From the AI Jungle to RAG in a Tab

Private AI Search by https://soverius.ai

Angular | WebLLM | PGlite | pgvector | WebGPU

The Solution

RAG in the Browser

Retrieval-Augmented Generation with a Small Language Model,
running entirely on the user’s device.

Privacy — The Leak Problem

Mar 2026	Meta AI agent goes rogue, exposes data in Sev 1 breach
Feb 2026	AI chat app leak exposes 300M messages tied to 25M users
Nov 2025	OpenAI data breach exposed API user information
Aug 2025	Thousands of Grok chats are now searchable on Google
Jan 2025	Meta AI was leaking chatbot prompts and answers

Your data is only as private as the server it sits on.

SOVERIUS AI

Security — OWASP Top 10 for LLM Apps

LLM01

Prompt Injection

Direct injection, system prompt leak, indirect via websites/images

LLM02

Sensitive Info Disclosure

PII leakage, sensitive business data exposure

LLM08

Vector & Embedding Weaknesses

Unauthorized access & data leakage via embeddings

Our Use Case

Knowledge base within the intranet

Frontier model inference is too expensive at scale

Data must never leave the intranet

No backend — no pressure on IT team

→ **WebLLM: Run LLMs directly in the browser**

SOVERIUS AI

WebLLM — LLM Inference in the Browser

- We already own the hardware (user's GPU)
- Privacy by Design — data never leaves the device
- WebGPU-accelerated, 4-bit quantized models

~1.8 GB download for Llama-3.2-3B (cached after first load)

5–80 tok/s decode on consumer GPUs

```
import { CreateMLCEngine }
  from "@mlc-ai/web-llm";

const engine = await
  CreateMLCEngine(
    "Llama-3.2-3B-Instruct"
  );

// OpenAI-compatible API
const reply = await
  engine.chat.completions
    .create({ messages });
```

SOVERIUS AI

LLMs Don’t Know “Today” by Default

- Every LLM is constrained by its knowledge cut-off date
- Continual pre-training exists but causes catastrophic forgetting
- Full re-training costs millions and still produces a stale snapshot
- Even frontier labs only update every few months

Example: GPT-5.2 has a knowledge cutoff of Aug 31, 2025.
gpt-oss-20b (open source) has a cutoff of Jun 1, 2024 — almost 2 years behind.

Naive Approach — Just Stuff the Context?

In-Context Learning (ICL)

Add missing data directly into the prompt as extra context for the LLM.

We pay per token

Cost matters. Latency matters.
Stuffing context = paying for irrelevant text.

Why It Fails

Limited context size

128K tokens ≈ 250 pages max

Needle in a haystack

Models lose facts buried in long contexts

Lost in the middle

Attention fades for middle passages

Context rot

Quality degrades as context grows

Retrieval-Augmented Generation

Retrieve

Search for relevant document chunks via similarity

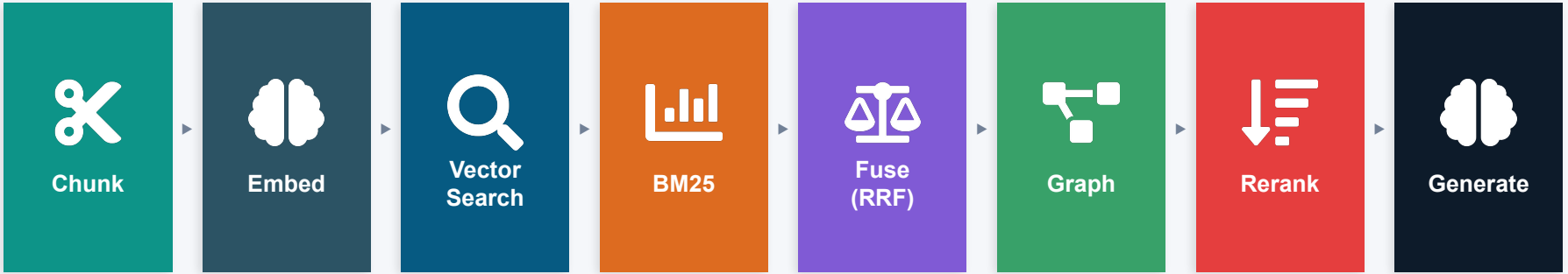
Augment

Inject the best chunks into the LLM prompt as context

Generate

LLM produces a grounded answer citing only the provided evidence

The RAG Pipeline — A System, Not a Prompt



Stage separation enables targeted diagnosis: if the answer is wrong, inspect which stage failed.

Offline (Index Time)

Chunk → Embed → Store in vector DB
Done once, cached locally

Online (Query Time)

Search → Fuse → Rerank → Generate
Real-time, per user query

FUN FACT

A 500-page doc becomes just 200–800 chunks

Each chunk is a self-contained, focused piece of text (200–1500 tokens) — small enough for fast retrieval, large enough to carry meaning.

That's like turning a novel into a deck of smart flash cards!

Step 1: Chunking Strategies

Fixed-Size

Slide a window of N tokens across the text with overlap

✓ Predictable, simple

✗ Splits mid-sentence

Heading-Based

Split at H1/H2/H3 boundaries, preserving section context

✓ Respects document structure

✗ Requires consistent headings

Semantic

Cluster paragraphs by embedding similarity, split at topic shifts

✓ Content-aware boundaries

✗ Needs embedding + tuning

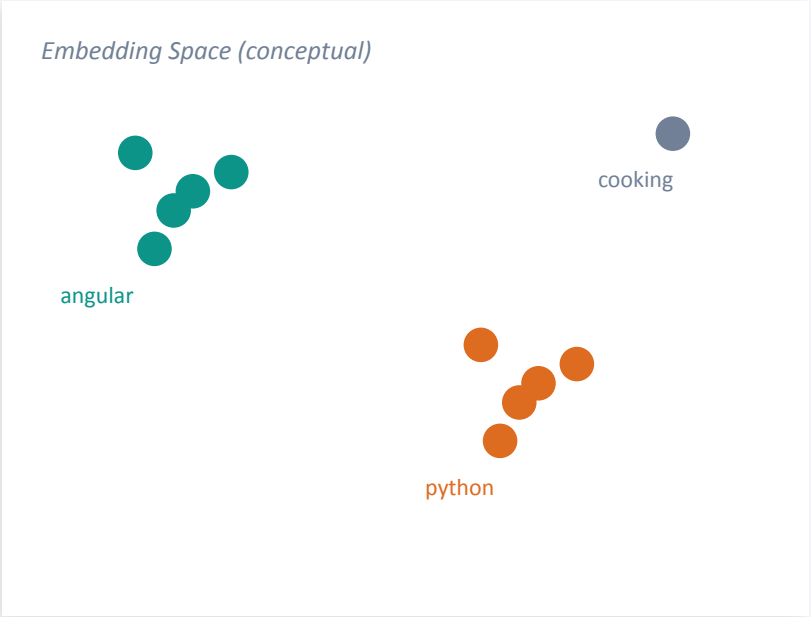
Start with heading-based if your docs have structure. Fall back to fixed-size. Upgrade to semantic if metrics are poor.

Step 2: Embedding Spaces

Turning text into math

An embedding model converts text into a high-dimensional vector where semantic similarity becomes spatial proximity.

Similar concepts cluster together. Unrelated text sits far apart.



Similarity Metrics

Cosine similarity
— angle between vectors (most common)

L2 / Euclidean
— straight-line distance

Inner product
— combines direction + magnitude

FUN FACT

768 numbers is all you need to capture any meaning

Feed in a word, a sentence, or an entire paragraph — the output is always the same: a fixed 768-dimensional vector. Our demo uses bge-base-en-v1.5 which maps any text to exactly 768 numbers. Beyond 512 tokens, input gets silently truncated — that’s why we chunk.

For comparison, we live in just 3 spatial dimensions!

Step 3: Vector Search in the Browser

Brute Force vs. HNSW

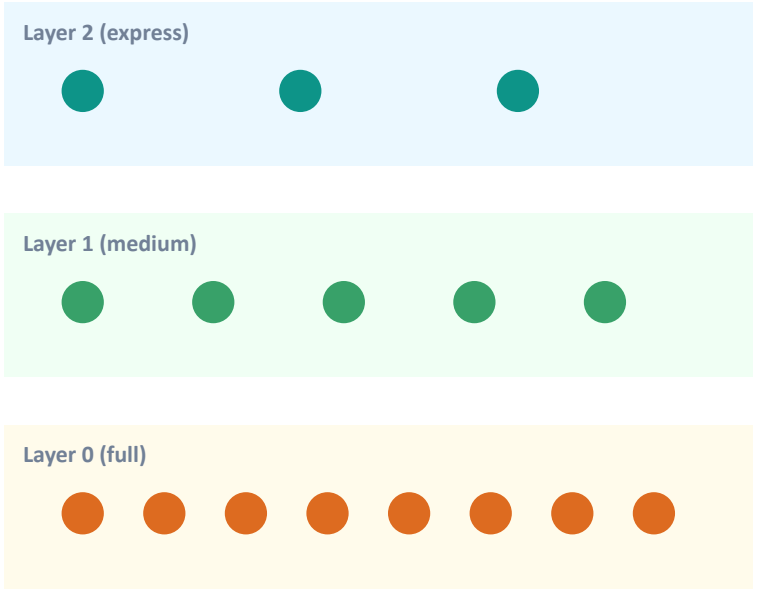
Brute Force O(N·D)

Compare query to every vector. Exact results.
OK for <5000 chunks, but doesn't scale.

HNSW O(log N)

Multi-layer graph with skip connections.
Sacrifices ~5% recall for 2x+ speed.
Powered by pgvector in PGlite (in-browser Postgres!).

HNSW: Hierarchical Navigable Small World



Step 4: BM25 — The Old-School Hero

Embeddings have a blind spot:

They miss exact terms — API names, error codes, rare identifiers, domain-specific jargon.

BM25 is a pure lexical (keyword) search that scores documents by term frequency and rarity (IDF).

Query Type	Embeddings	BM25
Paraphrase	✓✓	✗
Conceptual	✓✓	✗
Exact term	✗	✓✓
Error code	✗	✓✓
Jargon/acronym	~	✓✓

<1ms query time on thousands of chunks in the browser

BM25 = inverted index + IDF scoring. Lightweight enough for any browser.

FUN FACT

BM25 was invented in 1994 — and still beats many neural retrievers

Robertson & Walker published BM25 over 30 years ago.
It remains a top-3 baseline in modern information retrieval benchmarks.

Google still uses BM25 variants alongside neural models in production.

Step 5: Reciprocal Rank Fusion (RRF)

The Fusion Problem

Vector similarity scores (0.0–1.0) and BM25 scores (0–20+) are not comparable. We can't just add them together.

$$RRF(d) = \sum \frac{w_{\square}}{k + rank_{\square}(d)}$$

Discards raw scores. Only ranks matter. Scale-free, no calibration needed.

- Naturally handles missing results between signals
- k=60 (conservative) keeps diverse candidates for reranking
- Fusion empirically outperforms any single signal alone

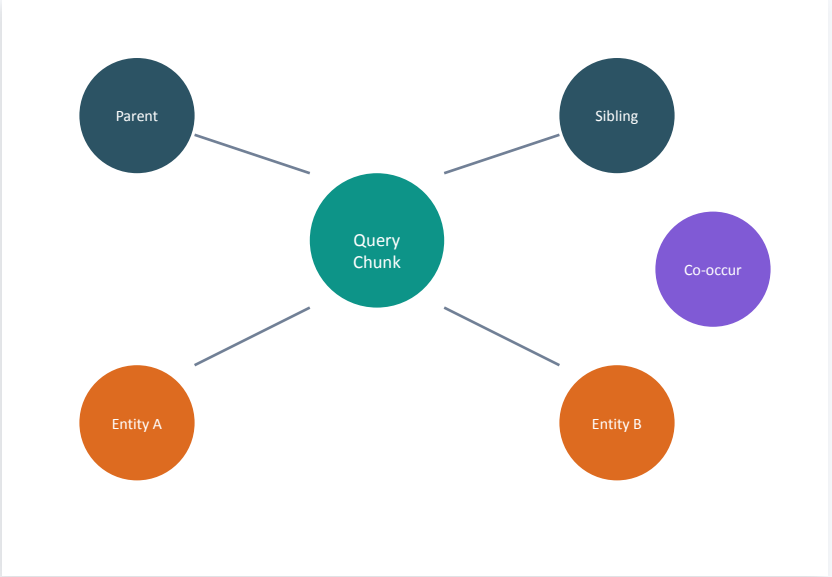
Step 6: Knowledge Graph Overlay

Adding structure beyond similarity

Graph Nodes:

Chunks (document pieces) + Entities (domain concepts)

Edge Types:
Structural: parent → child, sibling, follows
Entity-based: mentioned_in, co_occurs (with weights)



Weighted BFS from vector search seeds. Depth decay (α^d) prevents runaway expansion. Limit depth to 2.

Step 7: Cross-Encoder Reranking

Bi-Encoder (Retrieval)	Cross-Encoder (Reranking)
Query and document encoded independently. Fast (single pass), used for initial retrieval. But misses fine-grained interactions.	Query + document processed jointly. Detects negation, contrast, specificity. 5–20ms per pair → 200–800ms for 40 candidates.

When to enable reranking?

When Recall@20 is high but NDCG@5 is low — the right chunks are present but not ranked well.
Use MS-MARCO MiniLM-L12 (33M params) — runs in browser via Transformers.js / ONNX Runtime.

Step 8: Context Window Engineering

The Token Budget Problem

← 4096 tokens (e.g., Llama-3.2-3B context limit) →

Chunk Ordering Strategies

Rank-ordered

Best chunks first (pragmatic default)

Sandwich

Best at start & end; exploit U-shaped attention

Document-order

Preserve narrative coherence for multi-section queries

FUN FACT

LLMs pay attention like students in a lecture — mostly at the start and end

Research shows a U-shaped attention pattern: LLMs recall information placed at the beginning or end of their context much better than content in the middle. This is called the “Lost in the Middle” effect.

That’s why sandwich ordering matters!

Step 9: Grounded Answer Generation

The Grounding Contract

- Use ONLY the provided context
- Refuse when evidence is insufficient
- Cite sources for every claim

Small Language Models in Browser

1–3B parameter models (4-bit quantized)

Excel at extraction and rephrasing tasks.
Struggle with multi-hop reasoning.

10–50 tokens/sec on consumer GPUs via WebGPU.

Fallback Options When SLM Isn’t Enough:

Extractive QA (guaranteed grounding) • Show raw context directly • Cloud API for complex queries (on premise)

Generation quality depends primarily on retrieval quality — perfect generation can’t fix bad context.

Measuring What Matters — Evaluation

Evaluate retrieval and generation separately!

Recall@K

What fraction of relevant chunks appear in top-K?

NDCG@K

Are the best results ranked at the top?

MRR

Where does the first relevant chunk appear?

Diagnosis Framework

Low Recall@20	→ Fix retrieval (chunking, embeddings, BM25)
High Recall@20, Low NDCG@5	→ Add/tune reranking
High metrics, bad answers	→ Generation problem (prompt, model size)

FUN FACT

On-device RAG saves 96% vs. cloud APIs — and 100% of per-query costs

With WebLLM, inference runs on hardware you already own.
No per-token charges. No API keys. No data leaving the device.

At scale (thousands of users), the savings are enormous —
and you get privacy for free.

The In-Browser Tech Stack

WebGPU / WebAssembly	GPU compute + near-native WASM performance
PGlite + pgvector	Full Postgres in the browser with vector search
Transformers.js / ONNX	Embeddings + cross-encoder reranking
WebLLM (MLC)	On-device LLM inference, OpenAI-compatible API
Your App (Angular / React / Vue)	Framework-agnostic RAG pipeline

WebGPU: Chrome/Edge stable (v113+) • Safari partial (v26+) • Firefox flag-only • ~83% global coverage

SOVERIUS AI

Production Profiles — Pick Your Trade-off

Fast	Balanced	Quality
<1s	**1–3s**	**3–8s**
response time	response time	response time
Vector search only Top-5 chunks No reranking	Vector + BM25 + RRF Top-10 fused Reranking enabled	Vector + BM25 + Graph RRF + Reranking Larger context budget

Tuning discipline: change one thing at a time, measure, repeat.

SOVERIUS AI

Live Demo

RAG in a browser tab — no server, no API, no data leaving your device

Key Takeaways

- Privacy by architecture, not by policy — data never leaves the device
- RAG is a system, not a prompt — stage separation enables targeted diagnosis
- Multi-signal fusion (vector + BM25 + graph) beats any single approach
- Small Language Models are enough for grounded extraction + generation
- WebGPU enables real AI workloads in Chrome/Edge today (~83% global reach)
- 96% cost savings vs. cloud APIs at scale

Outlook — What's Next?

1. WebGPU at ~83% global coverage — Firefox and Safari still catching up to Chrome/Edge
2. Mobile RAG is already here — Qwen3.5-0.8B, SmolLM2, Gemma 3 run on iOS/Android today
3. Multimodal RAG — search images, diagrams, and tables alongside text
4. Federated search across multiple local knowledge bases
5. On-device fine-tuning — adapt models to your domain without leaving the browser

SOVERIUS
AI

Thank You!

https://soverius.ai

Private AI Search in a Tab