KyroBench: Evaluating Context Correctness Before AI Agents Act

Kishan
KyroDB
kishan@kyrodb.com

Abstract

AI agents increasingly depend on a runtime context layer that retrieves documents, incorporates tool observations, maintains memory, enforces scope, and decides what evidence reaches a model before action. Existing evaluations usually score final answers, raw retrieval quality, or systems throughput. Those measurements can miss a prior failure: the model may be given stale, polluted, wrong-scope, unsupported, or unauditable context before it reasons.

This paper introduces KyroBench, a benchmark for context correctness. KyroBench evaluates the context packet returned to an agent under mutable, multi-tenant, tool-connected workloads. It scores relevance, freshness, scope safety, pollution resistance, proof completeness, and efficiency while preserving hard certification findings outside the aggregate. The official-private frontier suite contains 2,048 cases, 39,424 documents, 66,560 workload steps, and 12,288 retrieve labels across support, code-agent, legal, SRE, CRM, synthetic healthcare, support-escalation, and public-policy domains. The suite stresses mutation windows, deleted content, memory scope, tool-result evidence, authority conflicts, active contradictions, prompt-injection pollution, and long-context budget pressure. Aggregate evidence from KyroDB, Graphiti, Qdrant, and Mem0 shows the central result: retrieval signal is not context correctness. Systems can return apparently relevant items while receiving zero proof-aware KyroBench score because they fail support, proof, pollution, freshness, or action-readiness gates.

1 Introduction

The modern agent stack contains a subsystem between the world and the model. This subsystem decides which documents are current, which memories are in scope, which tool observations are trustworthy, which revisions supersede older facts, and which evidence is safe to pass to a model with a limited context budget. When that subsystem is wrong, the model may still produce fluent output. The visible failure appears to be reasoning, but the root cause is often context.

KyroBench asks a narrow question: did the runtime return the right context before the model acted? The question is narrower than final task success and broader than vector recall. A correct context packet should contain the evidence required for the agent action, exclude forbidden or polluted material, respect tenant and session boundaries, serve the requested generation, and carry proof evidence when the system claims proof-aware correctness. This is the layer that determines whether the model receives a valid view of the world.

The distinction matters because common evaluations can reward systems that look useful while remaining unsafe for production agents. A vector index can retrieve semantically nearby text that is stale or unauthorized. A memory system can return another user's session state. A long-context strategy can include every plausible paragraph and bury decisive evidence under prompt-injection text. A final-answer benchmark can hide these defects when the model guesses correctly.

KyroBench is designed for systems that act as context runtimes: vector databases, memory stores, RAG middleware, agent frameworks, custom retrieval services, and databases with context-serving adapters. The benchmark does not assume these systems share an internal architecture. It only requires that they expose an adapter capable of accepting corpus and event updates, receiving retrieve requests, and returning bounded packets with item identifiers, revisions, content evidence, provenance, freshness metadata, and proof material when available.

This paper makes five contributions:

1. It defines context correctness as the benchmark object: the evidence packet returned before a model acts.
2. It introduces mutable, scoped, tool-connected workloads that expose stale serving, cross-scope leakage, proof gaps, and pollution.
3. It separates proof-aware scoring from retrieval and semantic diagnostics.
4. It specifies a private-custody publication model for frontier fixtures and leaderboard claims.
5. It reports aggregate official-private run evidence showing that retrieval signal is not context correctness.

2 Related Work

Information retrieval benchmarks such as BEIR and MTEB made broad retrieval comparison practical by stan-

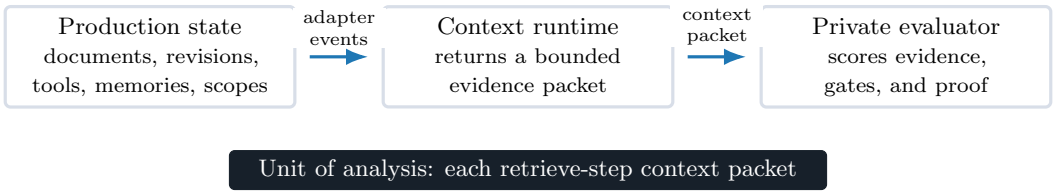

Figure 1: KyroBench isolates the pre-action context boundary. The benchmark does not score final model prose; it scores whether the evidence packet handed to the model is current, scoped, relevant, pollution-resistant, proof-bearing, and efficient.

dardizing datasets, metrics, baselines, and leaderboards [19, 15]. KILT and HotpotQA pushed retrieval evaluation toward provenance and supporting evidence [16, 22]. These benchmarks establish why evidence identity matters, but they do not directly test mutable production state, tenant boundaries, event-born tool observations, or proof completeness for agent context.

RAG evaluation frameworks such as RAGAS, ARES, RGB, and RAGBench study faithfulness, context precision, robustness, and automated evaluation [5, 17, 3, 6]. KyroBench borrows the premise that context should be judged directly, while keeping official scoring deterministic and proof-aware. Optional semantic judgments are diagnostics, not certification.

Agent and long-context benchmarks such as HELM, BIG-bench, SWE-bench, AgentBench, WebArena, OSWorld, GAIA, LongBench, LongMemEval, and LoCoMo broaden evaluation to models, tasks, environments, and memory behavior [9, 18, 8, 10, 23, 21, 13, 2, 20, 11]. KyroBench differs by isolating the pre-action context runtime. The benchmark asks whether the model was given the right evidence, before asking whether it produced the right answer.

Systems benchmarks such as YCSB, ANN-Benchmarks, and MLPerf demonstrate the value of versioned specifications, reproducible harnesses, official rules, and disclosure boundaries [4, 1, 12]. Reporting work such as Model Cards and Datasheets for Datasets informs the benchmark card, intended-use, and validity-threat structure [14, 7].

3 Benchmark Object

3.1 Context Packets

KyroBench evaluates retrieve-step context packets. The system under test receives documents, mutations, deletes, tool events, memory writes, cache warmup probes, and retrieve requests through a neutral adapter protocol. The returned packet contains item identifiers, revisions, optional content, provenance, scores, freshness metadata, and proof evidence when available.

The packet is evaluated before generation. This exposes failures that are otherwise conflated with model reasoning:

- stale evidence after a mutation;
- deleted-context resurrection;
- cross-tenant, cross-namespace, entitlement, user, session, or thread leakage;
- polluted tool-result or prompt-injection material;
- low-authority evidence overriding higher-authority sources;
- unsupported action context;
- proof-shaped metadata without reviewer-verifiable proof.

3.2 Formal View

Let a workload be an ordered event stream $E = (e_1, \ldots, e_n)$ containing corpus events, scope events, tool observations, memory writes, deletes, cache probes, and retrieve requests. At retrieve step t, the system observes the permitted state prefix and returns a bounded packet C_t. The private evaluator holds labels L_t that identify required evidence, forbidden evidence, freshness constraints, scope constraints, and proof requirements. A packet is correct only if it covers the required evidence, excludes forbidden evidence, respects the active scope and generation, and carries sufficient proof for asserted support.

3.3 Adapter and Packet Contract

KyroBench uses a neutral adapter boundary so that systems with different internal designs can be compared through the same observable behavior. The adapter receives corpus load events, mutation events, delete events, tool-call and tool-result events, memory writes, cache-warmup probes, and retrieve requests. For each scored retrieve, the system returns a bounded packet rather than a final answer.

The expected packet is intentionally evidence-oriented. It should identify returned items, expose document revisions or generations, provide content evidence when the benchmark requires content support, indicate freshness state, preserve provenance, and include proof evidence when the system claims proof-aware correctness. The benchmark therefore treats context service as a correctness boundary, not as an implementation detail hidden behind a model call.

Table 1: Context-packet contract. Fields are evaluated only through public-safe aggregate reports in this paper; private labels remain under evaluator custody.

Packet field	Evaluation role
Item identity	Matches returned evidence to required, forbidden, stale, deleted, polluted, or scope-trap material.
Revision/generation	Determines whether strict freshness was served after mutations, cache probes, and deletes.
Content evidence	Checks whether the packet actually supports the claim, not merely an id or score.
Scope metadata	Exposes tenant, namespace, entitlement, user, session, thread, and memory boundaries.
Provenance/proof	Distinguishes auditable evidence from proof-shaped metadata.
Budget/latency	Measures practical bounded-context behavior without allowing speed to offset correctness failures.

3.4 Non-Goals

KyroBench does not replace final-answer benchmarks, human preference evaluation, ANN recall benchmarks, database throughput benchmarks, red-team review, or product-specific safety review. Its purpose is to reveal whether the model was handed an action-ready packet. That packet may later feed a planner, a tool-using agent, a RAG answerer, or a human reviewer, but KyroBench scores the pre-action evidence boundary itself.

4 Workload Design

The official-private frontier suite is stress-scale and private-only. It is not a static QA dataset. It is a generated stateful workload whose cases contain current evidence, stale near-neighbors, active contradictions, restricted material, event-born tool observations, session or user memory, deleted archives, and pollution documents. Query text is a public execution input, but it must not contain answer tokens, document ids, revisions, split names, or hazard labels.

The suite spans support billing, code-agent repair, legal contracts, SRE runbooks, CRM memory, synthetic healthcare-style records, support escalation, and public policy change. Each domain pack is synthetic but production-shaped: cases are generated from deterministic domain-specific fact graphs rather than from reusable oracle strings. This preserves evaluator custody while forcing systems to handle evidence patterns that resemble real agent deployments.

The point is not to make retrieval harder for its own sake. The point is to reproduce conditions under which a context layer becomes part of the product's correctness boundary. A help-desk agent, code-repair agent, SRE assistant, or account-memory assistant cannot treat the newest matching paragraph, the most semantically similar memory, and the most convenient tool result as interchangeable evidence.

Hazard diagnostics further group labels by failure symptom: stale nearest-neighbor RAG, stale cache after mutation, concurrent update races, deleted-context resurrection, required session/tool memory freshness, tool-result pollution, tenant/namespace/entitlement/session/user leakage, prompt-injection pollution, lower-authority override, active same-scope contradiction, missing claim support, and long-context budget pollution.

5 Scoring

KyroBench reports a proof-aware headline score and diagnostics. For configurable suites, the weighted form is:

$$\mathrm{Score}_{\mathrm{weighted}} = 100 \left(w_q q + a \sum_{j \in O} w_j c_j \right). \tag{1}$$

Here q is context quality, a is the answerable-context rate, and O contains freshness, scope safety, clean-context behavior, proof completeness, and efficiency.

The official-private frontier suite used in this paper is stricter. Its headline score is an incident-success rate:

$$\mathrm{Score}_{\mathrm{frontier}} = \frac{100}{N} \sum_{t=1}^{N} \mathbf{1} \left[\bigwedge_{g \in \mathcal{G}_t} g \right]. \tag{2}$$

$\mathcal{G}_t$ includes answerable context, claim support, minimum authority, action readiness, content evidence, proof completeness, judged context, freshness, scope safety, pollution exclusion, deleted-context exclusion, unsafe-content exclusion, generation consistency, and strict-mode freshness when requested. A frontier incident receives credit only when every required gate passes.

The official-private frontier artifacts also record component weights: 35% context quality, 20% freshness, 18% scope safety, 15% pollution resistance, 10% proof completeness, and 2% efficiency. This distribution is verified in the private suite manifest and in every score artifact used for the aggregate results. The repository also contains a generic default manifest configuration; this paper reports the official-private frontier configuration because that is the suite used for the numbers in Section 6. The component weights explain the diagnostic profile; the frontier headline score reports all-gates incident success.

Retrieval Signal is a non-certifying diagnostic derived from raw retrieval behavior. Semantic Context Quality is also diagnostic and proof-neutral. Certification findings remain visible outside the aggregate. This separation is deliberate: a system should not earn a trustworthy context score by retrieving semantically nearby text that is stale, cross-scope, polluted, unsupported, or unauditable.

5.1 Certification Gates

KyroBench also reports certification status. Low latency, high recall, or a large context window cannot offset strict-

Table 2: Official-private frontier suite shape. These values are public-safe aggregate properties from the suite manifest and custody summary.

Property	Value	Property	Value	Property	Value
Suite id	official-private-frontier-v1	Scale	stress	Tracks	6
Cases	2,048	Documents	39,424	Workload steps	66,560
Retrieve labels	12,288	Query-bearing steps	20,480	Top-k	16
Tenants	8	Namespaces/tenant	4	Cases/namespace	64
Distractors/case	48	Pollution docs/case	12	Context budget	1,800 tokens
Public split	0%	Private holdout	85%	Canary	15%

Table 3: Domain packs and the production pressure each one contributes.

Domain pack	Production-shaped pressure
Support billing	Tickets, invoices, refund-policy revisions, entitlement overrides, support macros, tool observations, and escalation notes.
Code agent	Repository issue repair, branch-versioned source contracts, CI/tool observations, deleted CI summaries, developer memory, and scratchpad pollution.
Legal contract	Governing agreements, executed amendments, effective-date authority, superseded clauses, restricted addenda, and cross-matter traps.
SRE runbook	Incident response, authoritative runbooks, live alert/deployment state, superseded runbooks, deleted timelines, break-glass ACLs, and on-call memory.
CRM account memory	Account handoffs, opportunity state, stakeholder memory, stale relationship plans, deleted call notes, restricted overrides, and cross-account bleed.
Synthetic healthcare	Synthetic patient-style records, consent boundaries, temporal care-plan updates, corrected notes, active care-team memory, and privacy leaks without real patient data.
Support escalation trace	Current customer-impact policy, account-specific overrides, live case/tool state, callback memory, stale playbooks, active contradictions, and account traps.
Public policy change trace	Current public rules, controlling amendments, effective windows, policy-review memory, stale guidance, deleted FAQs, low-authority explainers, and policy traps.

Table 4: Workload axes. Each axis targets behavior that production agent runtimes cross routinely.

Axis	Purpose
Mutation	Revisions, supersession, upserts, and deletes after cache warmup.
Scope	Tenant, namespace, entitlement, user, session, thread, and memory visibility.
Tool events	Tool observations can create required evidence or adversarial pollution.
Memory	Session and user memories are event-born and must not bleed across owners.
Authority	Higher-authority evidence must defeat stale or lower-authority contradictions.
Budget	Packets must avoid stuffing plausible but harmful material.
Proof	Proof credit requires evaluator- or reviewer-verifiable evidence.

Table 5: Canonical scenario variants used as diagnostic slices. Each variant is represented in every official-private domain pack.

Variant	Stress pattern
full_adversarial	Combines stale, scope, memory, tool, pollution, authority, and budget pressure.
freshness_heavy	Old context is highly relevant but wrong for the current generation.
scope_heavy	Nearby evidence belongs to another tenant, namespace, entitlement, user, session, or thread.
session_memory_heavy	The decisive evidence is event-born memory scoped to the active owner.
tool_pollution_heavy	Tool observations can be required evidence or unsafe pollution.
authority_conflict_heavy	Lower-authority text is plausible but must lose to controlling evidence.
deleted_context_heavy	Archived or deleted material is tempting but forbidden.
sparse_business_lookup	The packet must find narrow business evidence without context stuffing.

mode stale serving, leakage, pollution, missing proof, or unsupported action evidence. Those failures remain visible as certification findings even when the aggregate score is reported. This makes the benchmark harder to game: a system cannot hide a boundary violation inside an average or compensate for missing proof by returning more text.

5.2 Diagnostic Slices

Every target reports domain, scenario-variant, track, family, and hazard diagnostics. These are not separate leaderboards; they explain why a score was obtained. A single headline score can hide different operational risks, so KyroBench exposes stale serve rate, leakage rate, pollution rate, deleted serve rate, proof completeness, p95 latency, context-budget overrun rate, precision, recall, and answerable-context rate for each diagnostic slice.

Table 6: Proof-aware scoring components for the official-private frontier suite.

Component	Weight	What it measures
Context quality	35%	Required support, answerable evidence, precision, recall, ranking, action readiness, and content support.
Freshness	20%	Correct generation under updates, stale cache windows, strict freshness, and delete/update ordering.
Scope safety	18%	Tenant, namespace, entitlement, user, session, thread, canary, and memory boundary isolation.
Pollution resistance	15%	Exclusion of tool-result pollution, prompt injection, stale contradictions, and budget-filling distractors.
Proof completeness	10%	Auditable support evidence rather than proof-shaped metadata.
Efficiency	2%	Latency and context-token behavior, reported without allowing speed to mask correctness failures.

6 Evaluation Evidence

This paper uses aggregate fields from local private run artifacts under the official-private frontier suite. The private fixture checksum inventory was verified before building this manuscript. The local score artifacts report `result_status = candidate_score`; consequently, this paper reports aggregate official-private run evidence and does not assert registry-derived rank.

6.1 Protocol

Each compared system is evaluated through the same adapter boundary and the same private workload. The evaluator executes corpus loads, cache probes, updates, deletes, tool observations, memory writes, and retrieve steps, then scores the returned packets against private labels. The aggregate reports used here contain three samples per system and 36,864 evaluated retrieve cases per system. The compared systems are KyroDB, Graphiti, Qdrant, and Mem0, selected because they represent different context-infrastructure families: database-native context serving, graph memory, vector retrieval, and agent memory.

The evaluation reports three distinct quantities that should not be conflated. KyroBench Score is the proof-aware aggregate. Retrieval Signal is a diagnostic for raw retrieval behavior. Semantic Context Quality is proof-neutral and diagnostic. Certification status and failure accounting explain whether the packet can be trusted, even when retrieval signal is high.

Tables 7–8 and Figure 2 show the central result. KyroDB has a Retrieval Signal of 79.13, Graphiti has 71.56, and Qdrant has 53.29. Yet all receive zero proof-aware KyroBench score because the official-private frontier score is an all-gates incident-success rate. Mem0 illustrates a different failure mode: memory systems need explicit lifecycle, scope, freshness, proof, and retrieval-surface alignment to operate as agent context runtimes.

This result is not merely a low-score observation. It is the benchmark's main falsification of a common industry assumption: retrieving plausible context is not enough. An agent context runtime must maintain a valid state boundary. It must not return stale evidence after an update, resurrect deleted material, cross an authorization boundary, include polluted instructions, omit proof, or give the model a packet that cannot support the requested action.

The gate and failure rates make the benchmark's purpose concrete. A retrieval benchmark may ask whether a relevant item appeared in the top k. KyroBench asks whether the packet is safe and useful for the agent action. In these artifacts, all four systems have zero answerable-context and claim-support pass rates under the frontier predicate. If a packet includes polluted material, misses proof, or cannot support the claim, it fails even when retrieval diagnostics look plausible.

6.2 Interpreting the Four Systems

KyroDB's aggregate shows the sharpest separation between retrieval and proof-aware correctness: high Retrieval Signal, full freshness, full scope safety, and full proof completeness, but zero pollution resistance and zero context-quality credit in this run. This indicates that the adapter returned plausible evidence but did not satisfy the complete action-ready packet contract under the pollution and support gates.

Graphiti also shows nontrivial retrieval signal but combines stale serving, missing proof, and full pollution failure. Qdrant shows the classic vector-retrieval limitation: relevant-neighbor retrieval is visible, but proof completeness and pollution resistance are absent under this adapter path. Mem0 has strong pollution resistance and scope safety but near-zero retrieval and missing proof, showing that memory systems need explicit alignment to the benchmark's lifecycle and evidence protocol before they can act as general context runtimes.

7 Discussion

KyroBench shifts the evaluation target from "can the system find nearby text?" to "can the system maintain an action-ready evidence boundary?" That shift has practical consequences. Retrieval APIs need lifecycle semantics, not only similarity scores. Memory APIs need ownership, authority, freshness, and deletion behavior that can be tested. Agent frameworks need context receipts that connect model input back to source state. Leaderboards need

Table 7: Aggregate official-private frontier results. Each system has three samples and 36,864 evaluated retrieve cases in the aggregate score report. The table is evidence for benchmark behavior, not a registry-rank claim.

System	KyroBench	Retrieval	Semantic	Fresh	Scope	Pollution	Proof	p95 ms	Tokens
KyroDB	0.00	79.13	0.00	1.00	1.00	0.00	1.00	520.2	869.3
Graphiti	0.00	71.56	0.00	0.77	1.00	0.00	0.00	167.9	1559.5
Qdrant	0.00	53.29	0.00	1.00	1.00	0.00	0.00	231.1	1473.0
Mem0	0.00	0.06	0.04	0.72	1.00	0.98	0.00	401.2	58.0

Table 8: Component profiles from the aggregate score reports. Component scores are on $[0, 1]$ and explain the zero proof-aware headline scores.

System	Context quality	Freshness	Scope safety	Pollution resistance	Proof completeness	Efficiency
KyroDB	0.000	1.000	1.000	0.000	1.000	0.440
Graphiti	0.000	0.769	1.000	0.000	0.000	0.407
Qdrant	0.000	1.000	1.000	0.000	0.000	0.333
Mem0	0.000	0.717	1.000	0.975	0.000	0.650

Table 9: Selected gate and diagnostic rates. Higher is better for answerable and claim-support pass rates; lower is better for stale, pollution, and proof-missing rates. Proof missing is 1− proof-complete rate.

System	Ans.	Claim	Stale	Poll.	No proof
KyroDB	0.0%	0.0%	0.0%	100.0%	0.0%
Graphiti	0.0%	0.0%	69.4%	100.0%	100.0%
Qdrant	0.0%	0.0%	0.0%	100.0%	100.0%
Mem0	0.0%	0.0%	0.0%	2.5%	100.0%

Table 10: Primary interpretation of aggregate failure surfaces.

System	Main reading from aggregate diagnostics
KyroDB	Retrieval signal is high, but pollution and support gates block proof-aware correctness.
Graphiti	Retrieval signal exists, but stale serving, pollution, and missing proof prevent certification.
Qdrant	Vector retrieval surfaces nearby evidence but lacks proof completeness and pollution resistance.
Mem0	Pollution resistance is comparatively strong, but retrieval, proof, and lifecycle alignment are insufficient.

diagnostic profiles because a single aggregate can hide distinct operational risks.

The result should not be read as a universal product claim about any one system. The artifacts disclose adapter boundaries, run modes, certification findings, and candidate-score status. The stronger conclusion is architectural: production agent infrastructure needs explicit correctness contracts for the context layer.

8 Governance and Reproducibility

KyroBench governance is part of the benchmark design. Official publication requires fixture custody, adapter compliance, repeated live runs, attempt accounting, review packets, bundle verification, and registry-derived leaderboard publication. Public packages must not include private evaluator fixtures, answer keys, private seeds, raw private scored context, canary contents, reviewer logs, or execution evidence.

The private frontier fixture lock binds the suite to hash commitments. The aggregate paper data are derived into public-safe JSON and CSV summaries. Private labels and raw scored packets remain under evaluator custody.

9 Validity Threats

Construct validity is the first threat. Context correctness is narrower than full agent success: a correct packet does not guarantee a correct final answer, and an incorrect packet may occasionally be rescued by model prior knowledge. KyroBench addresses this by stating the unit of analysis explicitly and by keeping final-answer quality out of the headline claim.

External validity is also limited. The official-private frontier suite uses synthetic but production-shaped domains. This supports custody, repeatability, and anti-gaming controls, but no fixed suite can cover every production integration. Continued domain-pack review, partner traces, and privacy-preserving contribution processes are necessary for broader coverage.

Internal validity depends on adapters and execution conditions. Backend timing, cache behavior, proof serialization, and delete semantics can change outcomes. The benchmark therefore treats adapter compliance, repeated live runs, attempt accounting, hashes, and custody receipts as part of the official publication process rather than as optional metadata.

Statistical validity requires reading diagnostics beside aggregates. One headline score can hide hazard-specific weaknesses, especially when certification failures are operationally different. KyroBench reports component scores, failure rates, domain slices, scenario slices, and hazard slices so that systems are not compared through a single opaque number.

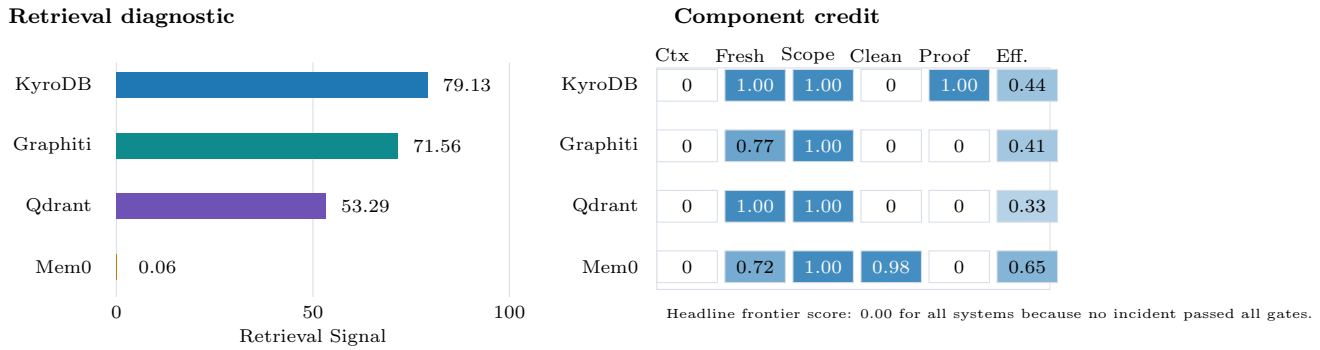

Figure 2: Retrieval Signal is not context correctness. The left panel shows raw retrieval diagnostics; the right panel shows component credit from the same official-private score artifacts. Nonzero component credit does not imply frontier incident success when context quality, pollution, proof, or support gates fail.

Governance validity matters because private fixtures can become an oracle if feedback is too detailed. The benchmark separates public artifacts from reviewer-private evaluator custody. Registry-derived publication artifacts should be used for rank language, while public papers should expose enough aggregate signal for progress without leaking labels.

10 Limitations

KyroBench is narrow by design. It measures context packets, not the full agent loop. It does not replace final-answer evaluation, human preference evaluation, ANN recall benchmarks, storage throughput benchmarks, or application-specific safety review.

The current evidence is aggregate and local. Since the available score files report `candidate_score`, official rank language should wait for a registry-derived publication artifact. The benchmark also uses synthetic but production-shaped domains. Synthetic domains improve custody and reproducibility, but external validity depends on continued domain-pack review, partner adoption, and trace-shaped contributions that pass privacy and anti-gaming checks.

A zero proof-aware score can be surprising when retrieval diagnostics are high. This is a feature of the benchmark contract, but it requires careful reader education. KyroBench should continue to present component and failure diagnostics beside the headline score.

11 Research Agenda

KyroBench is intended as a pressure surface for the next generation of agent infrastructure. Open problems include context transactionality after mutations and tool events, proof-carrying context that is useful to both models and reviewers, scoped memory without cross-owner leakage, pollution resistance before prompting, and benchmark governance that publishes enough diagnostic signal to drive progress without leaking private fixtures.

12 Conclusion

The next generation of agent infrastructure needs more than better nearest-neighbor retrieval. It needs context runtimes that can serve current, scoped, pollution-resistant, evidence-bearing packets before the model acts. KyroBench supplies a benchmark contract for that layer. The empirical evidence in this paper shows why the distinction matters: retrieval signal alone does not imply context correctness.

References

[1] Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems*, 2020.

[2] Yushi Bai, Xin Lv, Jiajie Zhang, et al. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

[3] Jiawei Chen et al. Benchmarking large language models in retrieval-augmented generation. *arXiv preprint*, 2024.

[4] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with ycsb. In *SoCC*, 2010.

[5] Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. Ragas: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217*, 2023.

[6] Robert Friel, Shubham Sanyal, Austin Schneider, et al. Ragbench: Explainable benchmark for retrieval-augmented generation systems. *arXiv preprint*, 2024.

[7] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, et al. Datasheets for datasets. *Communications of the ACM*, 2021.

[8] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

[9] Percy Liang, Rishi Bommasani, Tony Lee, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

[10] Xiao Liu, Hao Yu, Hanchen Zhang, et al. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023.

[11] Adyasha Maharana et al. Locomo: Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024.

[12] Peter Mattson, Vijay Janapa Reddi, Christine Cheng, et al. Mlperf: An industry standard benchmark suite for machine learning performance. *IEEE Micro*, 2020.

[13] Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, et al. Gaia: A benchmark for general ai assistants. *arXiv preprint arXiv:2311.12983*, 2023.

[14] Margaret Mitchell, Simone Wu, Andrew Zaldivar, et al. Model cards for model reporting. In *FAT**, 2019.

[15] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022.

[16] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. Kilt: a benchmark for knowledge intensive language tasks. In *NAACL*, 2021.

[17] Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. Ares: An automated evaluation framework for retrieval-augmented generation systems. *arXiv preprint arXiv:2311.09476*, 2023.

[18] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

[19] Nandan Thakur, Nils Reimers, Andreas Rucklé, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *NeurIPS Datasets and Benchmarks*, 2021.

[20] Yu Xia et al. Longmemeval: Benchmarking long-term memory in large language model agents. *arXiv preprint arXiv:2410.10813*, 2024.

[21] Tianbao Xie, Danyang Zhang, Jixuan Chen, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*, 2024.

[22] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, 2018.

[23] Shuyan Zhou, Frank F. Xu, Hao Zhu, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.