JooMMA: A Provenance-Preserving Knowledge Layer for Decentralized Language Intelligence

Version 1.0 – May 2026

Hassan Habib
BestBytes AI LLC
www.peerllm.com
hassan@bestbytes.ai

***Abstract*—JooMMA is not a bulk training corpus but a provenance-preserving knowledge layer for decentralized language systems. Its central claim is that the data side of artificial intelligence should be coordinated in the same economic structure as decentralized compute: contributors who supply useful work should be paid when that work is used, and the system should preserve attribution rather than erase it inside model weights. Existing language models concentrate value by scraping broad public corpora, internalizing the benefit, and making lineage difficult to audit. JooMMA reverses that flow at inference time. It treats an unanswered or weakly grounded request as a knowledge gap, routes the gap to a domain-verified contributor, validates the response, stores it as a semantic shard, and compensates the author and serving hosts whenever the shard is retrieved to ground a model answer.**

JooMMA is deliberately hybrid. Today the coordination server performs identity, routing, validation assignment, retrieval authorization, and accounting, while knowledge is authored and hosted by independent participants. This paper formalizes the design as the data layer of PeerLLM and the retrieval companion to LLooMA. It contributes four main ideas: (i) pull-based knowledge acquisition driven by observed user demand, (ii) domain-scoped expertise verification and two-layer answer validation, (iii) semantic shard storage with attribution, fallback hosting, and dissent records, and (iv) retrieval-token accounting that splits value between authors and hosts. Because a public deployment trace is not yet available, the paper also specifies the metrics that a real JooMMA deployment should report.

***Index Terms*—Decentralized AI, retrieval-augmented generation, data provenance, attribution, expert networks, knowledge markets, PeerLLM, JooMMA.**

I. Introduction

JooMMA[1] starts from the data-side analogue of the LLooMA premise. A language service is not only a model. It is an operating system around two scarce resources: compute and knowledge. Compute determines how fast and how deeply a request can be executed. Knowledge determines what the system can responsibly say. LLooMA addresses the first resource by routing inference over a network of community-operated hosts. JooMMA addresses the second by routing knowledge gaps to verified human contributors and preserving the lineage of what they contribute.

The problem is not merely that model training uses large quantities of public data. The deeper problem is that ordinary training collapses authorship into weights. Once a sentence, argument, observation, or explanation has been absorbed into a model, the system usually cannot say which human contribution supplied which part of an answer. That makes payment, correction, consent, and accountability difficult to implement after the fact. A model may answer well, but the answer came from somewhere, and the person who supplied the underlying expertise is often made invisible.

JooMMA does not claim to solve training-data provenance retroactively. Instead, it moves the attribution problem to a place where it can be made explicit: inference-time grounding. When PeerLLM receives a question, the retrieval layer asks whether verified knowledge already exists. If it does, the answer is grounded in attributed shards. If it does not, the missing knowledge becomes a task. The network identifies the relevant domain, routes the task to a verified contributor, validates the answer, and turns the accepted answer into a retrievable shard. Over time, repeated use converts anonymous dependency on inherited model weights into auditable dependency on human-authored knowledge.

The paper's core thesis is therefore simple: JooMMA is a pull-based, attribution-preserving data protocol rather than a scraped dataset, and data decentralization is a first-class systems axis next to compute decentralization in the broader PeerLLM economy [1]. In a decentralized AI economy, the system should not only decide which host performs inference. It should also decide which knowledge is trusted, who authored it, which hosts served it, how dissent is surfaced, and how value is split when it is used.

[1]JooMMA is a sibling name to LLooMA: the same syllabic structure and double-letter visual signature, paired as the data and compute layers of PeerLLM. The name itself is a personal one, retained from the author's home. In PeerLLM, JooMMA is the knowledge substrate that LLooMA's compute draws from at inference time.

II. JooMMA Knowledge Model

Operationally, JooMMA implements a lifecycle from gap detection to payout accounting. Five mechanisms define most of its technical character: pull-based question sourcing, domain-scoped contributor verification, two-layer validation, semantic shard storage, and retrieval-token accounting. Where the public design is intentionally high-level, this section proposes concrete instantiations consistent with the architecture.

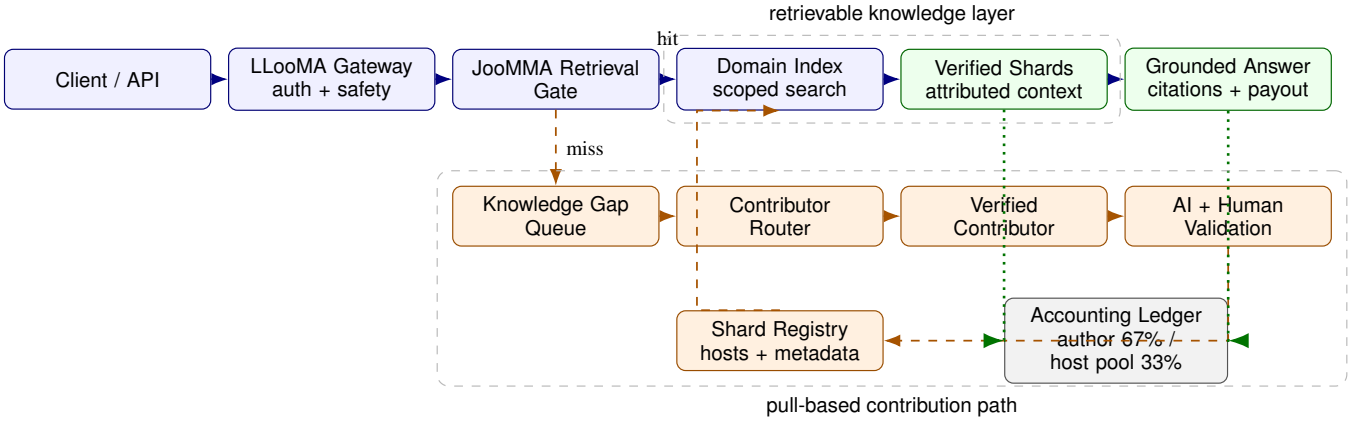

Fig. 1. JooMMA's inference-time knowledge lifecycle. A retrieval hit supplies attributed shards to the answering model. A retrieval miss becomes a knowledge task, is routed to a domain-verified contributor, validated, registered, hosted, and later retrieved with author and host accounting.

A. Pull-Based Gap Formation

Conventional training is a push system. Content is collected in advance, mixed into a corpus, and compressed into parameters. JooMMA is a pull system. It begins with observed demand. When LLooMA receives a request and the retrieval layer cannot find a verified shard that sufficiently grounds the answer, the request is logged as a knowledge gap. The model may still answer from its inherited weights, but the gap is now visible to the network.

The gap queue has two sources. The first is user demand: questions that arrive naturally and expose missing coverage. The second is structural audit: domains whose shard population is thin, stale, over-concentrated, or dependent on too few verified contributors. The queue therefore grows from both real usage and explicit coverage analysis. This avoids the need to guess in advance what should be scraped or stored.

B. Semantic Shards

A JooMMA shard is a meaningful, self-contained unit of verified knowledge. It is typically a paragraph, answer segment, worked explanation, structured claim set, or compact reference note that can be retrieved independently without losing its interpretation. Semantic sharding is distinct from cryptographic sharding. The goal is not to hide every byte from every storage node. The goal is to preserve retrievability, attribution, and reviewability while limiting the amount of any contributor's body of work held by a single fallback host.

Each shard carries metadata: author identity or anonymous contributor identifier, verified domain, validation status, reviewer lineage, timestamps, revision history, retrieval count, usefulness signals, hosting set, and any dissent record. The content and metadata together form the auditable object. A shard without provenance is merely text; a JooMMA shard is text with accountability.

C. Relationship to LLooMA

JooMMA is the data layer of PeerLLM and is intended to be retrieved by LLooMA at generation time [2], [3]. The request first enters the LLooMA control plane, which decides how inference should be executed. Before generation begins, the retrieval layer queries JooMMA for relevant verified shards. If shards are returned, they are inserted into the model context and the response cites the contributors whose knowledge was used. If no shard is returned, the request may still be answered, but the missing coverage is recorded as a candidate task. At the moment of generation, the user pays once, and the coordination server splits the payment between the LLooMA compute host, per the LLooMA economic model, and the JooMMA contributors and hosts whose shards were retrieved into context, per the rules in Section V.

The layers are independent in operation but coupled in purpose. LLooMA can run without JooMMA by relying on model weights. JooMMA without LLooMA is a knowledge archive with no native execution surface. Together they form a system in which both execution and grounding are coordinated, priced, and attributed.

III. Contributor Verification and Routing

A. Domain-Scoped Verification

JooMMA does not treat expertise as a global property. Verification is scoped to a domain in the knowledge index. A contributor verified in cardiology is not thereby verified in constitutional law, and a contributor verified in one programming language is not automatically verified in all software engineering topics. Multiple domains are possible, but each requires its own approval path.

Phase 1 uses manual verification. The contributor supplies credentials, references, representative work, or other domain evidence. The coordination team checks those materials

against verifiable sources, administers a domain test, and, where the field warrants it, conducts an online interview with a reviewer. Manual approval slows growth, but it keeps the initial network small and trusted. That tradeoff is intentional: a small verified network is more useful than a large unverifiable one.

Verification is also not permanent. Credentials can lapse, domain knowledge can become stale, and bad actors can pass an initial gate. JooMMA therefore uses continuing assessment through adversarial probing, reviewer feedback, shard usefulness, and reputation. The result is a verification status that remains current rather than a one-time badge.

B. Question Routing

A pending question is routed by domain, urgency, contributor availability, reputation, and conflict constraints. A concrete ranking rule is to choose the contributor c that minimizes

$$S(c, q) = \lambda_1 d(c, q) + \lambda_2 a(c) + \lambda_3 r(c) + \lambda_4 o(c), \quad (1)$$

where $d(c, q)$ is domain distance between the contributor's verified scope and the question, $a(c)$ is availability delay, $r(c)$ is a risk term derived from recent review outcomes, and $o(c)$ is overload. The exact weights are deployment parameters rather than protocol constants.

The router should not deterministically assign all valuable work to the highest-ranked expert. That would produce concentration, reduce network resilience, and weaken the long-term incentive for new contributors. A practical policy is to filter by hard eligibility, then sample from a high-quality bucket while excluding reviewers or contributors with overlapping reference networks when validation independence matters.

The bucket size and sampling distribution are deployment parameters that trade off retrieval quality against contributor diversity. A narrower bucket favors top-ranked experts and concentrates retrieval payments; a wider bucket distributes work and earnings more evenly across the verified contributor pool. Phase 1 favors the wider configuration to preserve incentive for new contributors.

IV. Validation, Storage, and Retrieval

A. Two-Layer Validation

A submitted answer is not accepted on the authority of the contributor alone. Validation has two layers. The automated layer checks internal consistency, cross-references existing verified shards, flags contradictions, tests structured claims where possible, and detects likely plagiarism or corpus laundering. This layer is fast and useful for obvious errors, but it is not a substitute for expert judgment.

The human layer assigns a different verified expert in the same domain to review the submission. The reviewer may approve, reject, request revision, or attach a dissent record. High-stakes domains can require multiple independent reviewers. In some cases, the reviewer may conduct a short live interview to confirm that the submission reflects genuine working knowledge rather than copied material. Only after the required validation path completes does the answer become an active shard.

B. Hosting and Availability

The original contributor is the primary host of their own shards. A copy is also distributed to opt-in fallback hosts that store and serve shards in exchange for a share of retrieval revenue. Fallback hosts improve availability when a primary host is offline and provide redundancy against single-node failure. They do not need to hold complete bodies of work. The registry can distribute shards so that no single fallback host reconstructs a contributor's full corpus.

This is a mediated decentralization model. The coordination server is trusted for identity, routing authorization, shard registry maintenance, and payment accounting. Hosts are trusted to serve registered shards but are not trusted to assign themselves retrievals, alter provenance, or bypass accounting. Future versions may decentralize more of this control plane, but Phase 1 favors deterministic coordination over ideological purity.

C. Retrieval Policy

At inference time, retrieval should be both relevance-aware and provenance-aware. A concrete scoring rule for a shard s and query q is

$$R(s, q) = w_1 \operatorname{sim}(q, s) + w_2 Q(s) + w_3 F(s) - w_4 A(s) - w_5 C(s), \quad (2)$$

where sim is semantic similarity, Q is a quality signal derived from review outcomes and downstream usefulness, F is freshness, A is an aging or staleness penalty, and C is a conflict or caution term. A dissent record should not necessarily suppress retrieval; in contested domains it may increase the value of presenting multiple grounded positions. The score therefore determines the retrieval set, while the answer policy determines how agreement and disagreement are surfaced.

A shard is paid when it is retrieved and inserted into context, not only when the final answer visibly quotes it. This is a simpler and more auditable rule than attempting to infer causal influence after generation. The retrieval ledger can record the shard identifiers, token counts, serving hosts, request identifier, and final attribution surface. This "pay on retrieval insertion" rule is the central accounting principle of JooMMA: it replaces the philosophically difficult question of which shard caused which token in the final answer with an auditable mechanical rule, and it aligns contributor incentives with what the network can actually measure.

V. Attribution and Economics

The economics of JooMMA are built around one principle: every party that contributes to the value of an answer should be compensated when that contribution is used. Authors create the knowledge. Hosts make it available. The coordination layer routes, validates, and accounts for the transaction. The user pays once; the network splits the payment.

TABLE I
PHASE 1 ACCOUNTING RULES

Rule	Purpose
Pay on retrieval insertion	Keeps accounting auditable without post-hoc causal attribution.
Author share: 67%	Rewards the scarce contribution: verified knowledge creation.
Host pool: 33%	Rewards availability, redundancy, and serving work.
Offline author payout	Author is paid even when a fallback host serves the shard.
Flat subscription	Funds coordination and deters low-effort participation without scaling against contributor success.
Anonymous attribution option	Preserves payment and verification while hiding public identity when requested.

A. Retrieval-Token Accounting

Let n_s denote the number of tokens from shard s inserted into a model context for a retrieval event, and let c_{tok} be the retrieval-token rate. The gross shard payment is

$$U_s = c_{tok} n_s. \tag{3}$$

JooMMA then divides U_s into author and host components:

$$P_{author}(s) = \alpha U_s, \tag{4}$$

$$P_{hosts}(s) = (1 - \alpha) U_s, \tag{5}$$

α is a protocol parameter. Phase 1 fixes $\alpha = 0.67$ to reflect the substitutability argument: hosting capacity can be added more easily than domain-verified expert knowledge, so the scarcer contribution receives the larger share. Future phases may revisit α as the verified contributor pool and the host population evolve.

B. Offline Authors and Subscription

Authors do not have to be online to be paid. If a fallback host serves a shard while the primary author node is offline, the author still receives the author share. Authorship is the source of the payment, not the act of serving bytes at that moment. This mirrors royalty logic in other knowledge industries: a writer does not need to operate a bookstore in order to be paid for a book sale.

The condition is continued participation in the network. Phase 1 uses a flat participant subscription to fund coordination, identity, validation operations, and accounting. The subscription is not a tax on success; it is a low-friction mechanism for keeping the coordination layer available and for deterring throwaway participation.

C. Attribution and Anonymity

Attribution is the default because it is the point of the system. When a contributor's shard grounds an answer, the answer should surface the contributor or the contributor's anonymous identifier, depending on their preference. Anonymous contributors remain verified and paid in the internal ledger. Their public identity is simply not shown in the user-facing response.

The system should also attribute reviewers when appropriate. In ordinary cases, reviewer identity may remain internal. In contested knowledge, a dissent record may surface the reviewer, the reviewer's verified domain, or an anonymous verified reviewer label. The important property is that disagreement is attached to provenance rather than silently flattened into a single confident answer.

VI. TRUST AND INTEGRITY

Any network that pays for contributions creates incentives to game contribution, validation, hosting, or retrieval. JooMMA's Phase 1 defense is intentionally conservative: high-friction verification, controlled routing, independent review assignment, corpus-overlap checks, reputation adjustment, and slashing for bad-faith behavior.

A. Sybil Resistance and Collusion

Sybil resistance begins at the gate. Manual approval, credentials, references, testing, and interviews make it expensive to create a verified identity. The system does not merely check that an account exists; it checks that a person can credibly contribute in a scoped domain.

Collusion risk appears during validation. The coordination server should randomly assign reviewers from the eligible domain pool, prevent contributors from choosing reviewers, and exclude reviewers whose reference network overlaps with the contributor when such overlap is known. High-stakes domains should require multiple reviewers. Automated validation acts as a background check rather than the final authority, flagging cases where human approvals diverge from expected evidence.

B. Plagiarism and Laundering

A contributor who submits scraped content under their own name undermines the purpose of JooMMA. The automated layer should compare submissions against existing shards and public corpora where available. Suspected overlap triggers rejection, revision, or a provenance challenge. After publication, verified experts can raise authorship objections during a challenge window. The system should distinguish legitimate quotation or reference from copied authorship, but copied work cannot become paid original knowledge.

C. Reputation, Slashing, and Probing

Each shard accumulates retrieval frequency, usefulness ratings, revision history, reviewer outcomes, and dispute outcomes. These signals update contributor reputation. Reputation gates submission rate and routing priority. A contributor whose shards consistently rate poorly or fail review is throttled. A contributor found to have acted in bad faith can have shards removed from active retrieval, unvested earnings reversed, and verification revoked.

Adversarial probing handles expertise decay. The coordination server periodically routes known-answer or reviewer-designed questions to verified contributors. These probes are not meant to punish honest uncertainty. They are continuing assessment for a network whose value depends on current professional credibility.

VII. Contested Knowledge

Real expertise contains disagreement. Treating every domain as if it has one settled answer is a failure mode of many language systems. JooMMA handles disagreement as metadata rather than noise. If a reviewer rejects an answer and the contributor stands by it, the system can publish the shard with a dissent record instead of pretending the conflict does not exist. The dissent record should include the substance of the objection, the review domain, the validation status, and any limits on use.

The answering model must then surface the disagreement when the shard is retrieved. In low-stakes domains, this may be a short note that qualified reviewers disagree. In high-stakes domains, the answer may present multiple positions, explain the reasoning on each side, and avoid giving a single prescriptive recommendation. This approach treats the user as capable of understanding dispute while keeping the system honest about the state of knowledge.

A dissent record is a structured object carrying: the original shard identifier, the reviewer's verified domain, the substance of the objection in the reviewer's own words, the validation outcome at time of publication, and any limits on use. At retrieval time, the answering model surfaces both the shard and the dissent in a single attributed citation block. For example: "Source: Contributor A (verified, cardiology). Dissent: Reviewer B (verified, cardiology) disagrees with the recommended dosage range; see dissent record." In low-stakes domains a one-line dissent note is sufficient. In high-stakes domains the answer presents both positions, the reasoning on each side, and avoids a single prescriptive recommendation.

VIII. Roadmap and Deployment Metrics

JooMMA should launch narrowly. The first deployment should use a small set of domains, a small contributor pool, and an explicit validation capacity limit. The goal is not maximum growth; it is proof that the lifecycle works end to end: gap detection, routing, validation, sharding, retrieval, attribution, hosting, and payout.

Later phases can add domains, accelerate verification with AI assistance while preserving human approval, support structured and multimedia knowledge, and improve real-time updates for domains where currency matters. The third PeerLLM phase, tooling, can then apply the same economic pattern to actions and capabilities: independent operators contribute useful work and are paid per use.

IX. Related Work

JooMMA intersects three literatures but is not reducible to any one of them. Retrieval-augmented generation grounds model outputs by retrieving external passages at inference time [4]. JooMMA shares that pattern, but its retrieved unit is not merely context; it is a verified, attributable, hosted, and paid contribution.

Dataset documentation, model reporting, and provenance systems explain how data and models should be described [5]–[7]. JooMMA makes provenance operational. It uses lineage not only for disclosure, but also for routing trust, surfacing disagreement, and paying the contributor whose shard grounded an answer.

TABLE II
Minimum Metrics for a JooMMA Deployment

Metric	Definition and why it matters
Gap closure rate	Fraction of observed knowledge gaps converted into active validated shards.
Validation outcome mix	Approval, rejection, revision, and dissent rates by domain. Reveals review quality and domain difficulty.
Retrieval latency	p50/p95/p99 time added by domain search, shard fetch, and context assembly.
Attribution coverage	Share of grounded answers that surface contributor attribution or anonymous verified identifiers.
Payout distribution	Author and host earnings by domain. Reveals concentration or underpayment.
Usefulness signal	User and downstream rating of answers grounded in retrieved shards.
Integrity events	Plagiarism findings, slashing actions, reviewer conflicts, and successful appeals.

Online expert networks and community knowledge platforms have generated a literature on reputation, answer quality, and reviewer dynamics in human-authored knowledge systems [8]. JooMMA shares the concern with expert credibility and validated answers, but differs in two respects: contributions are authored in response to system-detected gaps rather than free-form posts, and contributors are compensated per retrieval rather than rewarded with reputation alone.

Decentralized inference work in PeerLLM treats language execution as network orchestration over heterogeneous hosts. JooMMA applies the same network-native framing to knowledge. Its novelty is the combination of pull-based gap formation, domain-scoped human verification, two-layer validation, semantic shard hosting, dissent-aware retrieval, and retrieval-token payout accounting within one data layer.

X. Conclusion

JooMMA is a policy layer for the knowledge side of decentralized AI. It converts a missing answer into a routed expert task, converts a validated expert response into a retrievable shard, and converts retrieval into attribution and payment. The central claim is that the people whose knowledge makes artificial intelligence useful should not be erased by the systems that use it.

The architecture is intentionally pragmatic. A centralized coordinator handles identity, validation assignment, retrieval authorization, and accounting in Phase 1, while independent contributors author knowledge and independent hosts serve it. In PeerLLM, LLooMA decentralizes execution and JooMMA decentralizes attributed grounding. In a network-native language system, knowing who supplied the knowledge, how it was validated, and how value flows back to that contributor is not an implementation detail. It is the system.

Acknowledgment

The author thanks the PeerLLM contributors who tested early validation flows and surfaced the operational requirements that shaped this layered design, and the open-source AI community whose practical constraints continue to inform PeerLLM's direction.

REFERENCES

[1] H. Habib, "PeerLLM: A Manifesto for a Decentralized AI Economy," PeerLLM Technical Report PLLM-TR-2025-01, Aug. 2025.

[2] H. Habib, "LLooMA: A Network-Native Orchestration Architecture for Low-Latency Distributed Language Intelligence," PeerLLM Technical Report, Apr. 2026.

[3] PeerLLM, "LLooMA 1.0: Architecture and Request Lifecycle," PeerLLM Technical Report PLLM-TR-2026-01, Apr. 2026.

[4] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Proc. NeurIPS*, 2020.

[5] T. Gebru et al., "Datasheets for Datasets," *Communications of the ACM*, vol. 64, no. 12, pp. 86–92, 2021.

[6] M. Mitchell et al., "Model Cards for Model Reporting," in *Proc. FAT**, 2019.

[7] W3C, "PROV-DM: The PROV Data Model," W3C Recommendation, 2013.

[8] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec, "Discovering Value from Community Activity on a Knowledge Sharing Site: Stack Overflow," in *Proc. ACM SIGKDD*, 2012.