Context Ingestion:
Trustworthy Memory Infrastructure

The evolution of retrieval systems

- **Keyword search**: lexical matching with tools such as BM25, Lucene, and Elasticsearch.
- **Vector search**: embeddings retrieve semantically related text even when wording differs.
- **Retrieval-augmented generation**: retrieved evidence gives a static LLM access to new knowledge without modifying its parameters.
- **Graph RAG and memory systems**: structure is added to connect evidence, entities, and relationships across passages.

Why RAG became the practical continual-learning pattern

- LLM weights are costly and risky to update repeatedly; retrieval provides a non-parametric way to introduce new knowledge.
- A vector index is efficient for direct evidence, but independent passage representations can miss connections that span passages or documents.
- Private enterprise documents add requirements that research prototypes can understate: provenance, access control, recovery, reproducibility, and debugging.

The limitation that motivates Graph RAG

- Real questions often require multi-hop connections: a person, event, claim, source, and qualification may live in separate text blocks.
- A graph can express relationships and support bounded expansion from an initially retrieved evidence seed.
- But a graph introduces its own risks: noisy relations, hallucinated edges, graph drift, opaque retrieval paths, and loss of source evidence.

Research gap

- The problem is not merely how to add a graph; it is how to ensure every graph-assisted result can still be defended against the original source.

Research landscape

- **Embedding-centric retrieval**: DPR, Contriever, BGE, E5, GTE, and related retrievers improve semantic candidate generation.
- **Hierarchical and summary-driven retrieval**: RAPTOR organizes documents through generated abstraction layers.
- **Graph-centric retrieval**: Microsoft GraphRAG and LightRAG explore community, entity, and relationship structures.
- **Memory-inspired retrieval**: HippoRAG and HippoRAG 2 connect retrieval to associative long-term-memory ideas.

HippoRAG: associative memory for LLM retrieval

- HippoRAG frames retrieval as a long-term-memory problem inspired by hippocampal indexing theory.
- Offline: an LLM transforms a corpus into a schemaless knowledge graph through OpenIE-style extraction.
- Online: query concepts seed Personalized PageRank (PPR), which identifies a relevant graph region and supporting passages.
- Core insight: graph connectivity can integrate knowledge across passage boundaries in one retrieval step instead of relying solely on repeated retrieval-generation cycles.

Why HippoRAG 2 was needed

- HippoRAG 2 evaluates memory more broadly: factual memory, sense-making over larger context, and associativity across multi-hop connections.
- It retains the PPR foundation but adds deeper passage integration, stronger query contextualization, and online recognition of irrelevant retrieved triples.
- The paper argues that earlier structure-augmented methods may improve some complex tasks while degrading simpler factual retrieval; robust memory needs performance across all three dimensions.

Where this project fits

- Traditional RAG emphasizes chunk retrieval. Graph RAG emphasizes structure. HippoRAG emphasizes associative memory. This project emphasizes trustworthy, production-oriented evidence infrastructure.
- The project does not claim to replace HippoRAG with a new memory algorithm.
- Instead, it supplies the discipline needed to evaluate and safely operate graph-assisted retrieval: canonical artifacts, validated evidence, controlled graph admission, durable indexing, and auditability.

Design principles

- Accountability first: every accepted semantic record, retrieval hit, and graph path must trace to source/document/block/chunk identity and, where appropriate, an exact quotation.
- Canonical artifacts first: project-owned, versioned JSON is rebuildable and authoritative.
- Derived stores second: Qdrant and Neo4j are projections that can be cleared and rebuilt from canonical inputs.
- Conservative semantics: preserve uncertainty, negation, modality, temporal qualifiers, ambiguous identities, and rejection reasons.

Pipeline

1. Docking for preprocessing, parsing and chunking
2. Spacy + GliNER for Named Entity Recognition
3. FastCoref for Coreferencing
4. GLiREL for Relation Extraction
5. Qdrant as Vector Database
6. Neo4j as graph database