NeuralGraph

A context engine that stores knowledge as interconnected graphs and retrieves the right context at the right time. Not a chatbot framework. Not a RAG pipeline. A structured memory layer for AI.

◈ Graph Memory

Most AI memory systems store flat chunks of text and retrieve them with vector similarity. That works for simple recall, but it loses structure. **NeuralGraph stores knowledge as a graph** — nodes represent discrete pieces of knowledge, edges represent relationships between them.

A node can be anything: a fact, a person, a preference, a concept, an event. Each node has a type, importance score, and structured data. Edges carry meaning — **RELATED_TO**, **CONTRADICTS**, **SUBTOPIC_OF**, **SUPERSEDES** — so the system understands not just what it knows, but how things connect.

```
A conversation about a user's work life produces:

[David] --WORKS_AT--> [Acme Corp]
   |                          |
   MANAGES                SUBTOPIC_OF
   |                          |
[Team Alpha]             [Tech Industry]
   |
   RELATED_TO
   |
[Q3 Launch] --CAUSES--> [Stress]
```

When new information arrives, the graph updates. If David changes jobs, the old **WORKS_AT** edge gets superseded — not deleted. The graph maintains history while keeping current state clear.

⚡ Triggers

This is what makes NeuralGraph different. **Triggers are semantic hooks** — short phrases extracted alongside each node that describe when that knowledge should surface. They're not keywords. They're intent signals.

When a user says "I'm stressed about the launch," the system doesn't just vector-search for similar text. It scans for triggers. The node **[Q3 Launch]** might have triggers like "launch timeline", "project deadline", "work pressure". The node **[Stress]** might have "feeling overwhelmed", "anxiety". Both fire. Both contribute context.

```
Each node has triggers that control when it surfaces:

[Q3 Launch]
   trigger: "launch timeline"    strength: 1.4
   trigger: "project deadline"   strength: 1.1
   trigger: "work pressure"      strength: 0.9
   trigger: "release date"       strength: 0.7

Triggers strengthen with positive feedback (+0.1, cap 2.0)
and weaken when irrelevant (−0.05, floor 0.1)
```

Trigger strength is adaptive. When the AI uses a piece of context and it's relevant, the triggers that surfaced it get stronger. When context is irrelevant, those triggers weaken. **The graph learns what matters over time** — no retraining, no manual tuning.

Context Spaces

Knowledge lives in **spaces** — isolated graphs with their own schema, ingestion rules, and retrieval config. A space can be anything: a personal memory store, a domain knowledge base, a team wiki, a product catalog.

Each space has a **space type** defined in YAML that controls everything: what node types exist, what edges are allowed, how text is extracted into nodes, how nodes are scored during retrieval, and how knowledge decays over time.

```
Space: "memories"        Space: "research"        Space: "personality"
type: personal_memory     type: knowledge_base     type: ai_personality

┌──────────────┐         ┌──────────────┐         ┌──────────────┐
│ [Max the dog] │        │ [Quantum ML]  │        │ [Tone: warm]  │
│ [Thai food]   │        │ [RLHF paper]  │        │ [Curious]     │
│ [Q3 Launch]   │        │ [Attention]   │        │ [Concise]     │
└──────────────┘         └──────────────┘         └──────────────┘
```

Spaces are **isolated by default, composable at query time**. A single hydration request can search across any combination of spaces — merging a scientist's research graph with their personal memory graph and an AI personality graph in one query. The results are scored and ranked together.

Multi-Space Hydration

At query time, you pass a list of **space_ids** and NeuralGraph searches across all of them simultaneously. Each space contributes nodes, and they're scored together in a single ranked list.

Imagine a research assistant that knows both your personal context and a domain knowledge base. When you ask "What papers should I read next?", NeuralGraph pulls from your **personality space** (you prefer practical over theoretical), your **memory space** (you mentioned interest in reinforcement learning last week), and a **research space** (recent RLHF papers). All composed dynamically, no upfront merging needed.

```
Hydration request:
user_id:   "david"
space_ids: ["memories", "research", "personality"]
messages:  "What papers should I read next?"

Result: nodes from all three spaces, ranked together

[RLHF Survey 2025]      research      score: 0.91
[Interested in RL]       memories      score: 0.87
[Prefers practical]      personality   score: 0.72
[Attention paper]        research      score: 0.65
[PhD in CS]              memories      score: 0.41
```

The system prompt is built from the top-scoring nodes across all spaces. The AI gets exactly the context it needs — personal knowledge, domain knowledge, and behavioral guidelines — all in one call.

🔍 Retrieval Channels

NeuralGraph doesn't rely on a single retrieval method. It runs **three concurrent channels** and merges the results:

CHANNEL	HOW IT WORKS	WHAT IT CATCHES
Trigger Matching	Scans input for trigger phrases, weighted by strength	Contextually relevant nodes based on conversational patterns
Vector Search	Embeds the input and finds semantically similar nodes	Nodes with similar meaning that triggers might miss
Graph Expansion	Traverses edges from matched nodes to find connected knowledge	Related context that wouldn't match on text alone

All three channels run without an LLM call. Retrieval is pure computation — trigger matching, embedding lookup, graph traversal. **Context retrieval takes milliseconds, not seconds.** The only LLM call is the final response generation, which uses the context NeuralGraph assembled.

Scoring: Each node's final score is a composite of trigger strength, match count, importance, recency, and coverage. Recently mentioned topics get a short-term boost (configurable, default 4 hours). Scoring weights are fully configurable per space type.

Memory Lifecycle

Knowledge isn't static. NeuralGraph manages the full lifecycle of every node:

1. **Extraction** — conversations and content are ingested, an LLM extracts structured nodes, triggers, and edges into the graph
2. **Reinforcement** — when a node surfaces and is useful, its triggers strengthen and importance increases
3. **Decay** — nodes that haven't been accessed gradually lose importance. Configurable decay curves per space type
4. **Consolidation** — related nodes can be merged or summarized over time, keeping the graph dense and relevant
5. **Deletion** — nodes can be soft-deleted, removing them from retrieval while preserving graph structure

The result is a graph that **naturally reflects what matters now**. Recent, frequently-referenced knowledge rises to the top. Old, unused knowledge fades. No manual curation needed.

🎯 Relevance Feedback

After every response, the AI rates which context was relevant and which wasn't. These ratings flow back to NeuralGraph as **relevance feedback**, adjusting trigger strengths in real time.

```
User: "How's the weather?"
Hydration surfaces: [Weather preferences], [Q3 Launch], [Stress]

AI rates context:
[Weather preferences]  relevant      triggers +0.1
[Q3 Launch]             not_relevant  triggers -0.05
[Stress]                not_relevant  triggers -0.05

Next time someone asks about weather,
[Q3 Launch] is less likely to surface.
```

No retraining. No embeddings recomputed. Just edge weight adjustments in the graph. The system gets more precise with every interaction.

⚖ Compared to Other Approaches

APPROACH	RETRIEVAL	STRUCTURE	ADAPTS OVER TIME	MULTI-DOMAIN
RAG (vector only)	Semantic similarity	Flat text chunks	No	No
Entity Knowledge Graphs	Entity lookup	Entity-relation triples	No	No
Memory-augmented LLMs	Conversation window	Flat message history	Recency only	No
Graph Memory Systems	Graph relationships	Memory graph with updates/extends	Partial (rewrites)	No
NeuralGraph	Triggers + vectors + graph (concurrent)	Typed nodes, weighted edges, configurable schemas	Adaptive triggers, feedback, decay	Multi-space composition

Most existing approaches solve one piece of the problem. Vector RAG handles semantic recall but loses structure. Knowledge graphs preserve structure but can't adapt. Conversation windows are simple but forget everything outside the window. **NeuralGraph combines structured storage, multi-channel retrieval, and continuous adaptation into a single platform** — and lets applications compose across independent knowledge domains at query time.

⚙ Architecture

NeuralGraph is a single Go binary backed by four stores:

STORE	TECHNOLOGY	PURPOSE
Content	PostgreSQL	Nodes, profiles, jobs, space configs
Graph	Memgraph	Edges, triggers, graph traversal
Vector	Qdrant	Embeddings for semantic search
Cache	Redis	Sessions, rate limiting, hot data

Embeddings run locally via HuggingFace TEI — no external API calls for vector operations. The API is stateless and horizontally scalable. Multi-tenancy is built in at every layer. Space types are defined in YAML — new domains can be added without code changes.

Deployed on GKE Autopilot. Total infrastructure cost: **~$95/month** for API, databases, embeddings, and vector search.