	0.532	0.556 [0.51, 0.60]
	`interaction`	320	0.452	0.622	0.524 [0.48, 0.56]
	`procedural`	166	0.295	0.434	0.351 [0.29, 0.41]
	`infrastructure`	154	0.000	0.000	0.000 [0.00, 0.00]
	`clean`	169	0.526	0.728	0.610 [0.56, 0.67]
Prompted LLM	macro-F1			L2 0.325	L1 0.472
TRAIL	macro-F1			L2 0.274	L1 0.495
AEGIS	macro-F1			L2 0.208	L1 0.408

A.11 Ensemble Aggregation Ablation

A 100% clean-trace false-error rate suggests that splitting a judge into specialized sub-judges does harm in itself. An ablation over the same sub-judge outputs locates the cause elsewhere. The configuration reported in Table 1 flags a category whenever the corresponding sub-judge runs, whatever verdict that sub-judge returns, and it offers no abstention path. No clean trace can survive such a rule, so 100% is what the aggregation guarantees, not what ensembling costs.

We therefore re-aggregate the identical sub-judge outputs on their verdicts and let the ensemble abstain when no sub-judge reports a failure. Macro-F1 rises from 0.202 to 0.305 and the clean-trace false-error rate falls from 100% to 58%, with no sub-judge retrained, re-prompted or replaced; only the aggregation rule changed. The gain has to be read against the baselines: L2 macro-F1 is 0.325 for the prompted LLM, 0.274 for TRAIL and 0.208 for AEGIS (Appendix A.10), so re-aggregation lifts the ensemble past two established judges but not past the simplest one, and it is a post-hoc

adjustment the baselines did not receive. `reasoning_planning` stays at zero under both rules, which matches the structural account above: the deployed configuration has no planning sub-judge, so no aggregation rule can recover the category.

The ensemble also shares a metric family with Pumpkin, which contributes 85 of the 846 traces with gold labels derived from that suite's native outputs (Table 4). It does not appear to benefit: removing the Pumpkin-sourced traces moves the re-aggregated ensemble's macro-F1 by 0.010 and its clean-trace false-error by 0.4 points, and it scores no better on those traces than on the rest (macro-F1 0.289 against 0.295; clean-trace false-error 63.6% against 57.6%). The most likely reason is that the labels were produced by a different configuration, on a different backbone and with a planning sub-judge this one lacks.

A decomposed judge therefore needs an explicit abstention mechanism before it meets a stream containing valid executions; the cost of omitting one falls almost entirely on clean traces.

A.12 Length Effect Within a Single Source

Table 11: Clean-trace false-error (%), short (deciles 3–5) → long (6–9), computed inside one source so that annotation provenance is held fixed. No intervals: $n \leq 63$ per cell. Two-sided Fisher exact p: Exgentic $< 10^{-4}$ for TRAIL and the Prompted LLM; ARB/WebArena 0.06 (AEGIS), 0.24 (TRAIL), 0.10 (Prompted LLM).

Judge	**Exgentic** ($63 \to 36$)	**ARB/WebArena** ($10 \to 19$)
AEGIS	0.0 → 0.0	30.0 → 68.4
TRAIL	11.1 → 69.4	40.0 → 68.4
Prompted LLM	42.9 → 97.2	70.0 → 94.7

A.13 Judge Diagnostics

A low clean-trace false-error rate means little if a judge reports no error on a source regardless of content. Table 12 gives, for each source, the share of *error* traces on which each judge reports none. AEGIS reports no error on every Exgentic and TRAIL-sourced error trace and on 25 of the 26 collected ones, so its clean-trace rate on those sources carries no information about judgment. Its strongest categories also lean on its own source: without its 123 traces, AEGIS's `verification_termination` F1 falls from 0.351 to 0.272 and `mas_coordination` from 0.306 to 0.155. Without Exgentic, which holds 99 of the 169 clean traces, clean-trace false-error is 65.7% for AEGIS, 62.9% for TRAIL and 89.6% for the prompted baseline.

Table 12: Share of error traces on which each judge reports no error, by source.

Source	**Error traces**	**Prompted LLM**	**TRAIL**	**AEGIS**
`aeb`	27	0%	0%	4%
`aegis`	123	1%	6%	6%
`aftraj`	61	3%	12%	16%
`agentracer`	14	0%	0%	7%
`agentrx`	73	3%	1%	10%
`arb_assistantbench`	17	0%	0%	12%
`arb_webarena`	13	15%	0%	8%
`arb_workarena`	10	0%	0%	0%
`custom`	26	0%	0%	96%
`exgentic`	15	0%	13%	100%
`nlile`	54	0%	0%	4%
`pumpkin`	65	2%	6%	6%
`swebench`	18	0%	0%	22%
`trace_elephant`	31	0%	0%	6%
`trail`	20	0%	17%	100%
`who_and_when`	80	1%	8%	6%

Table 13 tests whether the best judge in each category is separated from the second: a paired trace-level bootstrap (1,000 resamples) on the 803 traces all three established judges cover. Only three categories have a separated winner. Macro-F1 over the categories a judge can express, which removes scheme-defined zeros, is 0.346 for AEGIS (six categories), 0.325 for the prompted baseline (ten) and 0.305 for TRAIL (nine).

Table 13: Best minus second-best judge F1 per category, paired bootstrap 95% interval; $*$ marks an interval above zero.

L2 category	Top two	F1 difference [95% CI]
reasoning_planning	Prompted − TRAIL	+0.017 [-0.033, +0.072]
hallucination	TRAIL − Prompted	+0.024 [-0.058, +0.098]
instruction_following	TRAIL − AEGIS	+0.011 [-0.044, +0.068]
tool_calling	TRAIL − Prompted	+0.035 [-0.031, +0.107]
mas_coordination	Prompted − AEGIS	+0.164 [+0.101, +0.234] $*$
context_state	AEGIS − Prompted	+0.017 [-0.051, +0.083]
verification_termination	AEGIS − TRAIL	+0.058 [-0.019, +0.137]
environmental	Prompted − TRAIL	+0.167 [+0.056, +0.278] $*$
api_system	Prompted − TRAIL	+0.115 [+0.019, +0.218] $*$
ideal	AEGIS − TRAIL	+0.004 [-0.063, +0.069]

Table 14 splits per-category F1 by how the gold labels were produced: native structured labels (AEGIS, AEB, TRAIL, AgentRx, Pumpkin, ARB/WebArena), LLM-mapped free text (AFTraj, AgenTracer, TraceElephant, Who&When) and the traces labeled in the review interface (Exgentic, NLiLe, SWE-bench, collected, ARB/WorkArena, ARB/AssistantBench); cells with fewer than ten positive traces are omitted. Both the prompted baseline and TRAIL score higher on api_system in the review-interface slice, where failures come from natural runs and are often explicit, and TRAIL also scores higher on environmental; with LLM-drafted labels, LLM-to-LLM agreement is a second possible cause that this split cannot separate.

Table 14: Per-category F1 by label source: native (nat.), LLM-mapped (map.), review interface (rev.).

	Prompted LLM			TRAIL			AEGIS		
L2 category	nat.	map.	rev.	nat.	map.	rev.	nat.	map.	rev.
api_system	0.05	–	0.64	0.13	–	0.49	0.00	–	0.00
environmental	0.28	0.38	0.34	0.04	0.19	0.26	0.00	0.00	0.00
hallucination	0.13	0.28	0.18	0.11	0.29	0.20	0.00	0.00	0.00
reasoning_planning	0.29	0.52	0.28	0.32	0.36	0.34	0.30	0.16	0.13
verification_termination	0.14	0.07	0.42	0.23	0.04	0.47	0.38	0.04	0.57
ideal	0.21	–	0.47	0.45	–	0.73	0.29	–	0.76

Table 16 compares short and long traces on both sides. On error traces the share with at least one error flagged barely moves with length, while the number of errors flagged on clean traces rises sharply, so the pooled AUROC of that count as a score separating error from clean traces falls for every judge. Pooling mixes sources, and Exgentic supplies 81% of short clean traces, so the table repeats the comparison without it: on 15 short and 52–55 long clean traces, the drop remains for AEGIS (0.82 to 0.67, with intervals that just separate), points the same way for the prompted baseline (0.85 to 0.72, with overlapping intervals), and does not appear for TRAIL (0.77 to 0.80).

Exgentic also dominates the clean set: 99 of the 169 clean traces. Its traces are short-horizon tasks (tau2 retail and telecom, AppWorld, BrowseComp-Plus) run mostly with strong models (GPT-4.1, Gemini 3 Pro, Claude Opus 4.5), and neither auditor flagged an error in any of them, against about 9% elsewhere. For a judge that does not engage with Exgentic, the clean-trace evidence therefore rests on the other 70 clean traces.

Inside Exgentic, length follows the task environment (Table 15). The short clean traces are almost all τ^2 retail dialogues and the long ones AppWorld and BrowseComp-Plus tasks, so the significant Exgentic split in Section 5.4 compares environments as much as lengths. The one environment with clean traces on both sides, τ^2 telecom, points the same way on 11 traces.

Table 15: Exgentic clean traces by environment and length: false errors / clean traces.

Environment	Deciles	Prompted LLM	TRAIL	AEGIS
τ^2 retail	3–5	26/57	7/57	0/57
τ^2 retail	6–9	1/1	0/1	0/1
τ^2 telecom	3–5	1/6	0/6	0/6
τ^2 telecom	6–9	5/5	4/5	0/5
AppWorld	6–9	21/22	16/22	0/22
BrowseComp-Plus	6–9	8/8	5/8	0/8

The audit also bounds the errors both auditors missed. The two flagged 18 and 17 of the 197 traces they could judge, 7 in common; a Lincoln–Petersen capture–recapture estimate puts the total at about 44, so about 16 were missed by both, roughly 9% of the 169 retained clean traces. The estimate is biased low if some errors are hard for both readers.

Table 16: Detection and discrimination by length. AUROC of the number of errors flagged, error traces against clean traces; bootstrap 95% interval.

Judge	Sources	Deciles	Flags any (error)	Errors/trace (error)	Errors/trace (clean)	AUROC [95% CI]
Prompted	all	3–5 (n_c=78)	99.2%	3.04	0.71	0.928 [0.897, 0.957]
		6–9 (n_c=88)	99.3%	3.07	2.09	0.714 [0.655, 0.774]
	w/o Exgentic	3–5 (n_c=15)	99.1%	3.05	1.27	0.850 [0.753, 0.939]
		6–9 (n_c=52)	99.2%	3.10	2.06	0.716 [0.633, 0.806]
TRAIL	all	3–5 (n_c=78)	95.8%	1.86	0.24	0.922 [0.883, 0.957]
		6–9 (n_c=91)	97.4%	2.45	1.22	0.781 [0.725, 0.833]
	w/o Exgentic	3–5 (n_c=15)	95.8%	1.87	0.80	0.774 [0.631, 0.904]
		6–9 (n_c=55)	98.1%	2.48	1.15	0.802 [0.733, 0.866]
AEGIS	all	3–5 (n_c=78)	86.8%	1.97	0.12	0.911 [0.883, 0.938]
		6–9 (n_c=91)	80.6%	1.70	0.67	0.749 [0.694, 0.802]
	w/o Exgentic	3–5 (n_c=15)	87.5%	1.99	0.60	0.822 [0.746, 0.887]
		6–9 (n_c=55)	84.4%	1.79	1.11	0.671 [0.597, 0.742]

A.14 Release and Privacy

The 26 collected traces come from natural runs. Five are OpenClaw agent runs (Claude Opus 4.6) in a fresh sandbox on benchmark-style tasks, recorded as the raw request and response log; the other 21 are Langfuse exports of multi-agent pipeline runs, mostly on GAIA tasks, and as with Pumpkin each carries its full execution record. We read all 26 for identifying or third-party content: the OpenClaw workspace files are the unfilled default templates, and the remaining content is public task material and fetched web pages. Traces are released as recorded, with one exception. Before release we scanned all 846 traces for provider API keys, cloud and repository tokens, private-key blocks, bearer and JWT tokens, credential-like key–value pairs, e-mail addresses and home-directory paths (`scripts/scan_secrets.py`). Every match was reviewed. One was a live API key inside a tool call in a collected trace; it is replaced by a placeholder in the released copy. The remaining matches are synthetic test accounts and tokens from the source environments (AppWorld, `tau2`, planted credentials in AFTraj tasks), placeholder values, public repository metadata such as commit author addresses, and false positives inside URLs and task names, all already present in the public source releases. Each trace's license and terms are recorded in the manifest (`license` and `terms` columns): MIT, CC BY 4.0 or CDLA-Permissive-2.0 for most sources, the TRAIL and GAIA condition that traces be shared only through a gated or private repository, the AgentRewardBench terms of use, and four sources with no stated license, included for research use while we seek permission.

A.15 Extended Limitations

Taxonomy and coverage. The L2 taxonomy was built by semantic clustering, which involves judgment calls; the mapping is documented in full, was reviewed by multiple authors, and we invite community revision. Coverage is deliberately broad and not representative of any production distribution, so label frequencies should not be read as deployment error rates.

Ensemble judge. The ensemble is evaluated without an abstention mechanism. Its results reflect a lower bound on precision and an upper bound on false-error rate; it is a diagnostic, not a finished system.

Clean-trace labels. The `ideal` category carries the headline false-error result. Its labels first came from several procedures: of the 169 clean traces, 40 rest on expert or metric-based provenance (29 from ARB/WebArena, which carries expert trajectory annotations, and 11 from Pumpkin, where every native metric reads ideal) and 129 were labeled through the review-interface protocol of Section 3.3 (Table 6). Every one was then confirmed by two human annotators, and the 31 traces they did not both confirm were removed. Task success alone does not establish that a trace is error-free, since a trace can reach the right answer through a faulty intermediate step, and two readers can miss the same error. Residual label noise would mean that some traces marked clean in fact contain errors. Such noise charges a judge for catching a real error, so it raises false-error most for the judges likeliest to catch hidden errors, and where it concentrates in long traces it inflates the length gap (Section 5.4). The absolute rates carry that uncertainty. Agreement between the two annotators was fair ($\kappa = 0.34$ at a low error rate), so a third reader on the confirmed traces would tighten this.

Mapped labels. For four sources, gold labels come from an LLM classifier reading human-written `mistake_reason` text, not from direct human categorization. The audit in Section 3.3 bounds the resulting noise, but these labels remain one step removed from native annotations, and they are less certain than native ones: two strong mappers agree on only 69.4% of the texts (Appendix A.4).

Length and source. The clean traces available at each length are not drawn evenly from the sources, so the pooled length effect in Section 5.4 mixes length with composition. We control for this with source fixed effects and by repeating the split within single sources: the effect holds for the prompted baseline and TRAIL, including significant Exgentic splits, while the ARB/WebArena splits point the same way without individual significance and AEGIS shows no effect on Exgentic. Inside Exgentic, length also coincides with task environment (Table 15), so the within-source evidence does not separate length from task type. A larger clean set spanning more sources at every length would tighten this.

Backbone scope. All four judges are compared on a single backbone. The backbone transfer in Table 2 covers only the prompted baseline, so we can claim that clean-trace over-flagging persists across model families but not that the ordering between judge designs does.