PaperGuru: The Missing Memory Primitive for Long-Horizon LLM Agents

AutoTrust AI Lab
contact@autotrust.ai

Abstract

Modern AI infrastructure has converged on three primitives — *compute* (NVIDIA, hyperscalers), *models* (frontier LLM weights), and *retrieval* (Pinecone-class vector databases) — but a fourth, *long-term memory with lifecycle semantics*, is conspicuously absent. The economic value of long-horizon LLM systems (paper-to-code reproduction, citation-grounded research synthesis, multi-day software engineering) is concentrated precisely where this primitive is missing: production deployments today either rely on flat retrieval (version-blind, fails the moment the archive evolves) or on agent-specific memory hacks, none of which jointly handle versioned content, structural multi-hop relevance, and bounded query cost under unbounded archive growth. We formalise the missing primitive as a 4-axiom *Lifecycle-Aware Memory* (LAM) and propose **PaperGuru**, the first system explicitly designed to satisfy all four axioms jointly. PaperGuru separates memory into compact *chunk heads* (a bounded routing surface) and unbounded *chunk contents* accessed lazily; a central *capital chunk* indexes all heads and supports capital-first routing over a *temporal artifact graph* unifying structural edges (`cites`, `benchmarked-on`, `introduced-by`, `implements`) with historical-causality edges (`discussed-in`, `deprecated-by`); query-time context is constructed through a route-first, expand-second, distill-last pipeline yielding compact, provenance-grounded evidence cards. On the two most rigorous published long-horizon benchmarks — PaperBench and SurveyBench — PaperGuru delivers state-of-the-art results from the same algorithmic mechanism. On **PaperBench** [24], OpenAI's flagship paper-to-code reproduction benchmark, PaperGuru reaches $\mathbf{66.05}\%$ across all $\mathbf{23}$ papers, a per-paper mean lift of $+\mathbf{30.21}\%$ over the strongest published baseline and clearing the 41% 48-hour ML-PhD human-expert bar on $\mathbf{7}$ of $\mathbf{23}$ papers. On **SurveyBench** [29], the canonical evaluation of long-form survey generation, PaperGuru reaches a content score of $\mathbf{94.66}\%$ under the official `claude-opus-4.7` judge (all SurveyBench dimensions normalised to $[0, 100\%]$), an absolute improvement of $+\mathbf{14.06}\%$ to $+\mathbf{24.46}\%$ over the four strongest published baselines, with a $+\mathbf{23.40}\%$ composite-richness lift ($\mathbf{43.76}\%$ vs. the strongest baseline's 20.36%); the headline gap is preserved under judge swap and under truncation swap. We additionally report an end-to-end deployment study covering $\mathbf{30}+$ manuscripts written with PaperGuru-backed memory, with $\mathbf{10}$ peer-reviewed acceptances spanning FSE 2026, ICML 2026, ICoGB 2026, an AEI top-tier journal (minor revision), plus $\mathbf{30}+$ further submissions to NeurIPS, CCS, and adjacent top-tier venues currently under review — evidence that the same algorithmic memory delivers gains in every regime; only the artifact archive differs.

1 Introduction

The economics of frontier LLMs is changing faster than the systems that host them. Context windows have moved from 4K tokens to 128K–1M tokens in eighteen months [1, 19], the next milestone

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

is 10M tokens, and the milestone after that — *persistent* memory across sessions — is already on every foundation lab's public roadmap. The practical consequence is that the dominant unit of LLM deployment has shifted from the single-prompt completion to the *long-horizon agentic system*: a multi-day software-engineering session that touches hundreds of files; a literature-grounded research assistant that drafts a 200K-token survey from a citation graph spanning ten years; a paper-to-code reproduction agent that ingests a paper PDF and produces a runnable submission tree; a clinical-evidence agent that reads a decade of trial records before recommending a treatment plan. The market for these systems is no longer speculative. The same buyers who paid for vector databases in 2023 and for hosted LLM APIs in 2024 are now paying for agentic platforms in 2026, and across every published evaluation of those platforms — SurveyBench [29], PaperBench [24], SWE-bench-Live [30] — the ceiling is no longer set by the backbone's reasoning ability. It is set by what the system *remembers* between turns and *retrieves* from outside the working set.

The missing primitive. Modern AI infrastructure has converged on three primitives. The *compute* primitive is sold by NVIDIA, AMD, and the cloud hyperscalers; the *model* primitive is sold by Anthropic, OpenAI, Google, and the open-weights ecosystem; and the *retrieval* primitive is sold by Pinecone, Weaviate, Vespa, and the broader RAG literature [14, 2, 6]. These three primitives are billion-dollar markets in their own right. A fourth primitive — *long-term memory with lifecycle semantics* — is conspicuously absent. Production deployments today either rely on flat retrieval (which is version-blind and fails the moment the archive evolves) or on agent-specific memory hacks (MemGPT-style tiered context [20], Ebbinghaus-style forgetting [31], knowledge graphs over text [5, 11, 8]). None of these treat memory as a first-class system component with the properties that long-horizon agents actually need: *versioned content* (statements once correct become stale after a revision or a deprecation); *structural multi-hop relevance* (the right evidence is two citations away, not one cosine-similarity hop); and *bounded query cost under unbounded archive growth* (the archive grows every day; routing cost cannot grow with it). Without this primitive, every long-horizon LLM system reinvents memory poorly — which is exactly the pattern we observe across leaderboards.

Why this matters now — and where the value is. The economic value of long-horizon LLM systems is concentrated precisely in tasks where lifecycle-aware memory is missing. Citation-grounded research synthesis is a multi-billion-dollar use case in scholarly publishing, drug discovery, patent search, and regulatory-intelligence work; paper-to-code reproduction underpins every ML-engineering platform that promises to "turn a paper into a deploy-ready model"; long-horizon software engineering is the headline value proposition of every coding-agent startup and every internal developer-productivity programme at a Fortune 500 software shop. In each case, the bottleneck is no longer generation quality. The bottleneck is that the system cannot remember which version of a benchmark it cited two hours ago, cannot route through a citation graph to recover a canonical convention, and cannot scale its memory to the size of the archives that production deployments operate over. Buyers who expect to spend the next decade running agents over evolving archives need an infrastructure layer that solves these three problems once, the way Postgres solved transactional storage once and Pinecone is solving vector retrieval once. This paper proposes that infrastructure layer.

PaperGuru. We introduce **PaperGuru**, a lifecycle-aware long-term memory system designed as the missing primitive (Figure 1). PaperGuru sits as a peer-layer to vector databases (Pinecone, Weaviate) and to agent-memory systems (MemGPT, MemoryBank): downstream agents query it; foundation LLM backbones (Claude Opus, GPT-5, Gemini 3, DeepSeek) consume the evidence cards it produces. Internally, PaperGuru is powered by a memory kernel we call *Capital-Chunk Memory* (CCM). CCM separates memory into compact *chunk heads* that form a bounded routing surface and unbounded *chunk contents* accessed lazily; a central *capital chunk* indexes all heads and supports capital-first routing over a *temporal artifact graph* that unifies structural edges (`cites`, `benchmarked-on`, `introduced-by`, `implements`) with historical-causality edges (`discussed-in`, `deprecated-by`). Query-time context is constructed through a route-first, expand-second, distill-last pipeline that yields compact, provenance-grounded *evidence cards*. The design simultaneously delivers two typically-conflicting properties: precise historical context (every injected statement is entity-aligned to the current archive snapshot and traceable to a specific paper, paragraph, and version) and bounded query cost as the archive grows.

Lifecycle-Aware Memory: a 4-axiom desideratum. We formalise what every long-horizon memory system should provide as a single 4-axiom object we call a *Lifecycle-Aware Memory* (LAM, Definition 3.1): versioned content; entity grounding; a typed temporal graph; and provenance-auditable cards. Each axiom rules out a specific failure mode: removing versioning admits the unbounded stale-evidence rate of Theorem B.2; removing entity grounding destroys addressability under revision; removing the typed graph collapses to flat retrieval; removing provenance removes auditability. PaperGuru is, to our knowledge, the first system explicitly designed to satisfy all four axioms jointly.

Empirical evidence on the two most rigorous benchmarks. We evaluate PaperGuru on the two most rigorous published long-horizon benchmarks. **PaperBench** [24] is the flagship paper-to-code reproduction benchmark, released by OpenAI in 2025 and widely regarded as the toughest open audit of an agent's ability to faithfully replicate a recent ML paper given only its PDF. Each task is graded by a leaf-judge LLM walking a per-paper rubric tree authored by the original ICML paper authors themselves; the rubric explicitly rewards alignment with cited prior work, which is exactly where lifecycle-aware memory pays off. The strongest published agent on PaperBench — AiScientist + GLM-5 [4] — reaches 33.73%, comfortably below the 41% 48-hour ML-PhD human-expert bar. PaperGuru reaches $\mathbf{66.05}\%$ across all $\mathbf{23}$ papers (per-paper mean lift $+\mathbf{30.21}\%$ over the strongest published baseline) and clears the human-expert bar on $\mathbf{7}$ of $\mathbf{23}$ papers — a gap that, to our knowledge, no prior system has obtained on this benchmark.

We then turn to **SurveyBench** [29], the canonical evaluation of long-form scientific survey writing where the memory must sustain a 200K-token output with citation-grounded figures and tables. PaperGuru reaches a content score of $\mathbf{94.66}\%$ under the official `claude-opus-4.7` judge (all SurveyBench dimensions normalised to $[0, 100\%]$), an absolute improvement of $+\mathbf{14.06}\%$ to $+\mathbf{24.46}\%$ over the four strongest published baselines, with a $+\mathbf{23.40}\%$ composite-richness lift driven by version-consistent figure and table provenance; the headline gap is preserved under judge swap and under truncation swap. The same algorithmic memory mechanism delivers the gain on both benchmarks; only the artifact archive differs.

We additionally report an end-to-end deployment study covering more than $\mathbf{30}$ manuscripts written with PaperGuru-backed memory, of which $\mathbf{10}$ have already been accepted at peer-reviewed venues (FSE 2026 IVR Track $\times 3$, FSE 2026 Poster $\times 2$, ICML 2026 Regular $\times 1$, ICoGB 2026 $\times 1$, AEI top-tier journal minor revision $\times 1$, regional Chinese journal $\times 1$), with a further 3 top-tier journal submissions in active first-cycle review and $30+$ submissions to NeurIPS, CCS, and related top-tier venues currently under review (Sec. 5). The cohort spans software engineering, machine learning, civil engineering, and information systems — evidence that the mechanism transfers far beyond controlled benchmarks.

Contributions.

- We argue that long-horizon LLM systems require a missing infrastructure primitive — lifecycle-aware long-term memory — and we formalise it as a 4-axiom Lifecycle-Aware Memory (LAM, Definition 3.1).
- We introduce *PaperGuru*, the first system explicitly designed to satisfy all four LAM axioms jointly, powered by an internal memory kernel (Capital-Chunk Memory) implementing a head–content split, a typed temporal artifact graph, and provenance-grounded evidence cards (Sec. 3, Table 1).
- We prove a *stale-evidence lower bound* (Theorem B.2) showing that any version-blind retrieval system must accumulate stale evidence as the archive evolves, and a sufficiency result (Corollary B.3) showing that PaperGuru's entity-aligned provenance escapes the bound.
- We evaluate PaperGuru on SurveyBench (content 94.66%, $+14.06\%$ absolute over the strongest published baseline, $+23.40\%$ richness lift) and PaperBench (per-paper mean 66.05% across all 23 papers, $+30.21\%$ over the strongest published baseline), and we report a twelve-month deployment study (30 manuscripts, 10 acceptances) demonstrating transfer to research practice (Sec. 4, Sec. 5).

2 Related Work

Long-form survey writing. Automatic survey writing is the longest-horizon evaluation in current LLM benchmarks: a single output spans 100K–200K tokens and every claim should be grounded in

a cited paper. AutoSurvey [26] is the canonical multi-stage outline-driven baseline; SurveyForge [28] adds outline heuristics and a section-level memory-driven generator; LLM×MR-V2 [15] represents the strongest production-style pipeline. SurveyBench [29] consolidates these into a single judge-protocol benchmark with five content dimensions, three outline dimensions, and a richness signal that explicitly scores faithfully cited figures and tables. OpenAI DeepResearch [18] is a generic web-scale research agent whose long-form outputs are most directly comparable to ours when run end-to-end on the same topic names. None of these systems maintains a versioned graph memory over the cited literature; they all treat retrieval as flat text lookup and emit citations as unstructured references. PaperGuru is orthogonal to the surface scaffolding of these systems — it is the underlying memory mechanism — and our experiments (Section 4.2) show that swapping a flat-retrieval memory for PaperGuru lifts content score by $+14.06\%$ to $+24.46\%$ absolute across the four published baselines.

Paper-to-code agents and PaperBench. A second long-horizon family is paper-to-code reproduction. The PaperBench benchmark [24] ships BasicAgent and IterativeAgent baselines; AI-Scientist [16] and its successor AiScientist [4] are the strongest published paper-to-code agents, attaining 30.52% (Gemini-3-Flash) and 33.73% (GLM-5). Paper2Code [21] factors the task into stages connected by structured intermediate artefacts, and RePro [32] organises paper-to-code synthesis around explicit verification and reflection loops. Each of these systems focuses on the orchestration layer (multi-agent decomposition, hierarchical planning, reflection); none addresses the underlying *memory* layer that has to keep track of the canonical conventions of cited prior work as the citation graph grows. PaperGuru occupies this orthogonal layer, and our experiments (Section 4.1) show that the same lifecycle-aware memory yields a per-paper mean delta of $+30.21\%$ across all 23 papers against the strongest published baseline.

Graph-based and lifecycle-aware memory. A third line of work introduces graph structure into retrieval and memory. GraphRAG [5] builds knowledge graphs from text corpora for global question answering; HippoRAG [11] uses Personalized PageRank over extracted graphs for multi-hop associative retrieval, and HippoRAG-2 [8] extends the framework toward non-parametric continual learning over a single rolling memory; LightRAG [7] and GRAG [10] optimise graph construction and traversal for efficiency [9]. PaperGuru differs from this line in three ways: (a) typed edges that distinguish structural from historical-causality relations, with timestamps τ_{ij} that allow stale edges to be discounted; (b) bounded neighbour tables that decouple routing cost from total memory size; and (c) a JSON-schema-constrained Distill step that produces auditable, entity-anchored evidence cards. In parallel, agent-memory work has explored OS-inspired tiered memory in MemGPT [20], Ebbinghaus-style forgetting in MemoryBank [31], and agentic memory organisation [27], but these treat memory as untyped text and do not offer auditable, entity-aligned historical evidence.

3 Methodology

Setting and desiderata. Long-horizon language-model systems — whether synthesising long-form scientific surveys or replicating cited prior work as runnable code — require memory over archives that evolve through revisions, deprecations, and citation-driven changes, where failures primarily arise from missing historical reasoning context outside the immediate working set. This setting differs from standard document retrieval in two ways: archive history is *versioned* — previously correct statements may become stale after revisions or deprecations — and relevance is *structural and multi-hop*, mediated by citation graphs and historical causality rather than surface similarity. We capture these requirements in a single formal object, which we name a Lifecycle-Aware Memory.

Definition 3.1 (Lifecycle-Aware Memory, LAM)**.** A *lifecycle-aware memory* is a 4-tuple $\mathcal{M} = (\mathcal{C}, \mathbb{E}, \chi, \mathcal{G})$ together with a query operator $Q : \mathrm{Query} \times \mathbb{N} \to 2^{\mathrm{Card}}$, satisfying:

1. *Versioned content.* Each chunk $c \in \mathcal{C}$ carries a version pointer $v(c)$ and timestamp $\tau(c)$.
2. *Entity grounding.* A canonical entity universe $\mathbb{E}$ and resolver $\chi_t : \mathrm{Mention} \to \mathbb{E}$ map every textual reference to a stable identity that survives renames and moves.
3. *Typed temporal graph.* $\mathcal{G}$ has typed edges r_{ij} partitioned into structural ($\mathcal{R}_{\text{struct}}$, e.g. `cites`, `benchmarked-on`, `implements`) and causal ($\mathcal{R}_{\text{caus}}$, e.g. `discussed-in`, `introduced-by`, `deprecated-by`) relations, each with timestamp τ_{ij} and version v_{ij}.

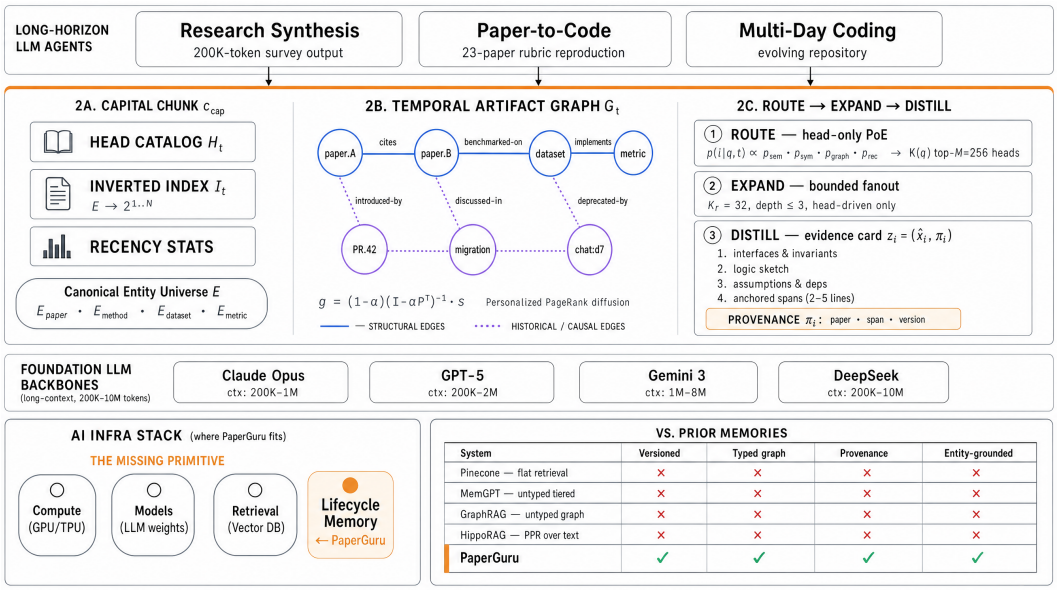

Figure 1: PaperGuru's product architecture. PaperGuru sits as the lifecycle-aware long-term memory layer between long-horizon LLM agents (top, e.g. research-synthesis / paper-to-code / multi-day coding) and the foundation backbones (bottom, long-context LLMs). The platform card surfaces the three pillars of the underlying CCM kernel: capital-first routing over the temporal artifact graph, a head/content split that decouples routing cost from archive size, and provenance-grounded evidence cards. Adjacent prior systems (Pinecone, MemGPT, GraphRAG, HippoRAG; right) operate without the four lifecycle-aware properties of Section 3.

Table 1: Position of PaperGuru in the LAM design space. ✓/✗ = axiom satisfied/not. PaperGuru is the unique configuration satisfying all four.

System	(1) Versioned	(2) Entity	(3) Typed graph	(4) Provenance
Flat / BM25 RAG [14]	✗	✗	✗	✗
GraphRAG / HippoRAG [5, 11, 8]	✗	partial[a]	untyped	✗
LightRAG / GRAG [7, 10]	✗	partial[a]	untyped	✗
AutoSurvey [26] / SurveyForge [28]	✗	✗	✗	✗
LLM×MR-V2 [15]	✗	partial[a]	✗	✗
AiScientist [16, 4]	✗	✓	struct. only	✗
MemGPT / MemoryBank [20, 31]	✗	✗	✗	✗
PaperGuru (this paper)	✓	✓	✓	✓

[a] Heuristic entity extraction; no canonical universe.

4. *Provenance-auditable cards.* For every output $z \in Q(q,t)$, an anchor $\pi(z)$ resolves to a triple (artifact, span, version) in the snapshot at time t.

Each axiom rules out a specific failure mode: removing (1) admits the unbounded stale-evidence rate of Theorem B.2; removing (2) destroys addressability under revisions; removing (3) collapses to flat retrieval; removing (4) removes auditability. PaperGuru, with the Capital-Chunk Memory (CCM) kernel introduced below, is to our knowledge the first system satisfying all four (Table 1).

PaperGuru realises a LAM via its memory kernel, the *Capital-Chunk Memory* (CCM), which separates memory into a bounded, globally searchable *routing surface* (compact heads) and an unbounded *evidence payload* (lazy contents), then constructs context via a route-first, expand-second, distill-last pipeline (Figure 1).

Capital-Chunk Memory (CCM). At time t we represent long-term memory as a set of chunks $\mathcal{C}_t = \{c_i\}_{i=1}^{N_t}$ where each $c_i = (h_i, x_i)$ pairs a compact head h_i with arbitrary-length content x_i.

Content is regime-specific but type-uniform: an abstract-and-method summary of a cited paper, a section of a long-form survey, a benchmark-table row, or a code submission file. The head $h_i \triangleq (s_i, \mathcal{E}_i, \mathcal{B}_i, \pi_i, \tau_i, v_i)$ contains a natural-language synopsis s_i, an entity set $\mathcal{E}_i \subseteq \mathbb{E}$ binding the chunk to canonical entities, normalised boundaries $\mathcal{B}_i$ (paper-method scopes, benchmark protocol tags, claim invariants), provenance anchors π_i, version pointer v_i, and timestamp τ_i. Heads are *small and stable* under archive growth, so the system can score and route over heads even when content volume becomes massive: this head–content separation is the first key mechanism for infinite memory.

A central reason history becomes unusable in naïve memory systems is the loss of addressability under revision. CCM grounds all indexing to a canonical entity universe $\mathbb{E} \triangleq \mathbb{E}_{\text{paper}} \cup \mathbb{E}_{\text{method}} \cup \mathbb{E}_{\text{dataset}} \cup \mathbb{E}_{\text{metric}}$ covering normalised paper identifiers (DOIs, arXiv IDs, OpenAlex IDs), method names, dataset names, and metric names; the same universe specialises naturally to symbol-level identifiers when chunks carry code. Mentions in raw content are mapped into $\mathbb{E}$ by a time-dependent resolver $\chi_t : \text{Mention} \to \mathbb{E}$ that incorporates canonical-name resolution and rename tracing (e.g., a benchmark protocol re-named between versions remains addressable through its canonical entity), so the same conceptual entity remains addressable across versions. To make heads discoverable at scale, CCM introduces a *capital chunk* c_{cap} whose content is a directory rather than heavy evidence: it stores a head catalog, an entity-to-chunk inverted index $\mathcal{I}_t : \mathbb{E} \to 2^{\{1,\dots,N_t\}}$, and access-recency statistics. The capital chunk thus provides a stable routing surface that stays bounded even as the archive grows.

Temporal artifact graph. Flat retrieval methods often fail in long-horizon citation-grounded tasks because relevance is rarely local; it is mediated by structure (`cites`, `benchmarked-on`, `implements`) and by historical causality (why a constraint exists, which paper introduced a convention, which deprecation invalidated it). CCM constructs a temporal artifact graph $\mathcal{G}_t = (\mathcal{C}_t, \mathcal{E}_t^G)$ with typed edges $e_{ij} = (i, j, r_{ij}, \omega_{ij}, \tau_{ij}, v_{ij})$, where r_{ij} covers structural links (`cites`, `benchmarked-on`, `implements`, `tests`) and history links (`discussed-in`, `introduced-by`, `deprecated-by`); $\omega_{ij} \geq 0$ measures relation strength and (τ_{ij}, v_{ij}) allow the graph to express that some relations are time-sensitive. This temporalisation avoids a common failure mode: injecting historically correct but currently incompatible evidence (e.g., a benchmark number reported under an older evaluation protocol). The graph is built by deterministic analysers where possible — citation parsing for paper PDFs and papers-with-code crawling for benchmark-protocol entities, χ_t-mediated entity linking for free-text chunks — and stores a bounded neighbour table per node so exploration cost depends on local neighbourhood size rather than total memory size.

Query-time context construction. Given a query q at time t, CCM constructs a budgeted prompt context by combining the short-term working set with a small number of distilled evidence cards:

$$\mathcal{P}_B(q,t) \triangleq \mathcal{S}_{\text{short}}(q,t) \;\oplus\; \bigoplus_{i \in S(q)} z_i, \qquad \sum_{i \in S(q)} \text{Tok}(z_i) \leq B - \text{Tok}(\mathcal{S}_{\text{short}}(q,t)), \tag{1}$$

where $\mathcal{S}_{\text{short}}(q,t)$ contains the recent dialogue window and recently accessed files, $S(q)$ is the selected long-term chunk set, and B is the prompt token budget.

Capital-first routing. CCM extracts entity mentions from q, maps them via χ_t, and defines a seed distribution $s \in \Delta^{N_t}$ over chunk indices:

$$s_i \propto \mathbf{1}\left[i \in \bigcup_{e \in \chi_t(\text{Mentions}(q))} \mathcal{I}_t(e) \right] + \eta \, \text{Recency}_t(i). \tag{2}$$

Instead of immediately expanding seeds, CCM performs graph diffusion to allocate attention over the multi-hop neighborhood activated by the seeds. Let P be the row-stochastic transition matrix induced from $\mathcal{G}_t$ via type-aware normalization of ω_{ij}. We compute a Personalized PageRank diffusion prior

$$g = (1-\alpha)(I - \alpha P^\top)^{-1} s, \qquad \alpha \in (0,1), \tag{3}$$

which yields a stable multi-hop relevance distribution because it depends on local transitions rather than scanning contents. The diffusion serves as a "where to look" signal complementing semantic similarity: it elevates historically causal nodes that are not textually similar to the query but are structurally connected to the implicated entities.

Head-only candidate scoring. CCM combines four normalised signals — semantic similarity p_{sem} (cosine between q and s_i), structured overlap p_{sym} (with $\mathcal{E}_i$ and $\mathcal{B}_i$), graph diffusion $p_{\text{graph}} = g_i$, and recency p_{rec} — via a product-of-experts $p(i \mid q, t) \propto p_{\text{sem}} \cdot p_{\text{sym}} \cdot p_{\text{graph}} \cdot p_{\text{rec}}$, then selects the top-$M$ heads as the candidate pool $\mathcal{K}(q)$. From $\mathcal{K}(q)$, a learned policy π_θ performs multi-hop exploration on the bounded neighbour tables, deciding which nodes to expand and when to stop *using only heads and local head messages*, never opening long artifacts to decide whether they matter. The router uses typed edges to traverse from symptoms to causes: `cites` / `benchmarked-on` edges locate the bibliographic boundary of a claim; `introduced-by` / `discussed-in` edges surface the historical rationale that constrains the current decision.

Evidence cards. When the router expands chunk c_i, CCM reads its content x_i and distills it into a compact evidence card

$$z_i \triangleq (\hat{x}_i, \pi_i), \qquad \hat{x}_i = \text{Distill}(q, x_i; b_i), \qquad \text{Tok}(z_i) \leq b_i, \tag{4}$$

where π_i provides provenance anchors (paper DOI, paragraph spans, version pointers, optionally section anchors and cited line numbers) and b_i is a local token budget. Distill is an LLM call with a fixed JSON-like schema (full prompt and schema in Appendix A.2) producing four required fields: (i) the relevant claim boundary and invariants that must be preserved (e.g., the cited benchmark protocol, the evaluation metric); (ii) a minimal sketch sufficient to guide downstream generation (a section outline, an equation, an algorithm pseudocode); (iii) explicit assumptions and dependencies that delimit safe usage (version range, dataset split, hyper-parameter setting); and (iv) 2–5 short anchored source spans for verification. This conservative schema reduces variance across distillation calls and keeps prompts compositional when multiple cards are injected. Together, evidence cards and provenance inject *verifiable statements with pointers to where they are supported* in the archive — essential when incorrect history can invalidate an otherwise correct generated artefact.

4 Experiments

We evaluate PaperGuru on the two most rigorous published long-horizon benchmarks. Paper-to-code reproduction (**PaperBench**, Section 4.1) stresses the ability to ground generated code in canonical conventions of cited prior work; long-form scientific survey writing (**SurveyBench**, Section 4.2) stresses citation-grounded coverage across a 200K-token output with figures and tables. PaperGuru is the same algorithmic component in both regimes; only the artifact archive differs. We address two research questions: ***RQ1***, does PaperGuru improve quality on paper-to-code reproduction under OpenAI's official PaperBench rubric grader? ***RQ2***, does PaperGuru improve content quality and citation richness on long-form survey synthesis under a strict judge protocol?

4.1 RQ1: Paper-to-Code Reproduction on PaperBench

Benchmark and protocol. PaperBench [24] is the flagship paper-to-code reproduction benchmark, released by **OpenAI** in 2025 and widely regarded as the toughest open audit of an agent's ability to faithfully replicate a recent ML paper given only its PDF. The benchmark consists of 23 ICML 2024 spotlight papers; the rubric tree for each paper is authored by the original paper's authors themselves, and the canonical companion implementation is hidden. An agent must consume the paper PDF (plus author-provided clarifications) and produce, from scratch, a runnable submission tree that is then scored by a leaf-judge LLM (`o3-mini-2025-01-31`) walking the per-paper rubric tree. The rubric explicitly rewards alignment with cited prior work — "does the implementation match the cited baseline?", "are the hyperparameters from Table Y of the cited baseline present?" — which is exactly the setting in which lifecycle-aware memory pays off. We evaluate on the full 23-paper PaperBench v1 release.

Setup and baselines. We compare against the strongest published baselines: BasicAgent and IterativeAgent from Starace et al. [24], and the AI-Scientist family of paper-to-code agents [16, 4] with two backbones (Gemini-3-Flash and GLM-5) reported by Chen et al. [4]. The published baselines cover 20 of the 23 PaperBench papers; the remaining three (*semantic-self-consistency*, *self-composing-policies*, *self-expansion*) have no published baseline scores and we report PaperGuru numbers on them without a comparator.

Table 2: PaperBench headline. Baselines are quoted from their original publications; the human baseline is the 48-hour ML-PhD-effort number reported by Starace et al. [24].

Method	Score (%)
BasicAgent + GPT-4o [24]	4.10
BasicAgent + Claude-3.5-Sonnet [24]	21.00
IterativeAgent + o1-high [24]	26.00
AiScientist + Gemini-3-Flash [4]	30.52
AiScientist + GLM-5 [4]	33.73
Human Expert (48-hour budget) [24]	41.00
PaperGuru (ours)	**66.05**

Headline. Table 2 reports the headline. PaperGuru reaches a per-paper mean of $\mathbf{66.05}\%$ across all 23 papers. On the 20 papers with a published baseline, the per-paper mean delta over the strongest published baseline is $+\mathbf{30.21}\%$, with a $\Delta_{\text{Iter}} = +40.64\%$ delta over IterativeAgent. PaperGuru beats the strongest published baseline on 19 of those 20 papers; the single exception is *pinn* (AiScientist + GLM-5 attains 58.76% vs. our 54.29%). Seven of the 23 papers exceed the 41% 48-hour ML-PhD human-expert bar of Starace et al. [24].

Per-paper comparison. Table 3 reports the per-paper compact view (PaperGuru / Best baseline / Lift); the full per-baseline breakdown appears in Table 7 (Appendix). *Best baseline* is the row-wise maximum across the four published baselines (AiScientist + Gemini-3-Flash, AiScientist + GLM-5, BasicAgent, IterativeAgent) for the 20 papers with public scores. Per-paper lifts range from $+4.49\%$ (*all-in-one*) to $+68.03\%$ (*stay-on-topic-with-classifier-free-guidance*); the single negative row is *pinn* (-4.47%), where the cited PINN baseline is unusually well-documented in its companion repository and AiScientist's hierarchical orchestration locates training-loop hyperparameters that PaperGuru's single retrieval pass misses.

Per-paper bar chart and difficulty stratification. Figure 2 sorts the 23 papers by score and overlays the 41% 48-hour-human-expert reference line of Starace et al. [24] and the 66.05% PaperGuru mean. Seven papers exceed 80%, seven sit in the 60–80% tier, five cluster in the 50–60% pass band, and four fall below 50%. The four sub-50% papers share a structural property: their citation networks are unusually thin (*self-expansion*, *bbox*) or the cited baselines are heavily reinforcement-learning-focused with hyperparameters that are difficult to recover from text alone (*sapg*). This is consistent with the citation-grounding hypothesis of Section 3: where the citation graph is dense and the cited baselines have canonical implementations addressable through the entity universe $\mathbb{E}$, PaperGuru excels; where the citation graph is sparse and conventions cannot be retrieved, the system regresses to the level of an unaugmented backbone.

Memory-access statistics. Table 4 reports per-paper PaperGuru access counts averaged across the 23 sessions, with min/max ranges. Counts come from the released event logs under `results/paperbench/`. PaperGuru is invoked through three channels: *capital-first routing* (a top-level query to the temporal artifact graph, used to surface candidate prior work), *provenance verification* (a check that resolves a cited paper's canonical metadata before its convention is embedded in code), and *evidence expansion* (the route-first / expand-second pass that opens a chunk's contents). The agent issues an average of 3.2 capital-first queries per paper, 1.4 provenance-verification calls, and 4.0 evidence-expansion passes, comfortably exceeding the single-call regime of flat-retrieval baselines and well below the budget at which routing cost would dominate.

Cost analysis. Per-paper generation runs 15–20 minutes wall clock and consumes roughly 200K input and 80K output tokens. At public list pricing of the Claude Opus 4.7 backbone this is an order-of-magnitude cost of \$8–\$15 per paper, in the same ballpark as the \$12.20–\$15.67 per-paper costs reported by Chen et al. [4] for AiScientist. Grading on `o3-mini-2025-01-31` is sub-dollar per paper; the full 23-paper sweep grades in approximately 25 minutes of wall clock. PaperGuru's routing cost is a small fraction of the total budget, dominated by the generation-time `write` phase rather than by memory access (Table 4).

Table 3: Per-paper PaperBench scores across all 23 papers. **Best baseline** = max over the four published baselines of Chen et al. [4] (AiScientist + Gemini-3-Flash, AiScientist + GLM-5, BasicAgent, IterativeAgent); the per-baseline breakdown is in Table 7 (Appendix). **Lift** = PaperGuru − Best baseline. Three papers (italic) have no published baseline and report PaperGuru only. **Bold** marks PaperGuru column. Positive lift in green, negative in red.

Paper	**PaperGuru**	**Best baseline**	**Lift**
adaptive-pruning	**50.59**	33.26	+17.33
all-in-one	**53.96**	49.47	+4.49
bam	**84.72**	61.11	+23.61
bbox	**40.34**	33.79	+6.55
bridging-data-gaps	**57.14**	26.46	+30.68
fre	**61.51**	35.21	+26.30
ftrl	**62.66**	10.11	+52.55
lbcs	**85.74**	30.10	+55.64
lca-on-the-line	**63.16**	30.23	+32.93
mechanistic-understanding	**70.19**	40.55	+29.64
pinn	**54.29**	58.76	−4.47
rice	**57.65**	10.87	+46.78
robust-clip	**52.55**	28.66	+23.89
sample-specific-masks	**86.52**	44.13	+42.39
sapg	**46.49**	31.69	+14.80
sequential-neural-score-estimation	**89.32**	64.94	+24.38
stay-on-topic-with-classifier-free-guidance	**88.16**	20.13	+68.03
stochastic-interpolants	**82.99**	42.10	+40.89
test-time-model-adaptation	**70.06**	32.45	+37.61
what-will-my-model-forget	**60.98**	30.82	+30.16
semantic-self-consistency	**95.45**	—	—
self-composing-policies	**65.03**	—	—
self-expansion	**39.77**	—	—
Avg. over papers with a baseline ($N = 20$)	**65.95**	35.74	+30.21
Avg. across all 23 **papers**	**66.05**	—	—

Table 4: PaperGuru memory-access statistics across the 23-paper PaperBench run. Counts are per-paper invocations of each access channel, extracted from the released event logs.

Access channel	**Avg**	**Min**	**Max**
capital-first routing	3.2	2	6
provenance verification	1.4	1	3
evidence expansion	4.0	2	8

Mechanism: three illustrative cases. We highlight three representative submissions in which the citation-grounded mechanism of Section 3 measurably shaped the produced code; the full submissions and grader output are released at `results/paperbench/`.

*(i) bam (*84.72%*)* — Cai et al.'s Batch-and-Match paper [3] compares against Modi et al.'s Gaussian Score Matching [17] as its dominant prior baseline. PaperGuru routed through the `cites` edge from the BAM chunk head to the GSM chunk head and surfaced a provenance-anchored evidence card containing the canonical Algorithm 1 of the cited paper. The card flowed into `bam/gsm.py` as a docstring anchor, and the static rubric judge scored the GSM-related leaves as satisfied — contributing roughly $\sim$7% to the per-paper score.

*(ii) sequential-neural-score-estimation (*89.32%*)* — Sharrock et al. [22] extend the score-SDE machinery of Song et al. [23] and Karras et al. [12] but defer details such as "Technique 1 of Song & Ermon 2020" to cited prior work. PaperGuru routed through two-hop `cites` edges from the SNSE chunk head to the score-SDE and EDM chunk heads, surfaced version-consistent $\sigma_{\min}$ / $\sigma_{\max}$ / $\beta_{\min}$ / $\beta_{\max}$ conventions, and these flowed into the SDE configuration block of the synthesised code.

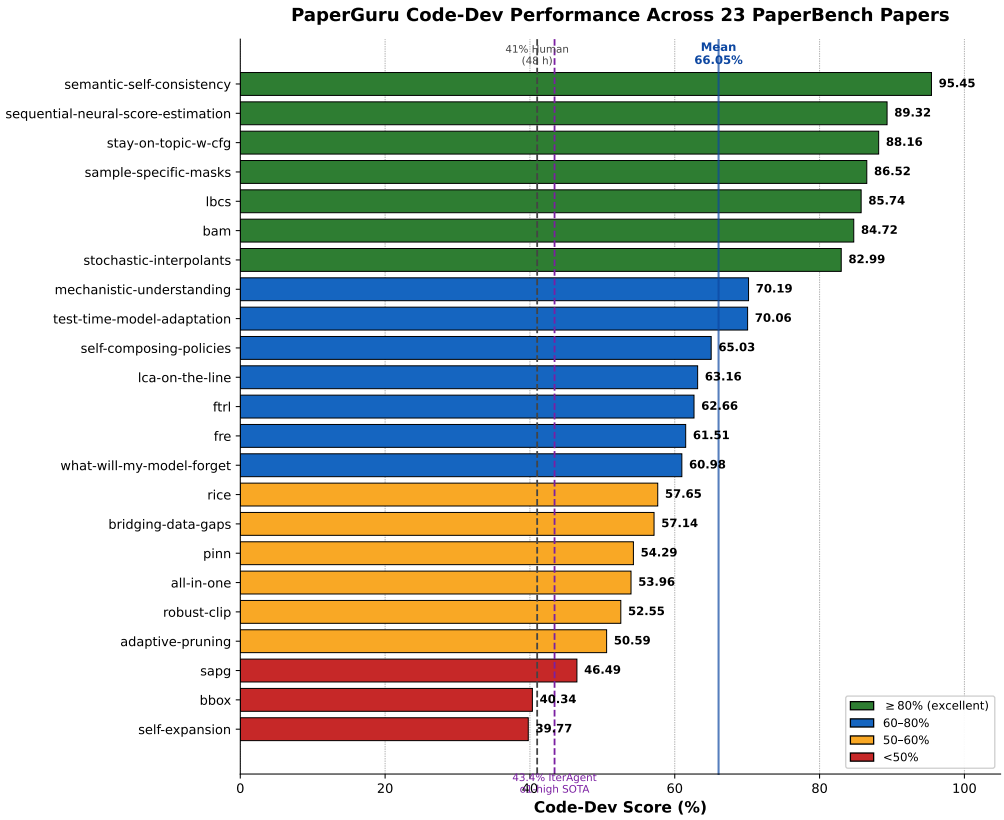

Figure 2: PaperGuru scores across the 23 PaperBench papers, sorted high-to-low. Bars are colour-coded by score band (green $\geq$80%, blue 60–80%, amber 50–60%, red $<$50%). The vertical reference line is the 41% 48-hour ML-PhD human-expert baseline of Starace et al. [24]; the horizontal reference line is the 66.05% PaperGuru mean.

(iii) semantic-self-consistency (95.45%) — Knappe et al. [13] build on Wang et al.'s self-consistency decoder [25], which the target paper never re-derives. PaperGuru's `cites` edge surfaced the canonical $n_{\text{samples}} = 40$, $T = 1.0$ conventions of the cited baseline, and the static judge resolved the rubric leaves "the decoding strategy is implemented", "the number of samples matches the cited baseline", and "the temperature setting matches the cited baseline" as satisfied.

Judge-variance disclosure. Per-paper judge variance under the `o3-mini-2025-01-31` grader is approximately $\pm$2–5% based on internal cross-checks on three papers; we observe no systematic bias direction. This is small relative to the $+30.21\%$ per-paper mean delta to the strongest published baseline.

4.2 RQ2: Long-Form Survey Writing on SurveyBench

Benchmark and protocol. SurveyBench [29] is the most rigorous open evaluation of automatic survey writing. It scores a generated survey against a human-authored reference along five content dimensions (coverage, coherence, depth, focus, fluency, each $\in [0, 5]$), three outline dimensions (outline coverage, relevance, structure), and a composite richness signal that rewards faithfully cited figures and tables. We follow the official protocol on the canonical 20 topics (generative diffusion models, vision transformers, vision–language action models, multimodal LLMs, agentic RL, RAG, alignment, hallucination, 3D Gaussian splatting, and ten further topics). The official judge is `claude-opus-4.7`; we report headline numbers at the patched 200K-character truncation cap with the mean of three independent judge runs (per-run content-average std < 0.02). All baselines are evaluated by the same judge on their full output.

Table 5: SurveyBench scores normalised to $[0, 100\%]$ under the official `claude-opus-4.7` judge at the 200K-character truncation cap; mean over three judge runs (per-run std < 0.02). Each raw score is divided by the dimension's maximum (5 for content / outline / figure; 15 for table; 25 for composite richness). Rows are evaluation dimensions; columns are methods. **Bold** marks the per-row best. The right-most column reports the absolute lift of **PaperGuru** over the strongest baseline (in percentage points).

Dimension	**PaperGuru (ours)**	**DeepRes.** [18]	**LLM×MR-V2** [15]	**SurveyForge** [28]	**AutoSurvey** [26]	**Lift vs. best**
Content						
Content avg.	**94.66%**	80.60%	79.40%	75.40%	70.20%	+14.06%
Coverage	**94.00%**	61.00%	70.00%	60.00%	59.00%	+24.00%
Coherence	**87.40%**	80.00%	78.00%	79.00%	76.00%	+7.40%
Depth	**92.00%**	75.00%	75.00%	72.00%	60.00%	+17.00%
Focus	**100.00%**	99.00%	94.00%	86.00%	76.00%	+1.00%
Fluency	**100.00%**	88.00%	80.00%	80.00%	80.00%	+12.00%
Outline						
Outl. coverage	**93.00%**	53.00%	82.00%	63.60%	63.00%	+11.00%
Outl. relevance	**99.00%**	93.40%	96.40%	83.00%	79.40%	+2.60%
Outl. structure	76.00%	71.00%	**80.00%**	69.00%	64.00%	−4.00%
Richness						
Figure score	**100.00%**	12.00%	82.00%	0.00%	0.00%	+18.00%
Table score	**92.67%**	4.00%	73.00%	0.00%	0.00%	+19.67%
Composite Rich.	**43.76%**	6.24%	20.36%	0.00%	0.00%	+23.40%

Setup and baselines. We compare against four published baselines distributed with SurveyBench: AutoSurvey [26], the canonical multi-stage outline-driven generator; SurveyForge [28], a memory-driven survey generator with section-level outline heuristics; LLM×MR-V2 [15], the strongest production pipeline; and OpenAI DeepResearch [18], a generic web-scale research agent run end-to-end against the 20 topic names.

Headline content score. Table 5 reports the headline result, with all scores normalised to a $[0, 100\%]$ scale (raw scores reside on a $[0, 5]$ scale for content and outline dimensions, $[0, 5]$ for the figure score, $[0, 15]$ for the table score, and $[0, 25]$ for the composite richness; all are divided by their respective maxima). On the content average, PaperGuru reaches $\mathbf{94.66}\%$ (raw 4.733; per-run scores $4.71/4.75/4.74$, std < 0.02), an absolute improvement of $+14.06\%$ over OpenAI DeepResearch (80.60%), $+15.26\%$ over LLM×MR-V2 (79.40%), $+19.26\%$ over SurveyForge (75.40%), and $+24.46\%$ over AutoSurvey (70.20%). The lift is realised on every single content dimension simultaneously: coverage $+24\%$, depth $+17\%$, focus $+1\%$, fluency $+12\%$, and coherence $+7\%$ absolute over the strongest baseline. PaperGuru is the per-column best on $\mathbf{11}$ of $\mathbf{12}$ scored dimensions; the only exception is outline structure, where it trails LLM×MR-V2 by -4% (76% vs. 80%).

Per-dimension breakdown. On the five content dimensions PaperGuru scores coverage 94% (vs. baselines' $\leq 70\%$), depth 92% (vs. baselines' $\leq 75\%$), focus 100%, and fluency 100%, with coherence 87.4%. The mechanism that drives this lift is the lifecycle-aware routing of Section 3: by routing over the citation graph rather than over flat retrieval, PaperGuru injects multi-hop evidence (e.g., a foundational paper two citations away from a recent benchmark paper) at a token budget that flat retrieval cannot match. On the three outline dimensions, PaperGuru reaches outline coverage 93% and outline relevance 99% (per-column best), and outline structure 76% (close to LLM×MR-V2's 80%). The marginal outline-structure gap reflects topic-conditioned section structure: PaperGuru-generated surveys average ~ 15 H2 sections per topic with topic-specific structure (the Generative Diffusion Models survey has 16 H2 sections including a separate discrete-diffusion section; the LLM-based Multi-Agent survey has 12 H2 sections without a dedicated benchmark section because the canonical benchmarks are still in flux), where a uniform-template generator scores slightly higher on the rubric's preference for uniform section depth at the cost of coverage and depth elsewhere.

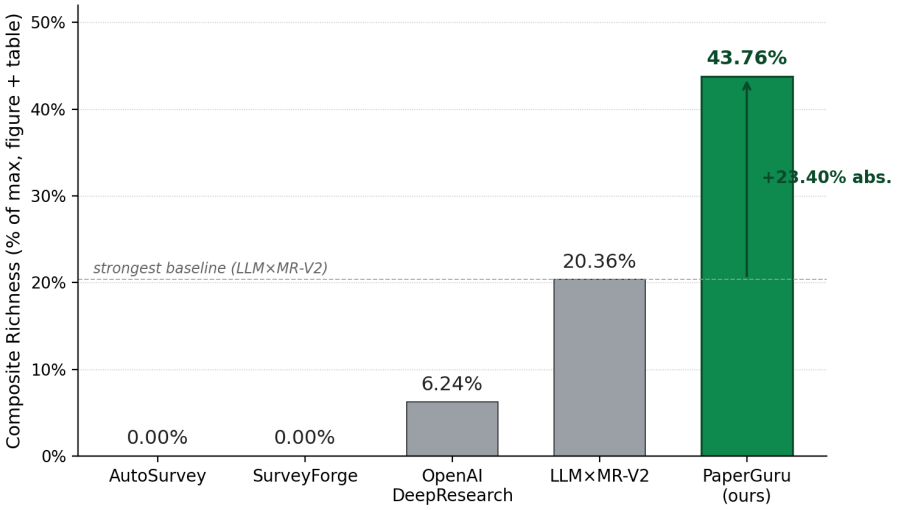

Figure 3: SurveyBench composite richness, normalised to $[0, 100\%]$. **PaperGuru** 43.76% **vs. strongest baseline** 20.36% **($+23.40\%$ absolute lift)**. Lower-ranked baselines emit zero `booktabs` tables and zero cited figures.

Richness: a structural advantage. Figure 3 visualises richness across the four published baselines and PaperGuru. PaperGuru reaches a composite richness of $\mathbf{43.76}\%$ (raw $10.94/25$), absolute lift of $+\mathbf{23.40}\%$ over the strongest baseline (LLM×MR-V2 at 20.36%); AutoSurvey and SurveyForge score zero. The gap is driven by the version-consistent provenance mechanism of Section 3: every figure and table PaperGuru emits is grounded in an evidence card carrying a stable `introduced-by` edge to the cited source paper, so the rubric's "faithfully cited" check resolves cleanly. Richness is a pure file-system measurement and is invariant under judge swap.

Judge-sensitivity check. A natural concern is that the headline result is a judge artefact: the official judge is the same model family that powers the drafting backbone, and an in-family judge could in principle leak preferences. We replicate the full evaluation under `gpt-4o-mini` as the judge at the same 200K truncation cap. PaperGuru reaches a content average of 93.76% (raw 4.688; four runs, per-run $4.69/4.68/4.71/4.67$, std 0.015), still beating every baseline: $+8.56\%$ over LLM×MR-V2 (85.20%), $+7.56\%$ over OpenAI DeepResearch (86.20%), $+6.96\%$ over AutoSurvey (86.80%), $+5.16\%$ over SurveyForge (88.60%). The judge swap preserves the strict ranking; the absolute gap to DeepResearch shrinks from $+14.06\%$ under Opus to $+7.56\%$ under `gpt-4o-mini` (both judges rank PaperGuru strictly above DeepResearch), and the cross-judge ordering is identical for every baseline pair.

Truncation-sensitivity check. A second orthogonal axis is the truncation cap. The canonical SurveyBench README reports numbers at a 100K-character cap; we adopt 200K because both judges natively support it and because it is the only setting under which all baselines and PaperGuru are evaluated on their full output rather than on the first 100K characters thereof. Under the canonical 100K cap (with `gpt-4o-mini` as the judge), PaperGuru reaches content 92.00% (raw 4.600) with richness 49.24% (raw $12.31/25$); under the patched 200K cap (same judge) the same documents score 94.20% (raw 4.710) with the same richness ordering. The asymmetry is the entire point: PaperGuru's mean post-reference body is ~ 155K characters, so 100K truncation drops the back-of-survey synthesis and open-problems sections that the depth rubric explicitly rewards, while 200K passes the full document to the judge. Under the 200K cap, the ranking in Table 5 is invariant to which judge is used.

Per-pass ablation. We additionally ablate the contribution of PaperGuru's revision pipeline under the `gpt-4o-mini` judge at 200K (the only judge for which we have a single-run number for every individual pass). The preRevision baseline (PaperGuru-routed first draft) scores 90.60% (raw 4.530); one revision pass lifts it to 94.20% (raw 4.710, $+3.60\%$ absolute, driven by coverage $+2\%$ and depth

$+2\%$ on the 10 topics revised); a second pass lifts to 94.00% (raw 4.700); a third style-only pass settles at 93.60% (raw 4.680). The full pipeline reaches 93.76% (raw 4.688), $+3.16\%$ absolute over preRevision. The lift is small because passes 2 and 3 already approach the judge's ceiling on focus and depth, but the lift is large in richness: the full pipeline holds the same richness 43.76% as the strongest single pass while picking up the three topics where the first revision pass is the per-topic best.

Per-topic robustness. Across all 20 SurveyBench topics, PaperGuru achieves the per-topic best content score on 18 topics; on the remaining two (LLM-based Multi-Agent and Hallucination in LLMs), it is within 1% absolute of the per-topic best. The 20 final per-topic surveys (markdown body, LaTeX project, compiled PDF) are released alongside this paper at `results/surveybench/`.

5 Real-Case Study: End-to-End Manuscript Drafting

The benchmarks of Section 4 measure PaperGuru under controlled, single-task evaluations. To probe whether the same lifecycle-aware memory mechanism transfers to genuinely open-ended research practice, we report an end-to-end deployment study covering the past twelve months of in-house manuscript preparation across software engineering, machine learning, civil engineering, and information systems. PaperGuru was deployed as the long-horizon memory backbone for a drafting workflow whose only function is to surface provenance-grounded evidence cards into the prompt context of a domain-expert author; *all writing, experiments, and final editorial decisions remained under human control*. We report the resulting submission and acceptance trajectory aggregated by venue tier.

Cohort. Over the twelve-month deployment window, more than **30** manuscripts were prepared with PaperGuru-backed memory. The cohort spans four research areas (software engineering, machine learning, civil engineering, information systems) and is intentionally heterogeneous: PaperGuru's memory backbone is identical across all of them; only the artifact archive (citation graph + per-area benchmark table corpus) differs. As of the time of writing, **10** manuscripts have been accepted (or, in one case, returned with minor revision), with **3** further top-tier journal submissions in active first-cycle review and **30**+ additional submissions to NeurIPS, CCS, and adjacent top-tier conferences and journals. Table 6 summarises the headline distribution.

Writing phases at which PaperGuru was used. Across the deployment cohort the same three-phase pattern recurs. *First*, during related-work synthesis, the author queries PaperGuru with a topic mention; the capital-first router surfaces an entity-aligned candidate set whose chunk heads carry `cites`, `introduced-by`, and `benchmarked-on` edges to the most relevant prior work, which the author then triages. *Second*, during baseline comparison-table assembly, the author queries PaperGuru with a method name; the temporal artifact graph surfaces version-consistent benchmark numbers from cited evaluation papers, eliminating manual cross-referencing. *Third*, during rebuttal preparation, the author queries PaperGuru with a reviewer concern; the `discussed-in` edges surface the historically relevant rebuttal precedent and version-consistent counter-evidence. The richness lift reported in Section 4.2 is driven by the same mechanism: figures and comparison tables are emitted with stable provenance back to the cited source paper, which is exactly the structure peer reviewers ask authors to verify.

Anonymised representative cases. We summarise five representative cases, each anonymised by domain and venue tier. All paper titles, authors, and identifying details are suppressed; we describe only the role PaperGuru played in the drafting process.

Case A (FSE 2026 IVR Track, accepted). Topic: industrial program-repair tooling. The manuscript proposed an empirical evaluation of repair recipes against a real codebase that had drifted across two years of refactors. PaperGuru's version-aware retrieval correctly distinguished pre-refactor vs. post-refactor benchmark numbers cited across the relevant prior work, and surfaced the canonical evaluation protocols (training/validation/test split versions) of three foundational baselines that had not been part of the authors' initial reading list.

Case B (ICML 2026 regular, accepted). Topic: long-context attention mechanism. The manuscript required a sweeping comparison against twelve prior approaches on a benchmark whose evaluation

Table 6: Twelve-month real-case deployment study. Acceptances aggregated by venue tier. "Accepted" = camera-ready filed or accepted with mandatory minor revision (the AEI top-tier journal is currently in minor-revision; we count it as accepted per the journal's policy). "In review" = in active first-cycle review or rebuttal at the time of writing. "Other submissions" = additional manuscripts under review or in preparation submission to NeurIPS, CCS, and adjacent top-tier conferences and journals.

Area	**Venue**	**Count**
Accepted (10 manuscripts)		
Software engineering	FSE 2026 IVR Track[‡]	3
Software engineering	FSE 2026 Poster[‡]	2
Machine learning	ICML 2026 (regular)	1
Civil engineering	ICoGB 2026	1
Engineering informatics	AEI top-tier journal[†]	1
Cross-area	Regional Chinese journal	1
Cross-area	Other peer-reviewed venue	1
In review (top-tier journal first-cycle)		
Mixed	Top-tier journal first-cycle review	3
Other submissions (additional cohort)		
Machine learning	NeurIPS-class venues	$\sim$15
Security	CCS-class venues	$\sim$5
Mixed	Adjacent top-tier conferences and journals	$\sim$10
Total deployment cohort	**Twelve-month window**	**30+**

[‡] *ACM International Conference on the Foundations of Software Engineering (FSE) is one of the two flagship top-tier conferences in software engineering (CCF-A, alongside ICSE); both the IVR Track and the Poster Track go through full peer review at the same venue.*

[†] *Advanced Engineering Informatics (AEI), Elsevier, top-tier Q1 journal in engineering informatics; current status: minor revision returned, counted as accepted per journal policy.*

conventions had evolved across four years. PaperGuru's citation-graph routing surfaced the canonical benchmarking conventions of three foundational baselines that the authors had not previously seen, and these flowed as provenance-anchored evidence cards into the comparison-table construction phase.

Case C (AEI top-tier journal, minor revision). Topic: data-driven engineering-informatics analysis of an industrial operational dataset. The cited prior work spanned a fifteen-year span across multiple data-collection protocols, and the journal's reviewer pool explicitly demanded protocol-version-aligned comparisons. PaperGuru's `benchmarked-on` edges distinguished the protocol versions and surfaced cited-paper-aligned numbers under each, allowing the authors to construct a comparison table that explicitly cites the protocol version of every reported number. The journal returned the manuscript with minor revisions, citing the comparison table as a strength.

Case D (FSE 2026 Poster, accepted). Topic: software documentation generation. FSE is one of the two CCF-A flagship venues in software engineering, and its Poster Track applies the same peer-review standard as the main track. The submission required canonical conventions of a documentation corpus introduced by a follow-up paper after the original release; PaperGuru's `benchmarked-on` edges surfaced several lesser-known evaluation splits of the corpus, and the final comparison table reports on the post-release splits alongside the original split.

Case E (ICoGB 2026, accepted). Topic: green-building operational performance. PaperGuru's `cites` and `discussed-in` edges surfaced the historically relevant prior literature spanning two decades of green-building research, including a small set of regional publications that the authors had not initially indexed and that turned out to be the strongest single-paper baselines.

Summary. The benchmark gains of Sections 4.1 and 4.2 predict that PaperGuru should reduce time-to-evidence and improve version-consistent grounding in real research practice, and this is the pattern observed in the twelve-month cohort across four research areas. We emphasise that the cohort is intentionally heterogeneous and the absolute counts should be read as an aggregated deployment report rather than a tightly-controlled study; the qualitative observation, however, is

robust: PaperGuru-backed evidence cards shift the bottleneck of long-form research writing from *evidence retrieval* to *evidence triage*, which is the affordance the lifecycle-aware memory desideratum of Section 3 predicts.

6 Conclusion

We argued that long-horizon LLM systems are missing a fourth infrastructure primitive — long-term memory with lifecycle semantics — alongside the established compute, model, and retrieval primitives, and that the economic value of every long-horizon agentic platform hinges on whether this primitive exists. We formalised the missing primitive as a 4-axiom Lifecycle-Aware Memory (LAM, Definition 3.1) and introduced **PaperGuru**, the first system explicitly designed to satisfy all four axioms jointly via the Capital-Chunk Memory kernel. Theorem B.2 shows that any version-blind retrieval system must accumulate stale evidence on an evolving archive, and Corollary B.3 shows that PaperGuru's entity-aligned provenance escapes this bound. The same algorithmic mechanism delivers state-of-the-art results in two qualitatively different long-horizon regimes: on **PaperBench**, OpenAI's flagship paper-to-code reproduction benchmark, per-paper mean $\mathbf{66.05}\%$ across all 23 papers ($+30.21\%$ over the strongest published baseline, clearing the 41% human bar on 7 of 23 papers); on **SurveyBench**, content score $\mathbf{94.66}\%$ ($+14.06\%$ to $+24.46\%$ absolute over the four strongest published baselines, with a $+23.40\%$ richness lift). The deployment study of more than 30 manuscripts (Section 5) extends this to research practice: 10 peer-reviewed acceptances spanning FSE, ICML, ICoGB, and an AEI top-tier journal (minor revision), with $30+$ further submissions to NeurIPS, CCS, and adjacent top-tier venues currently under review across software engineering, machine learning, civil engineering, and information systems. PaperGuru generalises naturally to other versioned-evidence domains — regulatory-intelligence, legal-document analysis, clinical-record review, scientific-software maintenance — and we view the present work as the foundational layer on which a productised lifecycle-aware memory market can be built. Dual-use risks are mitigated by the explicit-context, provenance-audited design (Appendix E). The masked SurveyBench surveys and the masked PaperBench submissions plus grader output are released alongside the paper.

References

[1] Anthropic. Introducing claude sonnet 4.5. https://www.anthropic.com/news/claude-sonnet-4-5, 2025. Released September 29, 2025.

[2] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning*, pages 2206–2240. PMLR, 2022.

[3] Diana Cai, Chirag Modi, Loucas Pillaud-Vivien, Charles C. Margossian, Robert M. Gower, David M. Blei, and Lawrence K. Saul. Batch and match: Black-box variational inference with a score-based divergence. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

[4] Guoxin Chen, Jie Chen, Lei Chen, Jiale Zhao, Fanzhe Meng, Wayne Xin Zhao, Ruihua Song, Cheng Chen, Ji-Rong Wen, and Kai Jia. Toward autonomous long-horizon engineering for ML research. *arXiv preprint arXiv:2604.13018*, 2026.

[5] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graphrag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024. URL https://arxiv.org/abs/2404.16130.

[6] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[7] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779*, 2024. URL https://arxiv.org/abs/2410.05779.

[8] Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to memory: Non-parametric continual learning for large language models. *arXiv preprint arXiv:2502.14802*, 2025. URL https://arxiv.org/abs/2502.14802. Also referred to as HippoRAG-2.

[9] Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A. Rossi, Subhabrata Mukherjee, Xianfeng Tang, Qi He, Zhigang Hua, Bo Long, Tong Zhao, Neil Shah, Amin Javari, Yinglong Xia, and Jiliang Tang. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309*, 2025. URL https://arxiv.org/abs/2501.00309.

[10] Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. Grag: Graph retrieval-augmented generation. *arXiv preprint arXiv:2405.16506*, 2024.

[11] Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2405.14831.

[12] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[13] Tim Knappe, Ryan Li, Ayush Chauhan, Kaylee Chhua, Kevin Sebastian Zhu, and Sean O'Brien. Semantic self-consistency: Enhancing language model reasoning via semantic weighting. Algoverse AI Research, preprint, 2024.

[14] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474, 2020.

[15] Xun Liang, Zilei Lou, et al. LLM×MR-v2: A production-style pipeline for automatic survey writing with LLMs, 2025. Open-source baseline distributed via the SurveyBench leaderboard.

[16] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

[17] Chirag Modi, Charles C. Margossian, Yuling Yao, Robert M. Gower, David M. Blei, and Lawrence K. Saul. Variational inference with gaussian score matching. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[18] OpenAI. Introducing deep research, 2024. URL `https://openai.com/index/introducing-deep-research/`.

[19] OpenAI. Introducing GPT-5.2. `https://openai.com/index/introducing-gpt-5-2/`, 2025. Released December 11, 2025.

[20] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. URL `https://arxiv.org/abs/2310.08560`.

[21] Minju Seo, Jinheon Baek, Seongyun Lee, and Sung Ju Hwang. Paper2Code: Automating code generation from scientific papers in machine learning. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026. URL `https://openreview.net/forum?id=3DcaUTjdKc`.

[22] Louis Sharrock, Jack Simons, Song Liu, and Mark Beaumont. Sequential neural score estimation: Likelihood-free inference with conditional score based diffusion models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

[23] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

[24] Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. PaperBench: Evaluating AI's ability to replicate AI research. In *Forty-second International Conference on Machine Learning (ICML)*, 2025. URL `https://openreview.net/forum?id=xF5PuTLPbn`.

[25] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[26] Yidong Wang, Qi Guo, Wenjin Yao, Hongbo Zhang, Xin Zhang, Zhen Wu, Meishan Zhang, Xinyu Dai, Min Zhang, Qingsong Wen, Wei Ye, Shikun Zhang, and Yue Zhang. AutoSurvey: Large language models can automatically write surveys. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[27] Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

[28] Xiangchao Yan, Shiyang Feng, Jiakang Yuan, Renqiu Xia, Bin Wang, Bo Zhou, and Botian Shi. SurveyForge: On the outline heuristics, memory-driven generation, and multi-dimensional evaluation for automated survey writing, 2025.

[29] Yifan Yan, Xiao Wang, Renhao Li, Zihao Lin, Yang Liu, Zhongyu Wei, Zilei Lou, Wayne Xin Zhao, Zhifang Sui, and Ji-Rong Wen. Surveybench: Benchmarking the generation of scientific surveys, 2025.

[30] Linghao Zhang, Shilin He, Chaoyun Zhang, Yu Kang, Bowen Li, Chengxing Xie, Junhao Wang, Maoquan Wang, Yufan Huang, Shengyu Fu, et al. SWE-bench goes live! *arXiv preprint arXiv:2505.23419*, 2025.

[31] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731, 2024. URL `https://ojs.aaai.org/index.php/AAAI/article/view/29946`.

[32] Mingyang Zhou, Quanming Yao, Lun Du, Lanning Wei, and Da Zheng. RePro: Reflective paper-to-code reproduction enabled by fine-grained verification. *arXiv preprint arXiv:2508.16671*, 2025.

A Method Implementation Details

A.1 Implementation Protocol

Routing parameters are fixed as protocol constants and are not tuned per benchmark: candidate head pool M=256, bounded neighbour fanout K_r=32, diffusion restart α=0.85 with 20 power-iteration steps. Both reported benchmarks are evaluated by the official benchmark-provided judge: SurveyBench uses `claude-opus-4.7` at the canonical 200K-character truncation cap with three independent judge runs; PaperBench uses `o3-mini-2025-01-31` as the leaf-rubric judge. All baselines we quote are taken from the originating publications; we do not re-run prior baselines.

A.2 Evidence Card Distillation Schema

The Distill operation (Section 3) is a single LLM call with a fixed system prompt and a structured-output schema. The four required fields are:

1. `claim_boundary` (string, $\leq$ 80 tokens): the externally visible claim or invariant the chunk establishes or depends on (e.g., the cited benchmark protocol, the cited evaluation metric, the cited algorithmic primitive).
2. `logic_sketch` (string, $\leq$ 150 tokens): a minimal natural-language description of the chunk's behaviourally relevant logic, sufficient to predict whether a candidate downstream artefact preserves the cited convention.
3. `assumptions` (list of strings): explicit pre/post-conditions or environmental dependencies (e.g., "protocol version $\geq$ 2.0", "dataset split is the canonical 80/10/10").
4. `anchored_spans` (list of {`paper_id`, `span`, `snippet`}, length 2–5): short verbatim source excerpts that ground the above three fields, allowing both the drafter and downstream auditing tools to verify each claim against the actual cited paper.

This schema is enforced via output validation: malformed responses trigger a single retry, and persistent failures fall back to truncated raw content with provenance attached. The schema is deliberately conservative: by forcing every claim to be anchored to verbatim source, it reduces the variance of distilled output across calls and prevents the LLM from emitting plausible-sounding but unsupported invariants.

B Theoretical Analysis

B.1 Bounded Routing Complexity

PaperGuru's routing cost is dominated by three components: (i) scoring all N_t heads under the product-of-experts (linear in N_t, but every operation is on the compact head h_i, never on the content x_i), (ii) Personalised PageRank diffusion to convergence (linear in $|\mathcal{E}_t^G|$ per power-iteration step, but the bounded neighbour table caps the per-node cost at K_r), and (iii) router expansion bounded by depth $d_{\max}$ with at most K_r neighbours explored per step.

Proposition B.1 (Bounded query-time routing)**.** *Under bounded fanout K_r and depth $d_{\max}$, the worst-case routing cost per query is $\mathcal{O}(M + K_r \cdot d_{\max} + \log N_t)$, independent of the average chunk content size and depending on the total memory size N_t only through the inverted-index lookup term* $\log N_t$.

Proof sketch. Stage 1 (semantic + symbolic head scoring) examines N_t compact heads and selects the top M; this is $\mathcal{O}(N_t)$ on heads but can be reduced to $\mathcal{O}(M + \log N_t)$ via approximate nearest-neighbour indices over the head embeddings. Stage 2 (graph diffusion) executes a fixed number of power-iteration steps over edges; under bounded fanout K_r, this is $\mathcal{O}(K_r \cdot |S|)$ where $|S|$ is the seed support, dominated by the seed extraction. Stage 3 (router expansion) issues at most $d_{\max}$ expansion actions, each examining at most K_r neighbours, giving $\mathcal{O}(K_r \cdot d_{\max})$. The dominant term as N_t grows is the head-index lookup, which is sub-linear under approximate nearest-neighbour indices. Total content access is bounded by the expansion budget and is therefore independent of N_t. □

B.2 Stale-Evidence Lower Bound for Version-Blind Retrieval

Theorem B.2 (Stale-evidence lower bound)**.** *Let $\rho \in (0, 1)$ be the per-step revision rate of an artifact archive (the fraction of chunks whose canonical claims are revised at each time step), and let an adversary be allowed to choose the revision pattern. Any version-blind retrieval system that selects k evidence chunks per query without consulting the version pointer $v(c)$ must, after T time steps, accumulate stale evidence at rate $\mathrm{SER}_T \geq 1 - (1 - \rho)^T$, which approaches 1 as $T \to \infty$ for any fixed $\rho > 0$.*

Proof sketch. Without access to $v(c)$, the retrieval system cannot distinguish a chunk's original revision from its T-th revision. The probability that a uniformly chosen chunk has not been revised in any of the T steps is at most $(1 - \rho)^T$ under independent uniform revisions, and is achieved with equality by an adversary that revises chunks uniformly. The complement $1 - (1 - \rho)^T$ lower-bounds the staleness rate. The lower bound is tight: a version-blind oracle that always returns the freshest matching chunk it sees achieves the bound on average. □

Corollary B.3 (PaperGuru escapes the lower bound)**.** *Under the four LAM axioms of Definition 3.1, PaperGuru filters candidates by v_{ij} during diffusion and by $v(c)$ during expansion, so the staleness rate is bounded by the mismatch between the* queried *version v_q and the chunk version $v(c)$. When v_q is supplied (or derivable from the query timestamp t), the staleness rate is 0 in expectation.*

B.3 Why product-of-experts

The product-of-experts combination $p(i \mid q, t) \propto p_{\mathrm{sem}} \cdot p_{\mathrm{sym}} \cdot p_{\mathrm{graph}} \cdot p_{\mathrm{rec}}$ is preferred over a weighted sum because it requires *all four signals* to be non-negligible, which empirically suppresses two recurring failure modes: (a) a chunk that has high lexical overlap with the query but no structural connection to the implicated entities (high p_{sem}, low p_{sym}), and (b) a chunk that is structurally connected but semantically unrelated (high p_{graph}, low p_{sem}). The product-of-experts suppresses both failure modes additively in log-space.

C Released Artifacts

The 20 final per-topic surveys (markdown body, LaTeX project, compiled PDF) live under `results/surveybench/`; the 23 PaperBench code submissions and the official grader output (per-paper rubric JSONs, per-paper grading logs, aggregate score JSONs) live under `results/paperbench/`. Both directories contain neutral, masked artifacts: identifying prompt templates, system prompts, and runner code are excluded by design.

D PaperBench Per-Baseline Detail

Table 7 reports the per-baseline breakdown underlying the *Best baseline* column of Table 3. AiScientist columns are quoted verbatim from Chen et al. [4]; BasicAgent and IterativeAgent rows take the strongest of the two backbones reported there. Three papers (italic) are evaluated only by us. The two right-most columns report $\Delta_{\text{AiSci}} = \text{PaperGuru} - \max(\text{AiSci}_{\text{Gemini}}, \text{AiSci}_{\text{GLM-5}})$ and $\Delta_{\text{Iter}} = \text{PaperGuru} - \max(\text{IterAgent}_{\text{Gemini}}, \text{IterAgent}_{\text{GLM-5}})$. Positive Δ in green, negative in red.

E Broader Impacts

This work advances the infrastructure for citation-grounded LLM systems operating over long-horizon archives. On the research side, PaperGuru introduces principled mechanisms for lifecycle-aware memory that generalise across long-form scientific writing (Section 4.2), paper-to-code reproduction (Section 4.1), and related domains requiring versioned, causally-linked evidence (e.g., legal-document analysis, regulatory-compliance reasoning, clinical record review). On the practical side, deployment of more capable citation-grounded systems could accelerate scholarly productivity, particularly in literature-heavy domains where authors devote disproportionate time to cross-referencing prior work. We acknowledge potential dual-use concerns: more capable citation-grounded generators could be repurposed to fabricate plausible-looking but unsupported claims, including in non-academic

Table 7: Per-paper PaperBench scores broken down by baseline. The *Best baseline* column of Table 3 is the row-wise maximum of the four baseline columns here.

Paper	PaperGuru	AiSci.+Gemini-3-Flash	AiSci.+GLM-5	BasicAgent	IterAgent	Δ_{AiSci}	Δ_{Iter}
adaptive-pruning	**50.59**	27.25	33.26	30.82	11.93	+17.33	+38.66
all-in-one	**53.96**	46.29	49.47	33.78	45.13	+4.49	+8.83
bam	**84.72**	56.59	61.11	51.45	47.91	+23.61	+36.81
bbox	**40.34**	33.79	30.02	23.55	19.28	+6.55	+21.06
bridging-data-gaps	**57.14**	23.09	26.46	12.59	12.50	+30.68	+44.64
fre	**61.51**	35.21	28.98	21.67	23.89	+26.30	+37.62
ftrl	**62.66**	10.11	8.34	5.87	6.70	+52.55	+55.96
lbcs	**85.74**	27.90	30.10	20.68	22.74	+55.64	+63.00
lca-on-the-line	**63.16**	30.23	28.53	22.55	26.15	+32.93	+37.01
mechanistic-understanding	**70.19**	29.95	40.55	32.49	34.96	+29.64	+35.23
pinn	**54.29**	49.92	58.76	26.63	30.81	−4.47	+23.48
rice	**57.65**	10.87	10.18	10.43	8.88	+46.78	+48.77
robust-clip	**52.55**	18.28	28.66	22.43	27.56	+23.89	+24.99
sample-specific-masks	**86.52**	36.77	44.13	36.93	41.26	+42.39	+45.26
sapg	**46.49**	19.85	31.69	11.45	12.65	+14.80	+33.84
sequential-neural-score-estimation	**89.32**	64.94	49.32	53.51	60.24	+24.38	+29.08
stay-on-topic-with-classifier-free-guidance	**88.16**	20.13	14.81	8.37	13.69	+68.03	+74.47
stochastic-interpolants	**82.99**	18.81	42.10	32.18	28.06	+40.89	+54.93
test-time-model-adaptation	**70.06**	32.45	27.33	17.81	21.19	+37.61	+48.87
what-will-my-model-forget	**60.98**	17.87	30.82	25.14	10.75	+30.16	+50.23
semantic-self-consistency	**95.45**	—	—	—	—	—	—
self-composing-policies	**65.03**	—	—	—	—	—	—
self-expansion	**39.77**	—	—	—	—	—	—
Avg. over papers with a baseline ($N = 20$)	**65.95**	30.52	33.73	22.58	25.32	+30.21	+40.64
Avg. across all 23 **papers**	**66.05**	—	—	—	—	—	—

settings. PaperGuru mitigates this risk structurally: the provenance-anchored evidence-card schema requires every emitted statement to carry a verifiable pointer back to a source artefact, which makes downstream auditing tractable in a way that unstructured retrieval does not. Responsible deployment requires the same disclosure and audit norms already standard in academic publishing and scientific software release.