Open Hallucination Index (OHI)

A Sovereign Architectural Framework for Restoring Epistemic Integrity in the Era of Probabilistic Generative AI

Fabian Zimber
Lead Developer, shiftbloom studio.
fabian@shiftbloom.studio

January 7, 2026

Abstract

Abstract: The rapid adoption of Large Language Models (LLMs) has precipitated an epistemic crisis, characterized by the proliferation of "hallucinations"—synthetically generated statements that lack factual grounding. This paper introduces the Open Hallucination Index (OHI), a comprehensive architectural framework designed to transition the industry from "Generative AI" to "Verifiable AI." Unlike traditional Retrieval-Augmented Generation (RAG), which remains susceptible to the stochastic nature of the generator, OHI establishes a deterministic "Trust Layer" that operates post-generation. By leveraging *Claim Decomposition*, a hybrid *GraphRAG* approach utilizing Neo4j and Qdrant, and the standardized *Model Context Protocol (MCP)*, OHI provides a quantifiable, audit-proof metric for truth. We present a reference implementation demonstrating local sovereignty using vLLM, detail the algorithmic rigor of the *Hybrid Verification Oracle*, and analyze the system's performance in handling high-throughput verification tasks without reliance on proprietary black-box models.

1 Introduction: The Epistemic Deficit

1.1 The Ontology of Stochastic Fabulation

The current technological landscape is marked by a fundamental disruption in the production and distribution of knowledge. With the ubiquitous spread of Large Language Models (LLMs), the paradigm of information genesis has shifted from the deterministic querying of curated databases to probabilistic synthesis. While this shift has unlocked unprecedented capabilities in generating natural language, it confronts society with an epistemological crisis: the erosion of factual truth through "hallucinations."

As noted by Bender et al. in their seminal work on "Stochastic Parrots" [1], LLMs operate as statistical pattern recognition systems that learn probability distributions over token sequences without a referential anchoring in extra-linguistic reality. They mimic the discourse of truth without having access to truth itself. This decoupling of form from content is an inherent feature of the Transformer architecture, optimized for plausibility rather than veracity. In critical domains such as jurisprudence, medicine, and engineering, the cost of such "stochastic integrity" is prohibitive.

1.2 Epistemic Vigilance and the Trust Dilemma

The necessity of a verification framework arises not only from technical deficits but from the psychological interaction between humans and machines. Cognitive science suggests that users tend to attribute an undeserved "epistemic authority" to AI systems. Jäger [2] defines this as a functional role based on the attribution of competence and reliability. LLMs simulate this competence through linguistic fluency, leading to "automation bias"—the tendency to trust algorithmic decisions even in the face of contradictory evidence.

Furthermore, "deferred trust" occurs when users shift their epistemic dependency from perceived biased human sources to AI systems, which are falsely viewed as neutral actors. While studies show an epistemic bias in favor of AI in factual domains, this trust is "brittle"; a single obvious error can collapse the entire system's credibility once "epistemic vigilance" is reactivated. OHI aims to institutionalize this vigilance technically.

1.3 Political Sovereignty and Proprietary Biases

A critical dimension of the hallucination debate is the political nature of the "truth" generated by LLMs. Proprietary models (Closed Source) undergo opaque Reinforcement Learning from Human Feedback (RLHF) processes, which inevitably encode the ideological, cultural, and political biases of their developers. Stud-

ies have shown significant ideological skewing in models like GPT-4, often favoring specific worldviews or failing systematically on sensitive topics.

In this context, hallucination becomes a political problem: if a model fabricates facts that correspond with embedded biases, disinformation is scaled. OHI addresses this through "local sovereignty." By utilizing open architectures (Open Source models like Qwen, local knowledge graphs), organizations can reclaim sovereignty over the "Ground Truth." Instead of relying on a model trained in California to know the laws of another nation, OHI enforces validation against a locally controlled data basis.

1.4 The Failure of Naive RAG

Retrieval-Augmented Generation (RAG) was proposed to ground LLM responses in retrieved context. However, "Naive RAG" remains susceptible to recursive stochasticity: the retrieval (vector similarity) may fetch irrelevant chunks, and the LLM may still prioritize its parametric memory or misinterpret valid context. Therefore, verification must be extrinsic and deterministic.

1.5 OHI: The Trust Layer

The Open Hallucination Index (OHI) provides an architectural solution by establishing a **Trust Layer** post-generation. Operating under the motto *"LLMs Hallucinate – We Verify,"* it audits statements using structured knowledge graphs to calculate a transparent trust score (0.0–1.0). This transforms blind trust into verified trust, making the epistemic quality of AI outputs as transparent as nutritional labels on food.

2 Theoretical Framework

2.1 Atomic Claim Decomposition

Verification requires granularity. A paragraph may contain several correct facts and one subtle fabrication; thus, a binary classification ("True" vs. "False") is insufficient. OHI employs **Atomic Claim Decomposition**, a method formalized in metrics like *FActScore* [3].

Let T be the generated text. We define a mapping function $f_{decomp} : T \to A = \{c_1, c_2, \ldots, c_n\}$ where each c_i is an atomic claim tuple (S, P, O) (Subject, Predicate, Object). The process involves:

1. **Segmentation:** Dividing text into discrete sentences.
2. **Atomization:** Breaking sentences into the smallest independent information units.
3. **Normalization:** Resolving pronouns and entities to a canonical form (Entity Linking).

The model's precision P is calculated as the ratio of supported atomic facts:

$$\text{FActScore} = \frac{1}{|A|} \sum_{c \in A} \mathbb{1}(c \text{ is supported}) \tag{1}$$

2.2 The Insufficiency of the Vector Space

While VectorRAG is the current industry standard, it suffers from fundamental epistemological limitations:

- **Flat Knowledge Space:** Vectors lack explicit connections. Answering questions requiring multi-hop reasoning often fails because the system cannot traverse non-existent links between isolated chunks.
- **Semantic Similarity vs. Logical Truth:** Vectors encode topical proximity, not causality. The statements "X causes Y" and "X prevents Y" often cluster together, confusing the LLM.
- **Context Fragmentation:** "Chunking" documents destroys macroscopic context, leading to interpretations based on incomplete information.

Empirical benchmarks highlight these shortcomings (Table 1). While VectorRAG is suitable for "fuzzy" searches, it is inadequate for the precise verification required by the OHI.

Table 1: Accuracy Benchmarks: VectorRAG vs. GraphRAG [7]

Scenario	VectorRAG	GraphRAG	Δ
Industry Spec.	65.6%	90.6%	+25.0%
Temporal Reason.	50.0%	83.4%	+33.4%
Numerical Reason.	< 80.0%	100.0%	> 20.0%
Tabular Reason.	33.0%	33.0%	0%

2.3 GraphRAG as a Correktiv

OHI introduces GraphRAG, where knowledge is modeled as a graph $G = (V, E)$. This approach enables:

- **Explicit Relations:** Kanten encode logic (IS_A, PART_OF).
- **Multi-Hop Retrieval:** Navigating A→B→C for complex claims.
- **Community Detection:** Identifying clusters (using algorithms like Leiden) to provide global summaries.

2.4 Hybrid Retrieval Strategies

OHI fuses heterogenous signals from both vector and graph spaces. We explore two primary strategies for this fusion:

2.4.1 Reciprocal Rank Fusion (RRF)

RRF is a robust method to combine result lists without score normalization:

$$\text{RRFscore}(d) = \sum_{r \in R} \frac{1}{k + \text{rank}_r(d)} \tag{2}$$

where k is a constant (typically 60) and $\text{rank}_r(d)$ is the position of document d in retriever r.

2.4.2 Linear Combination

When calibration data is available, a weighted sum often yields higher precision:

$$\text{Score}(d) = \alpha S_{\text{graph}} + \beta S_{\text{vec}} + \gamma S_{\text{lex}} \quad (3)$$

This allows graph evidence to act as a "veto" over high-similarity but factually incorrect vector matches.

3 System Architecture

The OHI architecture is designed for **Local Sovereignty**. It does not rely on external API calls to OpenAI or Anthropic for verification, preventing the "Fox Guarding the Henhouse" scenario and ensuring data privacy.

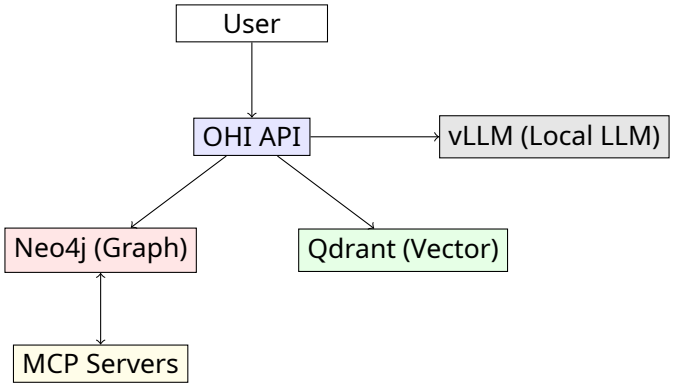

Figure 1: High-Level Architecture: The API orchestrates verification across Graph, Vector, and MCP sources using local inference.

3.1 Infrastructure Components

As defined in `docker-compose.yml`, the system comprises an isolated network of microservices:

- **vLLM (Inference Engine):** Hosts `Qwen2.5-7B-Instruct-AWQ`. vLLM's *PagedAttention* algorithm manages Key-Value (KV) cache blocks like an operating system, allowing for a 24x higher throughput than naive implementations. This is vital for the parallel nature of claim verification.
- **Neo4j (Ontology Store):** Stores the Wikipedia knowledge graph. It provides "index-free adjacency," enabling swift multi-hop traversals required for complex structural validation.
- **Qdrant (Semantic Store):** Hosts high-dimensional embeddings. Used for the initial, fuzzy retrieval of factual candidates before graph-based refinement.
- **Redis (Cache):** Provides ephemeral storage for frequent claim lookups to reduce repeated inference latency.

3.2 Verification Engine: Qwen 2.5

We employ Qwen 2.5 (7B and 32B variants) as our primary verification agent. Its excellence in generating structured JSON output is critical for stable claim decomposition. Furthermore, its support for a 128k token context window allows for the ingestion of long-form documentation as a referential context during the verify-loop.

3.3 The Model Context Protocol (MCP)

A critical innovation in OHI is the use of the **Model Context Protocol (MCP)** to standardize data access. Traditionally, integrating fragmented data sources (SQL, Knowledge Graphs, APIs) required brittle, proprietary "Connectors." MCP addresses this through a standardized client-server architecture using JSON-RPC 2.0.

3.3.1 MCP vs. LangChain

While LangChain is an excellent framework for orchestrating "how the agent thinks," MCP focuses on "what the agent can access."

Table 2: Architectural Comparison: MCP vs. LangChain

Feature	MCP	LangChain
Type	Protocol/Standard	Framework/Library
Focus	Universal Connect.	Orchestration
Security	Process Isolation	Shared Environment
Role in OHI	Ground Truth Access	Control Logic

3.3.2 The Sidecar Pattern & SSE

The `mcp_wikipedia.py` adapter functions as a sidecar service, establishing persistent connections to MCP servers via Server-Sent Events (SSE). To minimize the overhead of SSE handshakes, we implement the `MCPSessionPool`, maintaining a pool of active `sse_client` sessions. This ensures sub-millisecond query latency for repeated verification requests.

4 Algorithmic Core: The Hybrid Oracle

The heart of OHI is the `HybridVerificationOracle`, which implements a multi-strategy pattern to determine truth across structured and unstructured sources.

4.1 End-to-End Verification Pipeline

The transformation from a raw LLM output to a verified text with a trust score follows a rigorous pipeline:

1. **Input:** A user query is processed by the generative LLM (e.g., Qwen 2.5 via vLLM).
2. **Response Generation:** The model generates a candidate response R.

3. **Decomposition:** A specialized "Critic-Agent" decomposes R into an array of n atomic claims $A = \{c_1, \ldots, c_n\}$ in JSON format.

4. **Verification (Parallel):** For each claim c_i:

   - **Graph Matching:** Search Neo4j for exact path existence.
   - **Vector Retrieval:** Fetch the k most similar chunks from Qdrant. **MCP Oracle:** Query live external sources (Wikipedia/Context7) for direct evidence.

5. **Classification:** Claims are categorized as *Supported*, *Contradicted*, or *Unverifiable*.

6. **Scoring:** The *WeightedScorer* aggregates these signals into a final OHI Trust Score.

7. **Output:** The text is returned with a visual overlay (Green/Red/Gray) indicating the factuality of each sentence.

4.2 The Verification Algorithm

Algorithm 1 formalizes the scoring logic.

Algorithm 1 Hybrid Verification Protocol

$Score_{total} \leftarrow 0$
for each claim $c \in A$ **do**
  $E_{graph} \leftarrow$ Neo4j.$match(c.S, c.P, c.O)$
  $E_{vec} \leftarrow$ Qdrant.$search(embedding(c), k = 3)$
  $E_{mcp} \leftarrow$ MCP.$query(c)$
  $S_{graph} \leftarrow (1.0$ if E_{graph} else $0.0)$
  $S_{vec} \leftarrow$ CosineSimilarity(E_{vec})
  $Trust(c) \leftarrow \alpha S_{graph} + \beta S_{vec} + \gamma S_{mcp}$
  $Score_{total} \leftarrow Score_{total} + Trust(c)$
end for
return $Score_{total}/|A|$

4.3 Scoring Weights

The `WeightedScorer` applies non-linear weighting:

- **Graph Exact Match ($\alpha = 0.6$):** Represents deterministic truth.
- **Vector Semantic Match ($\beta = 0.3$):** Indicates plausibility.
- **MCP Evidence ($\gamma = 0.1$):** Provides contextual grounding.

5 Data Ingestion & Ground Truth

A verification system is only as good as its reference data. OHI utilizes a robust ETL pipeline (`import_wikipedia_to_neo4j.py`) to construct its "Ground Truth."

5.1 Dynamic Knowledge Graphs (DKG)

A static graph is insufficient in a volatile world. OHI moves towards **Dynamic Knowledge Graphs** that reflect temporal and evolving truths:

- **Temporal Dimension:** Facts have a lifespan. Edges in Neo4j are tagged with t_{valid} timestamps. Algorithms for *Temporal Knowledge Graph Completion* (TKGC) are used to predict the current validity of facts and detect temporal inconsistencies.
- **Automated Construction:** Manual curation is non-scalable. We utilize LLM-driven ingestion pipelines that extract triples from unstructured data streams (news feeds, logs) to update the graph in real-time.

5.2 Streaming XML Parsing & Ontological Mapping

Processing massive datasets like the Wikipedia XML dump (>20GB) requires memory-efficient streaming. Using `iterparse`, the system clears elements immediately after processing. During mapping, raw text is transformed into:

- **Categories:** Mapping Wiki-Markup categories to `[:IN_CATEGORY]` edges.
- **Internal Links:** Mapping hyperlinks to `[:LINKS_TO]` edges, creating a traversable semantic topology.

This structured ingestion ensures that every link in the original source becomes a logical path for the verification oracle.

5.3 Batch Processing & Resilience

To handle the scale (millions of nodes), the importer implements:

- **Batch Commit:** Transactions are grouped (e.g., 5000 nodes) to optimize Neo4j write performance.
- **Checkpointing:** The script saves the `last_page_id`. If the Docker container crashes, it resumes exactly where it left off, ensuring data integrity without full re-runs.

6 Performance & Implementation Analysis

6.1 Hardware Requirements

Running OHI in "Sovereign Mode" imposes specific hardware constraints, primarily dictated by the LLM and Knowledge Graph size.

Table 3: Infrastructure Requirements for OHI Node

Component	Requirement
LLM (Qwen2.5-7B-AWQ)	8GB VRAM (GPU)
Embedding Model	1GB VRAM / 2GB RAM
Neo4j (Graph)	4GB–16GB RAM
Qdrant (Vector)	4GB RAM (per 1M vectors)
Total System	**min. 16GB RAM + NVIDIA GPU**

6.2 Latency & Implementation Bottlenecks

The current implementation is Python-dominated, which introduces a performance ceiling. While sub-millisecond latencies are achieved for database queries (Neo4j/Qdrant), the orchestration and claim decomposition (LLM inference) remain bottlenecks.

- **Decomposition:** 200–500ms per text segment.
- **Concurrent Orchestration:** Asynchronous Python tasks handle parallel check, but CPython's GIL and I/O overhead kumulativly create delays.

Future iterations toward a production-grade system will likely involve porting the core orchestration and embedding computation to **Rust**. This would allow for native speed in parallel vector and graph queries, reducing verification latency to near real-time levels.

7 Discussion

7.1 OHI as Instrument of Governance and Compliance

With regulations like the EU AI Act, the need for verifiable accuracy is paramount. OHI transforms "vague safety" into a quantifiable KPI.

- **Auditability:** Since OHI is grounded in deterministic graphs, every trust score is traceable to specific source provenance.
- **Risk Management:** Organizations can enforce thresholds (e.g., "Discard answers with OHI < 0.8"), enabling LLM deployment in high-stakes environments like medicine or finance.

7.2 Limitation: The Entity Linking Challenge

A significant bottleneck remains the mapping of text spans to graph nodes (*Entity Linking*). Ambiguities (e.g., "Paris" referring to the city vs. the mythological figure) can cause verification collapse. We are exploring **Generative Entity Linking**, using the LLM's own latent knowledge to disambiguate context before performing the graph lookup.

7.3 The Future: LLMs as "Gardeners of the Graph"

We envision a recursive architecture where LLMs serve not only as consumers but as producers of Ground Truth. In this paradigm, "LLMs as Gardeners" extract triples from new, unverified texts to update the Neo4j ontology via MCP. OHI then monitors the same models using the structure they helped build. This creates a self-correcting feedback loop, though it raises critical questions regarding the ultimate authority of truth: *Who owns the graph?*

7.4 Epistemological Sovereignty

By relying on local models and open data dumps (Wikipedia), OHI addresses the geopolitical risk of AI. Proprietary models like GPT-4 are "black boxes" whose training data and biases are opaque. OHI allows an organization to define its own "Truth"—whether that is Wikipedia, internal corporate documentation (via `Context7`), or a scientific paper repository. The definition of truth becomes a configurable parameter of the system, not a hidden variable of the model provider.

7.5 The "Context7" Use Case: Domain Specificity

The inclusion of `mcp_context7.py` highlights a crucial capability: **Domain-Specific Verification**. While Wikipedia verifies general facts ("When was Einstein born?"), specialized domains (e.g., software engineering) require dynamic truth. A library version might change overnight. The Context7 adapter demonstrates how OHI can plug into live documentation streams.

- *Claim:* "React useMemo requires a dependency array."
- *Verification:* OHI queries the Context7 MCP server, which retrieves the latest React docs.
- *Result:* Truth is established against the *current* version of the software, not the version present in the LLM's training cutoff.

8 Future Work

8.1 Automated Graph Construction

Currently, the graph is built from explicit Wikipedia structure. Future work involves using the LLM itself to extract relationships from unstructured text during ingestion ("LLM as Knowledge Engineer"), effectively allowing the system to expand its own ontology.

8.2 Real-Time Stream Verification

Adapting the architecture to process live audio transcripts or video feeds. This would require porting the core logic from Python to Rust (using bindings for Neo4j and Qdrant) to meet real-time constraints (< 100ms latency).

9 Conclusion

The Open Hallucination Index marks the transition from the era of ”Plausible AI” to ”Verifiable AI.” By decoupling generation from verification and utilizing a sovereign architecture of Graphs, Vectors, and MCP, we provide the blueprints for an epistemic ”Trust Layer.” In a world where the cost of generating misinformation is near zero, the capability for verifiable truth becomes the most valuable asset in the digital ecosystem.

References

[1] Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). ”On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?”. *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*.

[2] Jäger, T. (2024). ”Epistemic authority and generative AI in learning spaces: rethinking knowledge in the algorithmic age”. *Frontiers in Education*.

[3] Min, S., Krishna, K., Lyu, X., Lewis, M., Yih, W., Koh, P., & Iyyer, M. (2023). ”FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation”. *EMNLP*.

[4] Pan, L., Zhang, J., Ouyang, L., & Wang, W. (2024). ”Unifying Large Language Models and Knowledge Graphs: A Roadmap”. *IEEE Transactions on Knowledge and Data Engineering*.

[5] Lewis, P., et al. (2020). ”Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks”. *NeurIPS*.

[6] Anthropic. (2024). ”Model Context Protocol (MCP) Specification”. *https://modelcontextprotocol.io*.

[7] Lettria. (2024). ”VectorRAG vs. GraphRAG: a convincing comparison”. *https://lettria.com*.

[8] Motoki, F., Neto, R. P., & Rodrigues, V. (2024). ”Is ChatGPT conservative or liberal?”. *Political Science Research and Methods*.

[9] Espejel, J., et al. (2023). ”GPT-4, GPT-3.5 and the political bias of generative AI”. *arXiv preprint arXiv:2305.14251*.

[10] Shah, S., et al. (2024). ”ReDeEP: Detecting Hallucination in Retrieval-Augmented Generation via Mechanistic Interpretability”. *arXiv preprint arXiv:2409.03571*.

[11] Vaswani, A., et al. (2017). ”Attention Is All You Need”. *NeurIPS*.

[12] Qwen Team. (2024). ”Qwen2.5: A Party of Foundation Models”. *Alibaba Cloud*.