

Localizing failures in AI agent pipelines with structured execution traces

Alexey Zabegin* *Senior AI Engineer, HautAI, Tallinn, Estonia*

* Corresponding author: Alexey Zabegin, Zabaled@gmail.com

OPEN ACCESS

Citation:

Alexey Zabegin (2026). Localizing failures in AI agent pipelines with structured execution traces. *Am. Impact Rev.* e69362342. [10.66308/air.e69362342](https://doi.org/10.66308/air.e69362342)

Received: April 9, 2026

Accepted: April 27, 2026

Published: April 30, 2026

DOI:

[10.66308/air.e69362342](https://doi.org/10.66308/air.e69362342)

ISSN: 3071-124X

Copyright:

© 2026 Alexey Zabegin. This is an open access article distributed under the terms of the Creative Commons Attribution License (CC BY 4.0).

Abstract

Failures in applications built around large language models may emerge several steps after an incorrect decision, making final responses inadequate for diagnosis. This study evaluates whether preserving the identity of execution-trace fields improves lightweight failure localization beyond their lexical content. A frozen public TraceElephant dataset was analyzed using source artifacts available by 30 April 2026. The eligible cohort contained 219 failed executions, 5,936 candidate steps, and 173 task groups across three agent systems. Regularized logistic rankers were evaluated using five task-grouped folds, matched text budgets, and a shared training-only vocabulary. Separate-field encoding achieved 24.7% exact Top-1 localization, compared with 23.3% for a flat representation containing the same lexical evidence. The paired difference was 1.4 percentage points (95% task-cluster bootstrap interval -1.9 to 5.2). Top-3 coverage increased from 46.6% to 50.7%, although its advantage varied across sensitivity analyses. When an entire agent system was withheld, separate-field Top-1 accuracy fell to 12.8%. A combined ranker placed the annotated step within the selected quarter of records in 67.6% of traces, compared with 18.3% for selecting the latest records. These findings support structured traces as inputs to inexpensive diagnostic triage, while providing no clear evidence of improved exact localization from field separation alone. The results concern agreement with benchmark annotations and do not establish causal diagnosis or reductions in human debugging time.

Keywords: AI agents, failure localization, execution traces, software engineering, observability, diagnostic triage, reproducible evaluation

1. Introduction

An application built around a large language model (LLM) is more than a sequence of prompts. It may retrieve documents, select tools, validate arguments, update application state, and route control between specialized components. Retrieval-augmented generation and reasoning–action loops make these interactions central to application behavior [1][2]. A plausible final response can therefore coexist with a failed retrieval, an incorrect intermediate interpretation, or an inappropriate tool invocation. For a software engineer, the relevant question is often where the execution first became irrecoverable, rather than whether the final response appears satisfactory.

This distinction matters for customer-facing AI services. Recommendation assistants, catalog-processing

agents, and enterprise support workflows commonly combine model decisions with deterministic services. Their developers need evidence that connects a decision to the input it received and the action it produced. Conventional exception monitoring identifies explicit execution failures, but many agent failures are semantically incorrect actions that complete successfully. Conversely, a visible error may be recovered later and should not automatically be treated as the decisive failure. Automated localization should therefore be evaluated against a defined attribution target rather than equated with searching for error messages.

Agent frameworks make increasingly complex coordination patterns accessible to application developers [3]. However, recording more telemetry and extracting useful diagnostic information are different engineering achievements. A trace can contain repeated conversation histories, transport metadata, and multiple representations of the same operation. Comparing a rich trace with a final answer changes both the amount and the organization of evidence. Such a comparison cannot isolate the incremental value of preserving field identity. A controlled comparison must hold observable content sufficiently constant while changing its representation.

The present study addresses this narrower question through a reproducible empirical reanalysis. It compares lightweight supervised rankers that use output text, flattened input and output fields, explicitly separated fields, or structural execution summaries. The matched comparison uses the same vocabulary, observed text, truncation rule, training folds, and classifier family. It is complemented by positional baselines, a cross-system transfer test, and an analysis of candidate-record selection under limited review budgets. These controls distinguish representation effects from task repetition, architectural regularities, and the advantages of simply supplying additional text.

Three research questions organize the study. RQ1 asks whether separate-field encoding improves exact localization relative to a flat encoding of the same lexical evidence. RQ2 asks how localization changes when the test set contains an agent architecture absent from training. RQ3 asks how often the annotated decisive step remains in a limited shortlist selected by a learned ranker or by simple chronological rules. The contribution is an empirical assessment of these relationships, including negative and uncertain findings. It is not a new tracing framework, a general theory of agent failure, or a claim of superior performance over contemporary LLM-based diagnostic systems.

2. Related work

Software fault localization traditionally ranks program elements using execution evidence, tests, or diagnostic constraints [4]. Agent workflows introduce a different unit of analysis: a decision or interaction expressed partly in natural language. The same action may have different consequences depending on the information available to the acting component. This motivates a distinction between observing an error, locating a decisive step, and demonstrating that intervening on that step would repair the execution. Static trace classification addresses only part of this chain.

Who&When formalized attribution to a responsible agent and a decisive failure step [5]. TRAIL provided annotated agent traces and a taxonomy covering several forms of reasoning and execution error [6]. MAST emphasized failures arising from system design, coordination, and verification [7]. AgentDebug connected structured failure analysis to corrective feedback [8], while AgentRx used executable constraints and a validation log to support diagnosis [9]. These studies establish failure attribution as an existing research area. They also show why an explicit error indicator is an incomplete proxy for the target considered here.

Two directly relevant studies first appeared in April 2026. TraceElephant examines full execution observability and input/metadata ablations [10]. CodeTracer reconstructs agent state transitions and evaluates failure-onset localization in coding trajectories [11]. Consequently, neither the claim that complete traces are useful nor the proposal to organize traces hierarchically would, by itself, constitute a distinct contribution. The present

analysis instead tests a modest, readily reproducible representation contrast using conventional statistical learning, task-level separation, and explicit architectural transport controls.

Instrumentation research addresses another part of the problem. AgentTrace describes structured agent telemetry, whereas AgentSight connects higher-level agent behavior with system activity [12][13]. These approaches support collection and interpretation of execution evidence. The experiment below does not evaluate either instrumentation system or measure its runtime overhead. Likewise, research on long-context language models demonstrates that adding text does not guarantee effective use of that text [14]. This motivates testing evidence budgets, but its findings are not assumed to transfer directly to the lexical rankers used here.

3. Materials and methods

3.1. Design, source eligibility, and cohort construction

The study is a retrospective analysis of frozen public artifacts. Literature and source-artifact eligibility were restricted to material available by 30 April 2026. This cutoff defines the eligible evidence, rather than the date of the new computations. The primary dataset was TraceElephant, licensed under CC BY 4.0. The analyzed archive was pinned to Hugging Face revision 218e392a4c921d4379f827355b9e1d8147748b82, dated 7 January 2026. Its checksum, source manifest, and reconstruction commands are preserved in Supplementary File S2. The corresponding paper is cited at arXiv version 1, released on 24 April 2026.

The archive supplied 220 annotated failed executions. One record had the literal value “none” instead of a decisive-step identifier and was excluded from exact-step evaluation. A second identifier written as “Step 36” was normalized to integer 36 without changing its meaning. The resulting cohort comprised 219 traces and 5,936 steps. All included target identifiers matched an existing step, and every trace had one target. The supplied actor name disagreed with the actor recorded at the normalized step in one case; the step label was retained, and actor-level accuracy was not reported. No new human annotations or consensus labels were created.

The cohort spans dynamically assembled Captain-Agent teams, Magentic-One orchestration, and SWE-Agent software-engineering workflows. The latter two system designs are described independently in their original publications [15][16]. Task sources include GAIA, AssistantBench, and SWE-bench; GAIA and AssistantBench provide general assistant and web-oriented tasks [17][18]. These are benchmark executions rather than a random sample of deployed products. Only failed traces are represented, so the data cannot estimate operational failure rates or the false-positive rate of a detector applied to successful runs.

Table 1. Eligible traces and candidate steps by agent system. IQR denotes interquartile range.

System	Traces	Steps	Median steps (IQR)
Captain-Agent	85	1746	13 (10–24)
Magentic-One	90	2639	28 (14–46)
SWE-Agent	44	1551	23.5 (19–38.75)
Total	219	5936	21 (12–39.5)

Records sharing a task identifier within the same task source, or an identical whitespace-normalized task instruction, were joined into the same group. The union of these relationships produced 173 groups. This prevented an execution of a repeated task in one system from appearing in training while another execution of that task appeared in the corresponding test fold. All steps of a trace remained together. The analysis plan was recorded locally before model fitting; it was not prospectively registered in an external registry. Figure 1 summarizes the separation of observable execution evidence from attribution labels.

Analysis stages	Design controls
1 Source cohort 220 public runs → 219 eligible 5,936 steps · three agent systems	Frozen source revision Data cutoff: 30 April 2026
2 Task grouping 173 groups · five test folds Repeated tasks remain together	No shared tasks between training and test partitions
3 Matched representations Flat fields vs. separate fields Same input, output/tool and actor text	Training-only vocabulary and scaling Matched text budgets and classifier
4 Held-out attribution Within-trace step ranking Exact Top-1 · Top-3 · MRR	Decisive-step labels are targets Never used as predictor fields
5 Robustness and review Unseen-system transfer Limited-record review budgets	Seven sensitivity settings Task-cluster uncertainty

Figure 1. Analysis workflow. Observable execution fields are processed separately from attribution labels. All repeated executions of a task remain within one fold. Labels supply training targets and evaluation references but are excluded from predictor construction. MRR denotes mean reciprocal rank.

3.2. Observable fields and matched representations

Predictor construction used a fixed allowlist: input messages, output messages and tool-call arguments, tool logs, the recorded actor name, and step order. Annotation explanations, decisive-step identifiers, reference answers, test-outcome metadata, task identifiers, run identifiers, and absolute timestamps were excluded from predictors. Attribution labels were used only as training targets or evaluation references. Image payloads were replaced by explicit placeholders; the experiment therefore evaluates textual evidence. Transport-oriented keys were removed during canonical serialization. The preprocessing code records the exact transformations rather than describing the inputs as an unmodified complete trace.

For each candidate step, input messages formed one text channel. Output messages and tool logs formed a second, and the actor name formed a third. Input and output/tool channels were each limited to 4,000 characters, preserving the first and last 2,000 characters when necessary. The same selected text was used in the flat and separate-field conditions. This limit is a reproducible character budget, not a tokenizer-dependent context limit. Actor names were retained in both conditions. A shared training-fold vocabulary of up to 12,000 unigrams and bigrams was fitted with a minimum document frequency of two and sublinear term frequency.

Let the three term frequency–inverse document frequency (TF-IDF) vectors for a step be the input, output/tool, and actor vectors. The flat condition sums these vectors and normalizes the result. The separate-field condition concatenates the same vectors and normalizes the concatenation. Thus, a term can receive different learned

coefficients depending on its originating field in the separate representation. No additional text is supplied to that condition. The transformation is defined in Equation 1, where N denotes Euclidean normalization and the semicolon denotes concatenation.

$$z_{\text{flat}} = N(v_{\text{in}} + v_{\text{out}} + v_{\text{actor}}), \quad z_{\text{field}} = N([v_{\text{in}}; v_{\text{out}}; v_{\text{actor}}]) \quad (1)$$

All main learned models included four positional controls: relative step position, its square and cube, and the logarithm of one plus trace length. They also included a categorical system indicator. Structural features consisted of input/output/tool text lengths, input-message count, tool-log presence, bounded error-term counts in outputs and tool logs, actor changes, and the occurrence count of the current actor. Numeric features were standardized using training data only. Actor identity was additionally one-hot encoded in the structural and combined models. Table 2 distinguishes evidence content from its encoding.

Table 2. Prespecified learned conditions. Every main condition includes position and system controls.

Condition	Additional evidence and encoding
Output + actor	Output/tool text and actor text, summed lexical vectors.
Flat fields	Input, output/tool, and actor vectors summed before normalization.
Separate fields	The same three lexical vectors concatenated before normalization.
Structural	Nine numeric execution summaries and categorical actor identity.
Combined	Separate lexical fields plus structural summaries and actor identity.

The structured representation has greater parameter capacity than the flat representation because field-specific coefficients are allowed. This is an explicit feature of the comparison, not a controlled-away difference. Both use the same regularization setting and lexical vocabulary, but they are not parameter-count-matched. The experiment therefore estimates the utility of this particular encoding and classifier combination; it does not isolate an abstract effect of “structure” independently of the model consuming it.

3.3. Training and validation

Each step received a binary training target indicating whether it matched the supplied decisive-step label. An L2-regularized logistic model produced a score used to rank steps within each held-out trace. These scores were not interpreted as calibrated probabilities of causal responsibility. Within every training trace, the positive example received weight 0.5 and the negative examples together received weight 0.5. Weights were then rescaled to sum to the number of training steps. This prevented long traces from dominating the objective solely because they supplied more negative candidates.

Equation 2 gives the penalized logistic objective, where i indexes the n training traces, t indexes the T_i steps in trace i , y is the step label, s is the linear score, w is the normalized sample weight, and C controls regularization. The coefficient vector β includes the penalized intercept under the solver’s default synthetic-feature treatment. The implementation used the liblinear solver with $C = 1$, tolerance 10^{-6} , and at most 2,000 iterations. Model fitting used scikit-learn [19]. No pretrained language model, diagnostic LLM prompt, external inference API, or new agent execution was used in the experiment.

$$L(\beta) = \sum_{i=1}^n \sum_{t=1}^{T_i} w_{it} [\ln(1 + e^{s_{it}}) - y_{it} s_{it}] + \frac{\|\beta\|^2}{2C} \quad (2)$$

Five grouped outer folds were constructed with a fixed random seed of 20260430. Vocabularies, inverse document frequencies, scalars, categorical encoders, and classifiers were fitted separately within each training

fold. No hyperparameters were selected against the test folds. Every trace received one out-of-fold ranking per method. Equal scores were resolved by ascending original integer step identifier. Three deterministic baselines ranked the earliest step first, the latest step first, or the greatest output/tool error-term count first. The error-term baseline was intentionally simple and was not a semantic fault detector.

A separate transport experiment withheld each of the three agent systems in turn. All traces from the held-out system were tested, and any training trace sharing a task group with that test set was removed. This evaluates transfer to an unseen architecture and partly different workload. It is harder than the primary task-grouped experiment and changes the amount and composition of training data. The resulting difference is therefore a transport stress test, not a controlled causal estimate of architecture alone.

3.4. Outcomes and uncertainty

The primary outcome was exact Top-1 localization, defined by equality between the highest-ranked integer step identifier and the benchmark target. Top-3 coverage and mean reciprocal rank (MRR) were secondary outcomes. The word “coverage” denotes inclusion of the target in a shortlist, not correctness of a diagnosis attached to that shortlist. Equation 3 defines Top-k accuracy, and Equation 4 defines MRR, where r is the target’s rank and n is the number of evaluated traces. Each trace contributes equally to the point estimates.

$$A_k = \frac{1}{n} \sum_{i=1}^n I(r_i \leq k) \quad (3)$$

$$\text{MRR} = \frac{1}{n} \sum_{i=1}^n \frac{1}{r_i} \quad (4)$$

The primary contrast was separate-field minus flat Top-1 accuracy. Paired differences and descriptive uncertainty intervals were calculated by resampling whole task groups 2,000 times, retaining all traces within each selected group. Percentile 95% intervals were reported [20]. These intervals condition on the fitted out-of-fold predictions; they do not include full uncertainty from repeating model training on every bootstrap sample. This distinction matters for small evaluation cohorts [21]. Secondary comparisons were treated as exploratory, without an omnibus claim of superiority based on whichever metric appeared favorable.

3.5. Review budgets, sensitivity analyses, and verification

The review-budget analysis selected a nominal 10%, 25%, 50%, 75%, or 100% of each trace’s step records, rounding the count upward. Four policies selected the earliest records, latest records, uniformly spaced records, or highest-ranked records from the combined model. Equation 5 defines target inclusion, where q is the nominal budget fraction, t_i^* is the annotated target, $S_i(k)$ is the set of k selected steps, and I is the indicator function. The learned policy used the complete available trace representation to form its ranking. It is therefore a post hoc review aid, not an online logging policy or a demonstrated reduction in telemetry storage. Retained input histories may also contain indirect evidence about unselected steps.

$$\text{Coverage}(q) = \frac{1}{n} \sum_{i=1}^n I(t_i^* \in S_i(\lceil qT_i \rceil)) \quad (5)$$

Sensitivity analyses changed the text cap to 1,000 or 16,000 characters, changed C to 0.1 or 10, repeated the grouped evaluation with two additional seeds, and removed positional controls from the matched pair. These settings were evaluated without selecting a preferred alternative after seeing their outcomes. Independent

checks recomputed all 1,314 learned model–trace ranks using a separate sorting implementation, verified group exclusivity, and tested that altering annotation and identifier columns left feature matrices unchanged. Archive integrity and candidate identifiers were also checked. The scripts, per-trace results, and audit records are supplied in Supplementary File S2.

4. Results

4.1. Cohort and overall localization

The eligible traces contained between 5 and 94 candidate steps, with a median of 21 and an interquartile range of 12–39.5. Captain-Agent contributed 85 traces, Magentic-One 90, and SWE-Agent 44. The task-source counts were 147 GAIA, 28 AssistantBench, and 44 SWE-bench traces. Input histories represented approximately 220.6 million serialized characters, compared with 2.84 million output characters and 0.80 million tool-log characters. These totals include repeated history across steps and must not be interpreted as unique information or inference token usage.

The separate-field model localized 54 of 219 targets exactly, yielding 24.7% Top-1 accuracy. The flat model localized 51 targets, or 23.3%, and the output-plus-actor model localized 42, or 19.2%. Structural features alone achieved 9.6%, whereas the position-plus-system baseline achieved 5.5%. The first-step, last-step, and error-term baselines achieved 5.5%, 2.3%, and 5.0%, respectively. Figure 2 presents the corresponding task-cluster intervals. Explicit error terminology was therefore a weak ranking signal for the supplied decisive-step target.

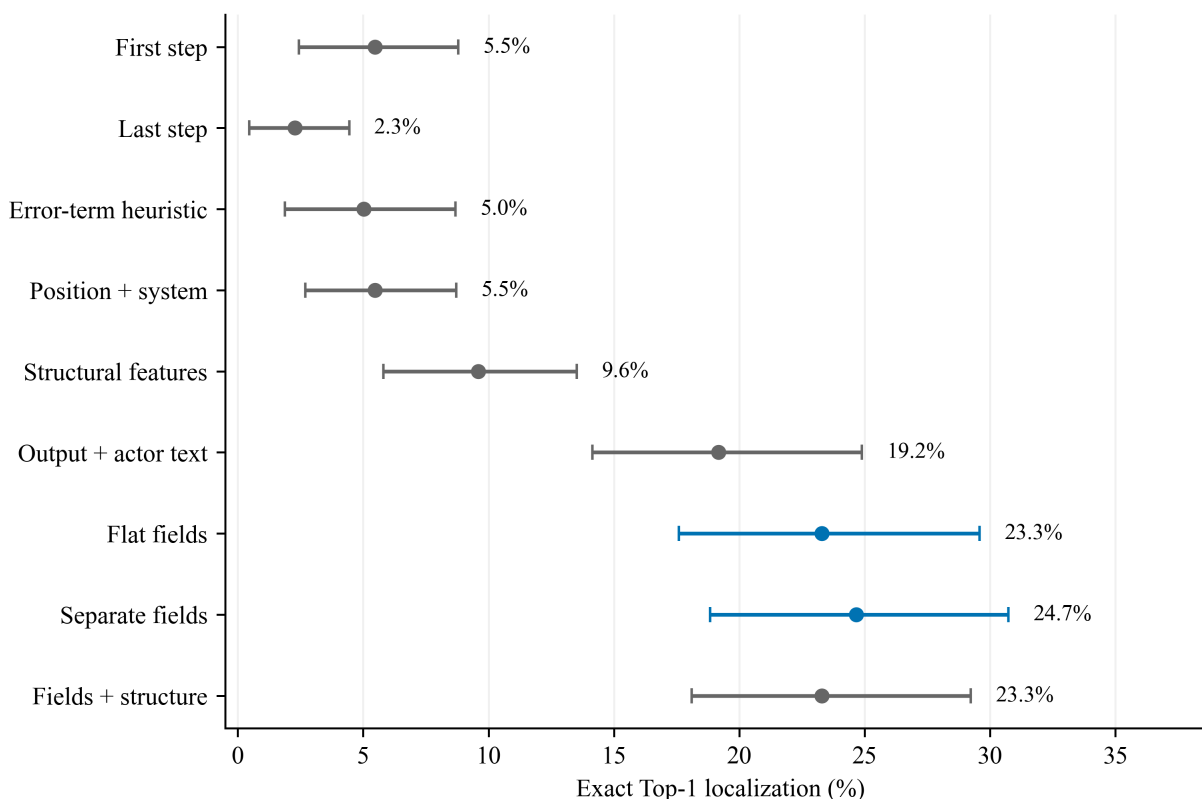


Figure 2. Exact Top-1 localization for 219 held-out traces. Points show trace-weighted accuracy; bars show percentile 95% intervals from 2,000 task-cluster bootstrap samples, conditional on the fitted out-of-fold predictions. The matched flat/separate-field pair is shown in blue.

4.2. Matched representation contrast

For the primary comparison, separate-field encoding improved exact localization by 1.4 percentage points, with a 95% interval from -1.9 to 5.2 points. The interval includes both a small disadvantage and a modest advantage. The study therefore provides no clear evidence that field separation improves exact Top-1 localization under the main configuration. The result should not be converted into an equivalence claim either, because no equivalence margin was specified and the interval remains compatible with effects that might matter in some applications.

The secondary Top-3 comparison was more favorable: 50.7% for separate fields versus 46.6% for flat fields, a difference of 4.1 points with an interval of 0.5 – 7.8 . MRR was 0.418 versus 0.400, with a paired difference interval of -0.002 to 0.040 . Adding structural summaries to separate lexical fields produced 23.3% Top-1 accuracy and 52.5% Top-3 coverage. Thus, the combined model offered the largest observed shortlist coverage among the prespecified learned conditions but did not improve exact Top-1 accuracy over the flat model. Table 3 reports the paired comparisons rather than relying on overlap between separate confidence intervals.

Table 3. Paired differences with 95% task-cluster bootstrap intervals. pp denotes percentage points; MRR denotes mean reciprocal rank. Only the first row is the primary comparison.

Contrast	Outcome	Difference	95% interval
Separate fields – Flat fields	Top-1 (pp)	1.4	–1.9 to 5.2
Separate fields – Flat fields	Top-3 (pp)	4.1	0.5 to 7.8
Separate fields – Flat fields	MRR	0.018	–0.002 to 0.040
Combined – Flat fields	Top-1 (pp)	0.0	–5.1 to 5.2
Combined – Flat fields	Top-3 (pp)	5.9	1.3 to 10.8

4.3. Architectural heterogeneity and transport

Results varied substantially by system. In task-grouped validation, separate-field Top-1 accuracy was 29.4% for Captain-Agent, 17.8% for Magentic-One, and 29.5% for SWE-Agent. The corresponding flat-model values were 29.4%, 18.9%, and 20.5%. The output-only lexical condition was strongest on SWE-Agent at 34.1%, indicating that the added input channel was not uniformly beneficial. These comparisons describe heterogeneous workloads and trace schemas; they do not establish that one architecture is inherently easier to debug.

When entire systems were withheld, overall Top-1 accuracy fell to 12.8% for separate fields and 13.2% for flat fields. Separate-field Top-3 coverage fell from 50.7% to 33.8%. Neither lexical representation localized a SWE-Agent target exactly when trained only on the other two systems, although the structural and combined models each reached 6.8%. Figure 3 makes this transfer limitation visible. Learning across related tasks within observed architectures was materially less demanding than transferring to an unseen coding workflow.

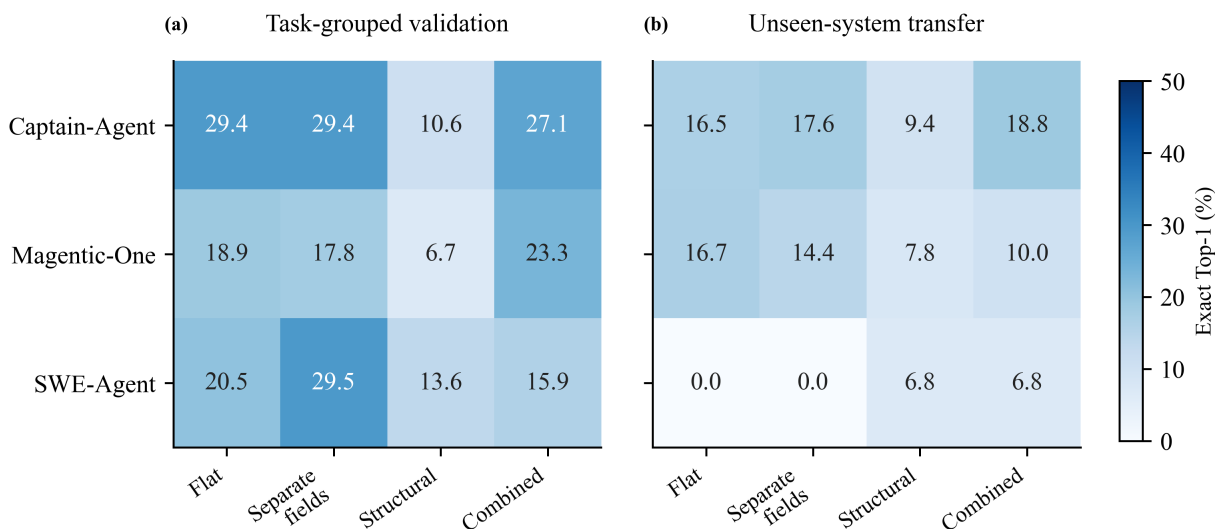


Figure 3. Exact Top-1 accuracy (%) under (a) task-grouped validation and (b) leave-one-system-out evaluation. The held-out systems contain 85, 90, and 44 traces, respectively. Tasks shared with a held-out system were also removed from training. The color scale is identical across panels; cell values are descriptive subgroup estimates.

4.4. Sensitivity and limited review

Sensitivity analyses did not establish a stable exact-localization advantage for field separation. Across the seven alternative configurations, the separate-minus-flat Top-1 difference ranged from -0.5 to 2.7 percentage points. At the 1,000-character cap, both representations reached 26.0% . At 16,000 characters, they reached 20.1% and 19.6% , respectively. Increasing the amount of retained text therefore did not monotonically improve these lexical models. Table 4 reports the sensitivity settings without using them to replace the original primary result.

Table 4. Sensitivity of the matched comparison. Top-1 values are percentages; differences are separate fields minus flat fields in percentage points.

Setting	Flat Top-1	Fields Top-1	Δ Top-1	Δ Top-3
1,000-character cap	26.0	26.0	+0.0	+0.9
16,000-character cap	19.6	20.1	+0.5	+1.4
C = 0.1	18.3	20.5	+2.3	+0.0
C = 10	20.1	19.6	-0.5	+6.8
Partition seed + 1	20.5	23.3	+2.7	+0.0
Partition seed + 2	22.4	25.1	+2.7	+3.2
No positional controls	21.9	22.8	+0.9	+2.7

The Top-3 difference was zero in two sensitivity settings and positive in the others. This weakens any interpretation of the favorable main Top-3 interval as a general representation advantage. The experiment instead suggests that field identity can be useful for shortlist construction under some configurations, while the magnitude depends on regularization, evidence budget, and partitioning. Larger independent cohorts are needed to determine whether the effect persists beyond these benchmark tasks.

At a nominal quarter of the records, the combined-model shortlist contained the annotated target in 67.6% of traces (95% interval $61.7\text{--}73.6$). Earliest-record selection achieved 33.8% , uniformly spaced selection 26.0% , and latest-record selection 18.3% . Upward rounding meant that the actual mean selected fraction was 27.5% . At a nominal half of the records, ranked coverage was 85.4% , compared with 42.9% for latest-record selection. Figure 4 shows that reviewing only the end of a failed run frequently excludes the explicitly annotated target record.

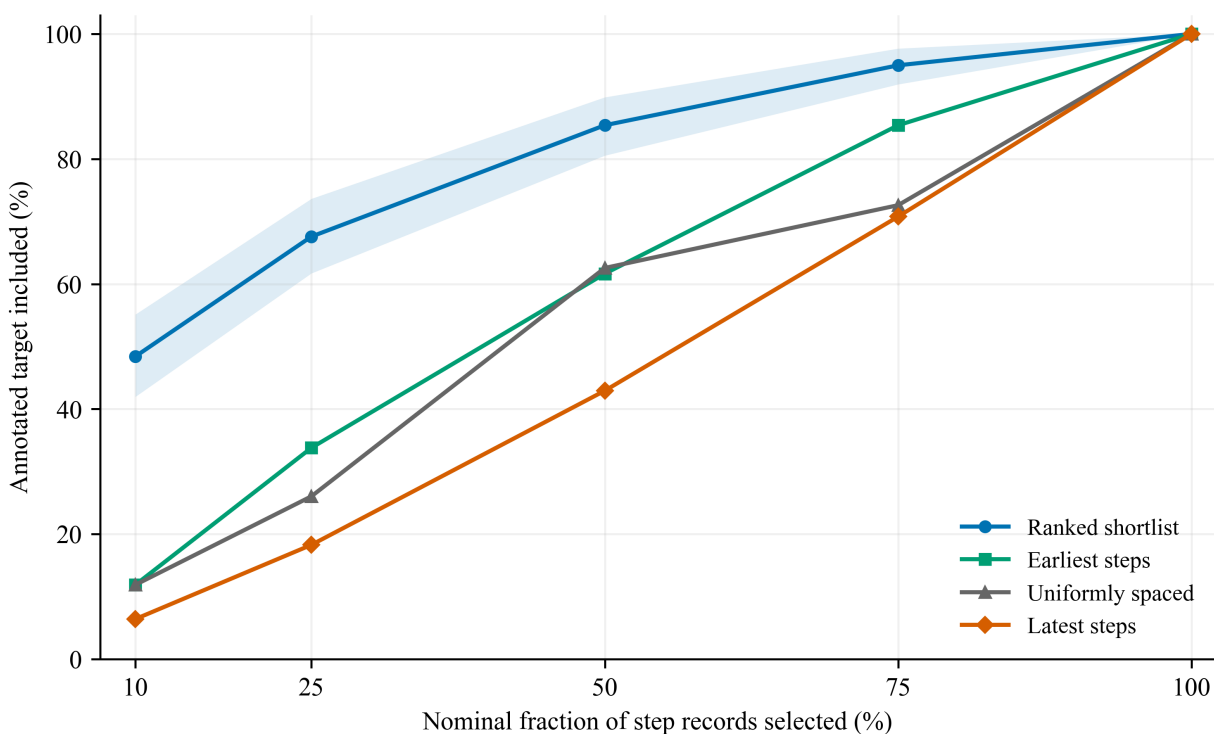


Figure 4. Inclusion of the annotated target record under limited post hoc review budgets. Counts are rounded upward within each trace. The shaded interval is the 95% task-cluster interval for the combined-model shortlist. Selection occurs after complete-trace scoring and does not measure storage savings, human effort, or diagnostic correctness.

5. Discussion

5.1. Interpretation for AI application engineering

The main result is a distinction between content availability and representation utility. Models with lexical execution evidence substantially exceeded the positional and structural-only baselines, but preserving field identity produced only a small, uncertain improvement in exact localization. Structured capture remains necessary for reconstructing what a component observed and produced; the experiment does not challenge that engineering purpose. It shows that structure alone should not be assumed to translate into a large diagnostic gain for every downstream model.

Published results provide context for the observed 24.7% Top-1 accuracy. In version 1 of the TraceElephant preprint, Chen et al. report “All-avg” step accuracies of 22.7%, 26.1%, and 27.6% for GPT-4o All-at-Once, Static Agentic, and Dynamic Agentic attribution without access to the reference task outcome; with that outcome, the corresponding values are 28.1%, 30.3%, and 33.3% [10, Table 2]. These are contextual comparisons, not a common leaderboard. Their All-avg results summarize system-level percentages for a 220-trace benchmark, whereas the present 24.7% estimate weights 219 eligible traces equally and uses task-grouped cross-validation. The attribution procedures and available evidence differ; the dynamic method additionally uses replay and counterfactual probes. No published method was rerun, and no state-of-the-art claim is made. Supplementary File S1 records the comparison protocols.

The shortlist findings support a narrower use case. A local ranker may help an engineer decide where to begin inspecting a failed pipeline. A system that surfaces several plausible steps can be useful even when

its first suggestion is wrong, provided the interface presents candidates rather than an asserted root cause. The observed 52.5% Top-3 coverage of the combined model also leaves almost half the targets outside that shortlist. A review workflow must therefore allow expansion to the complete trace and inspection of upstream inputs, rather than making the shortlist an irreversible filter.

5.2. Why transport and evidence budgets matter

The large transfer degradation indicates that a localizer can learn regularities tied to the systems represented in its training set. Actor names, characteristic outputs, and the relative location of particular actions are useful predictors without necessarily encoding a transferable explanation of failure. The different results for SWE-Agent are especially cautionary: a representation that performed adequately when coding traces appeared in training failed to identify any target exactly when those traces were absent. This does not invalidate the within-system results; it limits their deployment interpretation.

Evidence budgeting introduces a separate tradeoff. Additional input history may repeat task descriptions and earlier messages, changing term frequencies without adding discriminative local information. The larger text-cap sensitivity is consistent with that possibility, but the experiment did not isolate repetition as a causal mechanism. It also did not test an LLM's ability to reason over long contexts. An appropriate practical response is to measure the behavior of the intended diagnostic model under realistic trace formats and budgets, rather than assuming that either maximal retention or aggressive truncation is universally preferable.

The review-budget curves should likewise be interpreted as record-selection results. The learned policy has already consumed the trace to construct its ranking. Its selected fraction is not a measured saving in capture overhead, storage, prompt tokens, or human effort. Moreover, earlier decisions can be copied into later input histories. The reported coverage counts explicit target records within the review set and does not establish that other sets lack all evidence needed for diagnosis. These distinctions are necessary when translating an offline benchmark into observability-product claims.

5.3. Threats to validity and reproducibility

Construct validity is limited by the attribution labels. The benchmark target reflects an annotator-defined decisive step, including judgments about recoverability and component responsibility. It is not an interventionally verified causal variable. Several steps may contribute to a failure, and agreement with a single label can penalize another defensible explanation. The one missing target and one actor/step disagreement demonstrate why source validation should precede modeling. The supplied step labels were preserved rather than retrospectively changed to favor a method.

Internal validity is strengthened by task grouping, training-only preprocessing, matched content, and independent metric checks. Nevertheless, group identification used available identifiers and exact normalized instructions; it cannot rule out semantically equivalent tasks with different wording. Field-specific coefficients increase model capacity, and the cap removes some information from long messages. Structural summaries can encode dataset-specific conventions. These limitations mean that neither a positive nor a negative result should be generalized to every structured representation, neural encoder, or diagnostic prompting method.

External validity is constrained by 219 failed benchmark runs, three systems, and the absence of successful-run controls. The corpus excludes the operational diversity of production tenancy, private data sources, concurrent users, and evolving business policies. The study cannot estimate detection specificity or the prevalence of particular failures in deployment. Repeated tasks were clustered, but the three architectures are too few to support population-level inference over agent frameworks. The cross-system experiment exposes this limitation rather than resolving it.

Statistical uncertainty also remains. Bootstrap intervals condition on fitted predictions, and secondary analyses are exploratory. Alternative partitions offer a limited assessment of training variability, not a replacement for an external test cohort. No prospective human study was performed, so analyst effort and repair success remain unmeasured. The replication materials provide source hashes, exclusions, per-trace predictions, and executable preprocessing to make these boundaries inspectable. A suitable next study would combine an independent operational cohort with controlled repairs and a blinded comparison of engineer diagnosis effort.

6. Conclusions

A matched analysis of 219 failed agent executions found no clear improvement in exact Top-1 localization from separating trace fields rather than flattening the same lexical evidence. Separate-field encoding offered a modest main-configuration Top-3 improvement, but that advantage was sensitive to modeling choices. Generalization to unseen agent systems was substantially weaker than task-grouped evaluation within the observed systems. Ranked shortlists retained annotated targets more often than reviewing only the latest records, supporting a limited triage role. The evidence favors evaluating structured traces together with their downstream diagnostic method and workload, while keeping shortlist coverage, causal diagnosis, and human debugging efficiency distinct.

Declarations

Acknowledgments and AI assistance. No generative AI tools or AI-assisted technologies were used at any stage of this work. Literature discovery, source verification, research-code development, data processing, statistical analysis, visualization, manuscript drafting, and document formatting were performed entirely by the author(s). Numerical results were generated by author-written executable scripts applied to the identified public dataset. Execution traces and decisive-step annotations originated from that dataset; no new human annotations or synthetic observations were introduced.

Data availability. The source archive is available from <https://huggingface.co/datasets/TraceElephant/TraceElephant> at revision 218e392a4c921d4379f827355b9e1d8147748b82 under CC BY 4.0. Supplementary File S1 documents additional methods, results, and published-method comparisons. Supplementary File S2 contains four analysis scripts, the source manifest, derived cohort, per-step scores, per-trace rankings, numerical summaries, and verification records. Its README provides the immutable source-download URL and commands for preprocessing, analysis, independent validation, and figure generation. The original archive is downloaded separately and verified by SHA-256 before preprocessing. The replication package is published with this article as Supplementary File S2.

Ethics statement. The analysis reused public benchmark artifacts and did not recruit participants, collect personal user interactions, or conduct interventions on deployed systems. No claim of a formal ethics-board determination is made. Original dataset attribution and license information are retained in the accompanying materials.

References

- [1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, et al., “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” in *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020. <https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html>

- [2] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, et al., “ReAct: Synergizing Reasoning and Acting in Language Models,” in Proceedings of the 11th International Conference on Learning Representations (ICLR), 2023. https://openreview.net/forum?id=WE_vluYUL-X
- [3] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, et al., “AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversations,” in Proceedings of the First Conference on Language Modeling (COLM), 2024. <https://openreview.net/forum?id=BAakY1hNKS>
- [4] W. E. Wong, R. Gao, Y. Li, R. Abreu, F. Wotawa, “A Survey on Software Fault Localization,” IEEE Transactions on Software Engineering, vol. 42, no. 8, pp. 707–740, 2016. <https://doi.org/10.1109/TSE.2016.2521368>
- [5] S. Zhang, M. Yin, J. Zhang, J. Liu, Z. Han, J. Zhang, et al., “Which Agent Causes Task Failures and When? On Automated Failure Attribution of LLM Multi-Agent Systems,” in Proceedings of the 42nd International Conference on Machine Learning, Proceedings of Machine Learning Research, vol. 267, pp. 76583–76599, 2025. <https://proceedings.mlr.press/v267/zhang25cq.html>
- [6] D. Deshpande, V. Gangal, H. Mehta, J. Krishnan, A. Kannappan, R. Qian, “TRAIL: Trace Reasoning and Agentic Issue Localization,” arXiv preprint 2505.08638v3, 2025. <https://doi.org/10.48550/arXiv.2505.08638>
- [7] M. Cemri, M. Z. Pan, S. Yang, L. A. Agrawal, B. Chopra, R. Tiwari, et al., “Why Do Multi-Agent LLM Systems Fail?,” arXiv preprint 2503.13657v3, 2025. <https://doi.org/10.48550/arXiv.2503.13657>
- [8] K. Zhu, Z. Liu, B. Li, M. Tian, Y. Yang, J. Zhang, et al., “Where LLM Agents Fail and How They can Learn From Failures,” arXiv preprint 2509.25370v1, 2025. <https://doi.org/10.48550/arXiv.2509.25370>
- [9] S. Barke, A. Goyal, A. Khare, A. Singh, S. Nath, C. Bansal, “AgentRx: Diagnosing AI Agent Failures from Execution Trajectories,” arXiv preprint 2602.02475v1, 2026. <https://doi.org/10.48550/arXiv.2602.02475>
- [10] M. Chen, J. Wang, F. Mu, Y. Wang, Z. Liu, H. Feng, et al., “Seeing the Whole Elephant: A Benchmark for Failure Attribution in LLM-based Multi-Agent Systems,” arXiv preprint 2604.22708v1, 2026. <https://doi.org/10.48550/arXiv.2604.22708>. Version cited: <https://arxiv.org/abs/2604.22708v1>
- [11] H. Li, Y. Yao, L. Zhu, R. Feng, H. Ye, J. Wang, et al., “CodeTracer: Towards Traceable Agent States,” arXiv preprint 2604.11641v3, 2026. <https://doi.org/10.48550/arXiv.2604.11641>
- [12] A. AlSayyad, K. Y. Huang, R. Pal, “AgentTrace: A Structured Logging Framework for Agent System Observability,” arXiv preprint 2602.10133v1, 2026. <https://doi.org/10.48550/arXiv.2602.10133>
- [13] Y. Zheng, Y. Hu, T. Yu, A. Quinn, “AgentSight: System-Level Observability for AI Agents Using eBPF,” arXiv preprint 2508.02736v2, 2025. <https://doi.org/10.48550/arXiv.2508.02736>
- [14] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, et al., “Lost in the Middle: How Language Models Use Long Contexts,” Transactions of the Association for Computational Linguistics, vol. 12, pp. 157–173, 2024. https://doi.org/10.1162/tac1_a_00638
- [15] A. Fournay, G. Bansal, H. Mozannar, C. Tan, E. Salinas, E. Zhu, et al., “Magentic-One: A Generalist Multi-Agent System for Solving Complex Tasks,” arXiv preprint 2411.04468v1, 2024. <https://doi.org/10.48550/arXiv.2411.04468>

- [16] J. Yang, C. E. Jimenez, A. Wettig, K. Lieret, S. Yao, K. Narasimhan, et al., “SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering,” in *Advances in Neural Information Processing Systems*, vol. 37, pp. 50528–50652, 2024. <https://doi.org/10.52202/079017-1601>
- [17] G. Mialon, C. Fourrier, C. Swift, T. Wolf, Y. LeCun, T. Scialom, “GAIA: A Benchmark for General AI Assistants,” in *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, pp. 9025–9049, 2024. https://proceedings.iclr.cc/paper_files/paper/2024/hash/25ae35b5b1738d80f1f03a8713e405ec-Abstract-Conference.html
- [18] O. Yoran, S. J. Amouyal, C. Malaviya, B. Bogin, O. Press, J. Berant, “AssistantBench: Can Web Agents Solve Realistic and Time-Consuming Tasks?,” in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8938–8968, 2024. <https://doi.org/10.18653/v1/2024.emnlp-main.505>
- [19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, et al., “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.
- [20] B. Efron, “Bootstrap Methods: Another Look at the Jackknife,” *The Annals of Statistics*, vol. 7, no. 1, pp. 1–26, 1979. <https://doi.org/10.1214/aos/1176344552>
- [21] G. Varoquaux, “Cross-validation failure: Small sample sizes lead to large error bars,” *NeuroImage*, vol. 180, pp. 68–77, 2018. <https://doi.org/10.1016/j.neuroimage.2017.06.061>