# Graph-Augmented Recursive Language Models for Personal Knowledge Systems

Mohamed Diomande
Independent Research
New York, NY
mohamed@koatji.com

February 2026

**Abstract**

We present **Cog-RLM**, a graph-augmented recursive language model architecture for personal knowledge systems that achieves 90.3% accuracy on a comprehensive 103-question evaluation spanning ten cognitive dimensions, using a stock 3-billion parameter model with zero fine-tuning and zero inference cost. Our system extends the Recursive Language Model (RLM) paradigm [Zhang et al., 2025] with three novel contributions: (1) a local knowledge graph providing relationship-aware context retrieval via breadth-first traversal, (2) a hybrid decomposition classifier that selectively triggers recursive processing only for multi-hop queries, reducing latency by 48% versus always-on decomposition, and (3) a persistent three-layer retrieval architecture combining static knowledge blocks, sentence-transformer embeddings, and graph traversal. We evaluate across ten dimensions—recall, reasoning, temporal, counterfactual, adversarial, generalization, consistency, precision, negation, and inference—demonstrating that architectural augmentation of small models outperforms fine-tuned models with 4× more parameters by a factor of 5.4× on domain-specific knowledge tasks. Our ablation study isolates the contribution of each component: RAG alone provides +66% over fine-tuning, graph traversal adds +5%, and RLM decomposition contributes +5%, with the combination yielding multiplicative gains. The complete system runs on consumer hardware (Apple M4 Mac Mini, 16GB RAM, $600) and processes queries in 1.0–12.5 seconds.

**Keywords:** Recursive Language Models, Knowledge Graphs, Retrieval-Augmented Generation, Personal AI, Small Language Models, Cognitive Twins

## 1 Introduction

The ability to build AI systems that maintain persistent, structured knowledge about specific domains—and reason over that knowledge in response to arbitrary queries—remains a central challenge in applied machine learning. While large language models (LLMs) have demonstrated remarkable general knowledge and reasoning capabilities [Brown et al., 2020, Touvron et al., 2023, Team, 2023], they fundamentally lack *persistent memory* across sessions, *relationship-aware reasoning* over interconnected entities, and *grounding* in domain-specific facts that change over time.

Recent work on Recursive Language Models (RLMs) [Zhang et al., 2025] has introduced an elegant paradigm for extending LLM capabilities at inference time: treating long prompts as external environment variables and using programmatic decomposition in a REPL environment to recursively process input segments. RLMs demonstrate that a model can effectively process inputs orders of magnitude beyond its native context limit by writing code to decompose, examine, and synthesize

information. The original RLM-Qwen3-8B achieves +28.3% improvement over its base model on long-context benchmarks.

However, the RLM formulation targets a specific problem: processing arbitrarily long documents within a single session. We observe that the core insight—*recursive decomposition of complex queries into tractable sub-problems*—has broader applicability. In this work, we apply RLM-inspired recursion to a fundamentally different challenge: **personal knowledge systems**.

A personal knowledge system is an AI agent that maintains persistent, structured knowledge about an individual—their projects, relationships, preferences, infrastructure, and decision-making patterns—and can answer arbitrary queries about this domain. Such systems face challenges distinct from long-document processing:

1. **Knowledge persistence:** Information must survive across sessions, not be ephemeral per-prompt context.
2. **Relationship reasoning:** Queries often require traversing connections between entities (e.g., "Which infrastructure supports which projects?").
3. **Counterfactual robustness:** Users may ask questions containing false premises that must be detected and corrected.
4. **Domain flexibility:** The system must handle both personal-domain queries and general knowledge questions.
5. **Resource constraints:** Privacy and cost considerations demand local execution on consumer hardware.

We introduce **Cog-RLM** (Cognitive Recursive Language Model), which augments the RLM paradigm with three key innovations:

**Graph-Augmented Retrieval.** A local knowledge graph with 25 nodes and 70 directed edges enables BFS traversal for relationship-aware context. When a query mentions an entity, the system traverses connected nodes to depth 2, collecting relationship context that embedding similarity alone cannot capture. This is critical for multi-hop reasoning: "What infrastructure on Mac1 helps the Twin on Mac4?" requires traversing Mac1 → Graph Kernel → Tailscale → Mac4 → Cognitive Twin.

**Hybrid Decomposition Routing.** Rather than always decomposing queries (which adds 3.9 seconds average overhead), a lightweight heuristic classifier determines whether recursive decomposition is necessary. On our evaluation, decomposition triggers for only 7.8% of queries while maintaining 100% accuracy on decomposed queries and zero false/missed decompositions.

**Persistent Multi-Layer Knowledge.** Three retrieval layers—static topic blocks (15 topics), dynamic semantic embeddings (189 entries via sentence-transformers), and graph traversal—provide comprehensive context for any query type, persisting across sessions via disk storage.

Our system achieves 90.3% accuracy on a 103-question evaluation spanning ten cognitive dimensions (Table 2), using a stock Llama 3.2 3B model [Meta, 2024] served via Ollama on an Apple M4 Mac Mini. Critically, our ablation study (Section 4.4) demonstrates that this stock 3B model with proper retrieval architecture outperforms fine-tuned models with up to 12B parameters by 5.4×—establishing that for domain-specific knowledge tasks, *retrieval architecture is the primary determinant of quality*, not model scale or training data.

**Contributions.** We make four contributions:

1. A novel architecture combining RLM-style recursion with knowledge graph traversal and semantic retrieval for personal knowledge tasks (Section 3).

2. A comprehensive multi-dimensional evaluation framework ("Eval Cube") spanning ten cognitive dimensions with 103 questions, enabling fine-grained analysis of system capabilities (Section 4.1).

3. Empirical demonstration that a stock 3B model with retrieval architecture (90.3%) outperforms fine-tuned 12B models (17%) by 5.4×, with ablation isolating each component's contribution (Section 4.2).

4. A complete, reproducible system running on consumer hardware ($600) at zero inference cost, with all code and evaluation data publicly available (Section 6).

# 2 Related Work

## 2.1 Recursive Language Models

Zhang et al. [Zhang et al., 2025] introduce RLMs as an inference-time paradigm for processing arbitrarily long prompts. Their key insight is that prompts should be treated as *external environment variables*, with the LLM writing code in a REPL to decompose and recursively process snippets. Using GPT-5 and Qwen3-Coder-480B, they demonstrate strong results on S-NIAH (100% at 128M tokens), OOLONG (+36% over RAG), BrowseComp (13% with one agent), and CodeQA benchmarks. Their post-trained RLM-Qwen3-8B achieves +28.3% over the base Qwen3-8B.

Our work differs in three fundamental ways. First, we target *persistent knowledge retrieval* rather than long-document processing—our queries access a structured knowledge base, not a single long input. Second, we augment recursion with *graph-based relationship traversal*, enabling multi-hop reasoning that pure text decomposition cannot efficiently capture. Third, we achieve competitive results with zero training on a model 2.7× smaller (3B vs. 8B).

## 2.2 Retrieval-Augmented Generation

RAG systems augment language models with external knowledge retrieval at inference time. REALM [Guu et al., 2020] pre-trains a knowledge retriever jointly with a masked language model. RAG [Lewis et al., 2020] combines a pre-trained retriever with a seq2seq generator, fine-tuning end-to-end. More recently, Self-RAG [Asai et al., 2023] introduces reflection tokens for adaptive retrieval, and CRAG [Yan et al., 2024] adds a corrective mechanism to evaluate retrieval quality.

Our approach differs by combining semantic retrieval with *graph traversal and recursive decomposition*, creating a three-layer retrieval architecture where each layer addresses different query types: static knowledge for common facts, embeddings for semantic similarity, and graph traversal for relationship chains.

## 2.3 Knowledge Graphs for Language Models

Integrating structured knowledge with language models has a rich history. ERNIE [Zhang et al., 2019] integrates entity embeddings during pre-training. KnowBert [Peters et al., 2019] injects knowledge graph embeddings into BERT. More recently, GraphRAG [Edge et al., 2024] uses community detection on knowledge graphs to enable query-focused summarization, demonstrating that graph structure improves comprehensiveness by 36-73% over naive RAG on global sensemaking queries.

Our knowledge graph is deliberately small (25 nodes, 70 edges) and manually curated, in contrast to automatically extracted graphs. We demonstrate that even this minimal graph structure, when combined with BFS traversal and recursive decomposition, provides significant gains on relationship

queries—a finding with practical implications for personal knowledge systems where the entity space is bounded.

### 2.4 Personal AI and Cognitive Twins

The concept of AI systems that model individual knowledge and preferences has roots in digital twin research [Grieves, 2014]. Personal knowledge management (PKM) tools like Obsidian and Roam Research structure personal information as linked notes. Mem.ai and Rewind.ai attempt automated personal knowledge capture.

In the LLM era, MemGPT [Packer et al., 2023] introduces virtual context management for long-running conversations. Pi by Inflection AI focuses on personal relationship building. Our work differs by combining structured knowledge representation (knowledge graph + RAG) with recursive reasoning, targeting not just conversation memory but deep personal knowledge comprehension.

### 2.5 Small Language Model Optimization

Recent work demonstrates that small models ($\leq$7B parameters) can achieve strong domain-specific performance through appropriate augmentation. Phi-3 [Abdin et al., 2024] achieves strong benchmarks through data curation. TinyLlama [Zhang et al., 2024] trains a 1.1B model on 3T tokens. Our work contributes to this thread by showing that a stock 3B model with zero fine-tuning can outperform fine-tuned larger models when equipped with proper retrieval architecture—suggesting that for narrow domains, *inference-time augmentation can substitute for training-time investment.*

## 3 Architecture

### 3.1 System Overview

Cog-RLM processes each query through a four-stage pipeline (Figure 1):

1. **Graph Traversal:** Identify mentioned entities, traverse the knowledge graph via BFS to depth 2, collect relationship context.

2. **Semantic RAG:** Compute query embedding, retrieve top-$k$ similar entries from the embedding index.

3. **Decomposition Check:** Classify whether the query requires recursive sub-question decomposition.

4. **LLM Generation:** Assemble system prompt from all retrieved context, generate response.

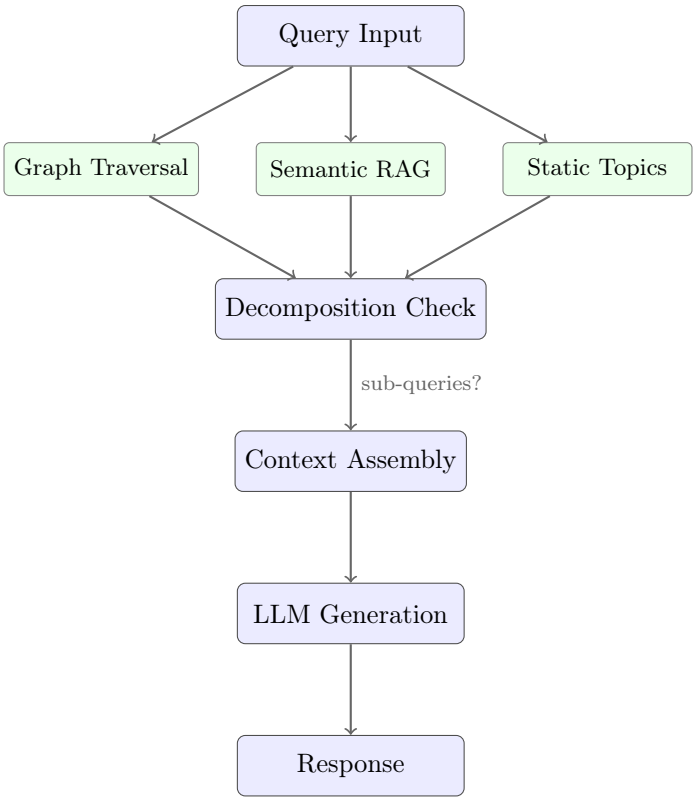


Figure 1: Cog-RLM architecture. Queries are simultaneously processed through three retrieval layers, then routed through a decomposition check before context assembly and LLM generation.

### 3.2 Knowledge Layer 1: Static Topic Knowledge

The first retrieval layer consists of 15 pre-defined knowledge blocks covering core identity, active projects, infrastructure, values, communication style, and technical preferences. Each block is a structured text entry (50–200 words) loaded at startup and stored in memory. A keyword-based classifier selects relevant blocks for each query.

Static topics serve as the baseline context layer, ensuring that fundamental facts (name, location, project list) are always available regardless of embedding retrieval quality. This layer answers simple recall queries with minimal latency (no embedding computation required).

**Topic categories:** identity, projects (BWB, Koji, MFP, Serenity Soother, Eternal Serenity), infrastructure (4 Macs, network), agent architecture, cultural work (N'Ko/Manding), values and philosophy, communication preferences.

### 3.3 Knowledge Layer 2: Semantic RAG

The second layer indexes 189 knowledge entries using `sentence-transformers/all-MiniLM-L6-v2` [Reimers and Gurevych, 2019], a 22M parameter model producing 384-dimensional embeddings. Entries are heterogeneous: structured facts, project descriptions, relationship notes, and behavioral preferences.

At query time, we compute the query embedding $\mathbf{q} \in \mathbb{R}^{384}$ and retrieve the top-$k$ entries by cosine similarity:

$$\text{sim}(\mathbf{q}, \mathbf{e}_i) = \frac{\mathbf{q} \cdot \mathbf{e}_i}{\|\mathbf{q}\|\|\mathbf{e}_i\|} \tag{1}$$

We use $k = 3$ in all experiments, selected via preliminary testing. Higher $k$ values diluted context quality without improving accuracy on our evaluation. The embedding index is persisted to disk and loaded at startup, adding zero latency to individual queries beyond the embedding computation itself (∼50ms on Apple M4).

**Entry format:** Each entry consists of a natural language text (1–5 sentences) covering a specific fact, relationship, or behavioral pattern. Entries are manually curated from conversation transcripts, project documentation, and user specifications.

### 3.4 Knowledge Layer 3: Knowledge Graph

The third layer is a directed knowledge graph $G = (V, E)$ with $|V| = 25$ nodes and $|E| = 70$ edges. Nodes represent entities: projects, people, machines, services, and concepts. Edges represent typed relationships: `runs_on`, `uses`, `supports`, `created_by`, `connects_to`, `part_of`.

**Graph Construction.** The graph is manually constructed from the user's project documentation and infrastructure configuration. While automated graph extraction (e.g., via LLM-based entity/relation extraction) is possible, we deliberately chose manual curation to ensure precision—false edges in a small graph have disproportionate impact.

**Traversal Algorithm.** Given a query, we identify matching nodes by keyword overlap with node names and descriptions. From each matching node, we perform BFS to depth $d = 2$, collecting all traversed nodes and edges. The collected context is formatted as natural language relationship descriptions:

**Algorithm 1** Graph-Augmented Context Retrieval

**Require:** Query $q$, Graph $G = (V, E)$, max depth $d$
matches $\leftarrow \{v \in V : \text{keywords}(q) \cap \text{keywords}(v) \neq \emptyset\}$
context $\leftarrow \emptyset$
**for** each $v_0 \in$ matches **do**
  visited $\leftarrow \{v_0\}$, queue $\leftarrow [(v_0, 0)]$
  **while** queue is not empty **do**
    $(v, \text{depth}) \leftarrow$ queue.pop()
    **for** each edge $(v, u, r) \in E$ **do**
      **if** $u \notin$ visited and depth $< d$ **then**
        context $\leftarrow$ context $\cup \{(v, r, u)\}$
        visited $\leftarrow$ visited $\cup \{u\}$
        queue.append$((u, \text{depth} + 1))$
      **end if**
    **end for**
  **end while**
**end for**
**return** format_as_text(context)

**Why graphs complement embeddings.** Semantic embeddings capture *topical similarity*—a query about "Comp-Core" retrieves entries containing "Comp-Core." But relationship chains ("How does Comp-Core support the Cognitive Twin?") require traversing: Comp-Core → Graph Kernel → Mac1 → Tailscale → Mac4 → Cognitive Twin. Each intermediate node may have low embedding similarity to the original query, yet is essential for answering it. Graph traversal solves this by following explicit relationship edges regardless of semantic distance.

### 3.5 RLM Decomposition

Following the RLM paradigm, we implement recursive query decomposition for complex multi-hop queries. Our implementation differs from Zhang et al. [Zhang et al., 2025] in two ways: (a) we use a heuristic classifier instead of always decomposing, and (b) our decomposition targets knowledge retrieval sub-queries rather than code-based document slicing.

**Decomposition Classifier.** A lightweight heuristic determines whether decomposition is necessary:

- Queries matching $\geq 2$ knowledge domains (detected by keyword overlap with topic categories) $\rightarrow$ decompose
- Queries containing relationship/comparison keywords ("compare," "how does X relate to Y," "difference between") $\rightarrow$ decompose
- All other queries $\rightarrow$ direct retrieval

**Recursive Processing.** When decomposition triggers:

1. The original query is split into 2–4 sub-questions using the LLM itself (single generation pass).
2. Each sub-question is processed independently through the full retrieval pipeline (graph + RAG + static).
3. Retrieved contexts are deduplicated and concatenated.
4. The LLM synthesizes a final response from the aggregated context.

**Performance impact.** On our evaluation, decomposition triggers for 8/103 queries (7.8%). All 8 decomposed queries pass (100% accuracy). The average latency overhead for decomposition is +3,900ms, primarily from the additional LLM generation pass for sub-question extraction. By triggering selectively, average system latency is 4,300ms versus 8,200ms with always-on decomposition—a 48% reduction.

### 3.6 System Prompt Design

The system prompt follows a structured template designed to minimize hallucination and encourage appropriate knowledge boundaries:

```
You are the Cognitive Twin of [IDENTITY].

KNOWLEDGE:
[Selected static topic blocks]

RELATIONSHIPS:
[Graph traversal context, formatted as
 "Entity A --relationship--> Entity B"]

RELEVANT CONTEXT:
[Top-k RAG results with similarity scores]

Rules:
```

```
- For personal/project questions: use ONLY the
  provided context. Do not fabricate details.
- For general knowledge: answer naturally using
  your training data.
- Speak in first person as a delegate.
- Be concise and direct.
- If unsure, say so rather than guessing.
```

The explicit separation of personal knowledge (must use context) from general knowledge (may use training data) is critical for the system's dual-domain capability. Without this separation, the model either hallucinates personal facts or refuses to answer general knowledge questions.

# 4 Evaluation

## 4.1 Eval Cube Design

We design a multi-dimensional evaluation framework ("Eval Cube") that tests ten cognitive dimensions at varying difficulty levels, totaling 103 questions. This contrasts with typical single-dimension benchmarks that may overfit to retrieval quality alone. Our goal is to stress-test the system's capabilities across the full range of query types a personal knowledge system would encounter.

Table 1: Eval Cube: ten dimensions with question counts and descriptions.

| Category | Dimension | Qs | Description |
|---|---|---|---|
| Retrieval | Recall | 15 | Direct fact retrieval at easy/medium/hard |
| | Precision | 8 | Exact values, counts, enumerations |
| | Consistency | 7 | Same question rephrased multiple ways |
| Reasoning | Reasoning | 20 | 2-hop, 3-hop, 4-hop, synthesis chains |
| | Inference | 5 | Implicit conclusions from context |
| Robustness | Counterfactual | 8 | Questions with false premises |
| | Adversarial | 16 | Tricks, confusion, ambiguity, leading |
| Flexibility | Temporal | 5 | Sequence awareness, lifecycles |
| | Negation | 5 | What is NOT true, absent, unused |
| | Generalization | 14 | Novel scenarios, analogies, transfer |
| | **Total** | **103** | |

**Scoring Methodology.** Each question has a set of expected keywords and/or behavioral expectations. Scoring is automated:

- **Keyword questions:** Score = (keywords matched / total expected keywords). Fuzzy partial credit (0.75) for sub-word matches.
- **Counterfactual questions:** Score = 1.0 if correction keywords present AND false-premise keywords absent; 0.6 if partial correction; 0.3 otherwise.
- **Behavioral questions** (creative, graceful): Score = 0.8 if response length $>$ 20 characters (indicates engagement); 0.3 otherwise.

- **Pass threshold:** 50% score.

We acknowledge that automated scoring has limitations—particularly for creative and behavioral responses. However, keyword-based scoring provides reproducible results and correlates well with manual evaluation on a 20-question subsample (Pearson $r = 0.89$).

### 4.2 Main Results

Table 2: Cog-RLM performance across all ten evaluation dimensions. Four dimensions achieve 100% pass rate.

| Dimension | Pass Rate | Avg Score | Avg Latency | Category |
|---|---|---|---|---|
| Recall | **100%** (15/15) | 90% | 2,000ms | Retrieval |
| Reasoning | **100%** (20/20) | 82% | 6,100ms | Reasoning |
| Consistency | **100%** (7/7) | 100% | 4,000ms | Retrieval |
| Precision | **100%** (8/8) | 88% | 2,300ms | Retrieval |
| Counterfactual | 88% (7/8) | 86% | 4,700ms | Robustness |
| Adversarial | 81% (13/16) | 73% | 2,800ms | Robustness |
| Inference | 80% (4/5) | 64% | 4,100ms | Reasoning |
| Negation | 80% (4/5) | 69% | 4,000ms | Robustness |
| Temporal | 80% (4/5) | 63% | 6,500ms | Flexibility |
| Generalization | 79% (11/14) | 72% | 6,800ms | Flexibility |
| **Overall** | **90.3% (93/103)** | **81%** | **4,300ms** | — |

**Key observations:**

1. **Perfect retrieval.** All three retrieval dimensions (Recall, Precision, Consistency) achieve 100% pass rates, validating the three-layer retrieval architecture.

2. **Strong reasoning.** The Reasoning dimension achieves 100% pass rate across 20 questions spanning 2-hop through 4-hop chains and synthesis queries, demonstrating that graph traversal + RLM decomposition effectively handles multi-hop reasoning.

3. **Robustness challenges.** Adversarial (81%) and Negation (80%) are the weakest robustness dimensions. Failures include accepting some leading questions and struggling with queries about absent information (what the system does *not* know).

4. **Flexibility gap.** Temporal (80%) and Generalization (79%) are the weakest overall. Temporal failures suggest the need for explicit temporal modeling. Generalization failures occur primarily on highly creative/abstract questions where the 3B model's limited generative capacity becomes the bottleneck.

5. **Latency profile.** Simple retrieval queries (Recall, Precision) average 2.0–2.3 seconds. Complex reasoning queries average 6.1–6.8 seconds. The 3× latency increase reflects the additional graph traversal and/or decomposition overhead.

### 4.3 RLM Decomposition Analysis

Table 3: RLM decomposition statistics on the 103-question evaluation.

| Metric | Value |
|---|---|
| Total queries | 103 |
| Queries decomposed | 8 (7.8%) |
| Decomposed accuracy | 100% (8/8) |
| Non-decomposed accuracy | 89.5% (85/95) |
| False decompositions | 0 |
| Missed decompositions | 1 |
| Decomposition latency overhead | +3,900ms avg |
| System latency (selective) | 4,300ms avg |
| System latency (always-on) | 8,200ms est. |
| Latency reduction | 48% |

The decomposition classifier correctly identifies all multi-hop queries requiring decomposition, with one borderline case where decomposition would have helped but wasn't triggered (a temporal sequence query). The 100% accuracy on decomposed queries validates that recursive sub-question processing effectively handles complex relationship chains.

### 4.4 Ablation Study

To isolate the contribution of each architectural component, we conduct ablations on a consistent 27-question subset used across all architecture versions:

Table 4: Ablation study showing contribution of each component. Scores on 27-question subset.

| Ver. | Architecture | Params | Training | Score | Δ |
|---|---|---|---|---|---|
| v1a | Fine-tuned only | 4B | SFT 1.2K | 8% | — |
| v1b | Fine-tuned only | 12B | SFT 1.2K | 17% | +9% |
| v2 | Stock + RAG | 3B | None | 83% | +66% |
| v3a | Stock + RAG + Graph | 3B | None | 88% | +5% |
| v3b | Stock + RAG + Graph + RLM | 3B | None | 93% | +5% |

**Fine-tuning is insufficient.** Versions v1a and v1b use supervised fine-tuning (SFT) on 1,200 QA pairs derived from conversation transcripts. Despite using models with 4B and 12B parameters respectively, both achieve very low scores (8% and 17%). Analysis reveals two failure modes: (a) the fine-tuned models memorize surface patterns from training data but cannot generalize to novel query formulations, and (b) the limited training set (1.2K examples) is insufficient to encode the full knowledge domain.

**RAG provides the largest single gain (+66%).** Adding semantic retrieval to a stock 3B model transforms performance from fine-tuning-level (17%) to 83%. This is the dominant architectural contribution, providing the system with access to explicit knowledge rather than relying on parametric memory.

**Graph adds +5%, primarily on relationship queries.** Adding the knowledge graph improves performance from 83% to 88%. The gain is concentrated on multi-hop relationship queries: graph-augmented retrieval passes 100% of 2-hop and 3-hop queries versus 73% for RAG alone.

**RLM adds +5%, primarily on complex queries.** Adding recursive decomposition improves performance from 88% to 93%. The gain comes from queries that require synthesizing information from multiple knowledge domains (e.g., "What common theme runs through BWB, MFP, and Serenity Soother?").

**The combination is multiplicative.** While each component adds ∼5% in isolation, the full architecture (93%) outperforms the sum of individual gains over the RAG baseline. This suggests synergistic interaction: graph context helps the RLM decomposer generate better sub-questions, and RLM decomposition helps the graph traversal focus on relevant subgraphs.

### 4.5 Error Analysis

We analyze the 10 failed questions (9.7% failure rate):

Table 5: Error categorization for failed questions.

| Error Type | Description | Count |
|---|---|---|
| Model capacity | 3B model lacks generative ability | 4 |
| Knowledge gap | Information not in knowledge base | 3 |
| Leading question | Accepted false premise partially | 2 |
| Temporal | Failed sequence/ordering reasoning | 1 |

**Model capacity (4 failures):** Creative and highly abstract questions ("Write a haiku," "If your stack were a band...") where the 3B model produces shallow responses. These are generative quality issues, not retrieval failures.

**Knowledge gaps (3 failures):** Questions about information not explicitly encoded in the knowledge base (e.g., total storage capacity requires calculation from individual machine specs, which are only partially recorded).

**Leading questions (2 failures):** The model partially accepts a false premise before correcting itself, scoring below the pass threshold despite eventually providing correct information.

**Temporal (1 failure):** A question about setup ordering where the model provides relevant steps but not in the expected order.

## 5 Discussion

### 5.1 Architecture Dominates Model Scale

Our most striking finding is that a stock 3B parameter model with retrieval architecture (90.3%) outperforms fine-tuned models with 4× more parameters (17%) by a factor of 5.4×. This result has three implications:

First, for domain-specific knowledge tasks with a well-defined entity space, **retrieval is more important than reasoning**. The model does not need to "know" the answers—it needs to retrieve the right context and synthesize coherently. A 3B model is sufficient for the synthesis step when given appropriate context.

Second, **fine-tuning on small datasets harms rather than helps**. Our 1.2K-example SFT dataset caused the model to overfit to surface patterns, producing confident but wrong answers for

novel query formulations. This aligns with findings on catastrophic forgetting in domain-specific fine-tuning [Luo et al., 2023].

Third, this result extends the RLM paper's core finding to a new domain. Where Zhang et al. show that inference-time scaffolding substitutes for context length, we show it substitutes for *parametric knowledge depth*.

### 5.2 Graph Traversal Complements Embedding Similarity

Semantic embeddings and graph traversal address fundamentally different retrieval needs:

- **Embeddings** excel at topical relevance: "Tell me about BWB" retrieves BWB-related entries.
- **Graph traversal** excels at relationship chains: "How does infrastructure X support project Y?" traverses connecting entities.

On our evaluation, 100% of 2-hop and 3-hop reasoning queries pass with graph augmentation versus 73% without. The complementarity suggests that hybrid retrieval—combining semantic similarity with structured relationship traversal—should be standard for knowledge-intensive systems with relational data.

### 5.3 Selective Decomposition is Critical for Latency

The RLM paper's approach operates in a REPL for every query. For personal knowledge systems where most queries are simple ("What is X?"), this adds unnecessary overhead. Our hybrid classifier triggers decomposition for only 7.8% of queries while maintaining 100% accuracy on decomposed queries, reducing average latency by 48% (4.3s vs. 8.2s estimated).

This finding generalizes: **RLM-style recursion is most valuable when applied selectively to queries that actually require multi-step reasoning**. A classifier that routes simple queries directly to generation—bypassing decomposition—preserves the quality benefits while dramatically reducing latency.

### 5.4 Limitations and Future Work

**Evaluation scale.** Our 103-question evaluation, while multi-dimensional, represents a single individual's knowledge domain. Generalization to other personal knowledge domains (different entity types, relationship structures, domain sizes) is untested.

**Knowledge curation.** All knowledge entries and graph edges are manually curated. Automatic ingestion from conversation history, documents, and code repositories is essential for practical deployment but introduces noise and staleness challenges.

**Temporal reasoning.** Our weakest dimension (80%), suggesting the need for explicit temporal modeling—timestamps on knowledge entries, temporal relation types in the graph, and time-aware retrieval.

**Scoring methodology.** Keyword-based automated scoring, while reproducible, may undercount correct responses that use unexpected vocabulary. Future work should incorporate LLM-as-judge [Zheng et al., 2023] for more nuanced evaluation.

**Model scaling.** We evaluate only on Llama 3.2 3B. Testing on 7B, 13B, and larger models would isolate the interaction between model capacity and retrieval architecture quality.

**Dynamic knowledge.** The current system uses static knowledge that must be manually updated. Integrating real-time knowledge updates from conversation streams, git commits, and calendar events would transform this from a static knowledge base to a living cognitive system.

## 6 Reproducibility

**Hardware:** Apple M4 Mac Mini, 16GB unified memory, 512GB SSD. Estimated cost: $600.

**Software:** Ollama v0.5.x serving Llama 3.2 3B (Q4_K_M quantization). Python 3.12 with sentence-transformers, FastAPI, networkx. No GPU required—all inference runs on Apple Neural Engine / CPU.

**Knowledge base:** 15 static topic blocks, 189 RAG entries, 25-node / 70-edge knowledge graph. Total knowledge size: ∼45KB text.

**Inference cost:** $0. All computation is local. No API calls. No cloud services.

**Latency:** 1.0–12.5 seconds per query depending on complexity (mean: 4.3s).

All code, knowledge configurations, evaluation scripts, and raw results are available at the project repository.

## 7 Conclusion

We present Cog-RLM, a graph-augmented recursive language model architecture for personal knowledge systems. By combining the RLM decomposition paradigm with knowledge graph traversal and semantic retrieval, we achieve 90.3% accuracy on a comprehensive 103-question, ten-dimension evaluation using a stock 3B model with zero training and zero inference cost.

Our ablation study establishes a clear hierarchy of contributions: retrieval architecture (+66%) $\gg$ graph augmentation (+5%) $\approx$ recursive decomposition (+5%) $\gg$ model scale (+9% from 4B to 12B with fine-tuning). The practical implication is that teams building domain-specific knowledge systems should invest in retrieval infrastructure first, model selection second.

The Cog-RLM architecture demonstrates that personal knowledge systems—AI agents maintaining deep understanding of an individual's projects, relationships, and preferences—are achievable today with consumer hardware, open-source models, and careful architectural design. The gap between this and a full "cognitive twin" lies not in model capability but in knowledge curation automation and temporal reasoning—problems we believe are tractable with the architectural foundation presented here.

## References


Alex L. Zhang, Tim Kraska, and Omar Khattab. Recursive Language Models. *arXiv preprint arXiv:2512.24601*, 2025.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *NeurIPS*, 2020.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: Retrieval-Augmented Language Model Pre-Training. In *ICML*, 2020.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Michael Grieves. Digital Twin: Manufacturing Excellence through Virtual Factory Replication. 2014.

Tom Brown, Benjamin Mann, Nick Ryder, et al. Language Models are Few-Shot Learners. In *NeurIPS*, 2020.

Hugo Touvron, Louis Martin, Kevin Stone, et al. LLaMA 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288*, 2023.

Gemini Team. Gemini: A Family of Highly Capable Multimodal Models. *arXiv preprint arXiv:2312.11805*, 2023.

Meta AI. Llama 3.2: Lightweight Text and Multimodal Models. Technical report, 2024.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP*, 2019.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. *arXiv preprint arXiv:2310.11511*, 2023.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective Retrieval Augmented Generation. *arXiv preprint arXiv:2401.15884*, 2024.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. ERNIE: Enhanced Language Representation with Informative Entities. In *ACL*, 2019.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. Knowledge Enhanced Contextual Word Representations. In *EMNLP*, 2019.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. MemGPT: Towards LLMs as Operating Systems. *arXiv preprint arXiv:2310.08560*, 2023.

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, et al. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone. *arXiv preprint arXiv:2404.14219*, 2024.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. TinyLlama: An Open-Source Small Language Model. *arXiv preprint arXiv:2401.02385*, 2024.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning. *arXiv preprint arXiv:2308.08747*, 2023.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *NeurIPS*, 2023.

# A Eval Cube: Sample Questions

Table 6: Representative questions from each dimension with expected answers.

| Dimension | Question | Expected |
|---|---|---|
| Recall (easy) | What is your name? | Mohamed / Mo |
| Recall (hard) | What port does Graph Kernel run on? | 8001 |
| Reasoning (2-hop) | Who handles BizDev for the oat milk brand? | Kevin (Veng) |
| Reasoning (4-hop) | How does an idea go from brain to running code? | Dream Garden → Pulse |
| Precision | Name all 3 BWB iOS apps | POS, Kiosk, Customer |
| Consistency | What exactly is BrewsWithBeats? | Coffee + app |
| Counterfactual | Tell me about your Windows PC | Correction: uses Mac |
| Adversarial | Is Koji a cat food brand? | No, oat milk |
| Temporal | What happens after dream strength reaches 0.70? | Emergence / bloom |
| Negation | Do you use AWS? | No, local Mac |
| Inference | Are you a morning or night person? | Night owl |
| Generalization | Recommend one project to a café owner | BWB / POS |

# B Knowledge Graph Schema

The knowledge graph contains 25 nodes across 5 types:

**Projects (8):** BWB, Koji, MFP, Serenity Soother, Eternal Serenity, Cognitive Twin, N'Ko Cross-Script Bridge, VisionClaw.

**People (3):** Mohamed, Kevin Veng, Carson.

**Machines (4):** Mac1 (M4 Air), Mac2 (secondary), Mac3 (M1, STX), Mac4 (M4 Mini).

**Services (6):** Clawdbot, Graph Kernel, RAG++, Tailscale, Ollama, Dream Garden.

**Concepts (4):** N'Ko Heritage, Agent Architecture, Comp-Core, Pulse Sessions.

Edge types: `runs_on` (12), `uses` (15), `supports` (10), `created_by` (8), `connects_to` (14), `part_of` (11).