

brainjar — Go-To-Market Content

Generated: 2026-04-01 GitHub: <https://github.com/Farad-Labs/brainjar>

1. Subreddit Research

Subreddit	~Members	Why It's Relevant	Self-Promo Rules
r/rust	~320K	Rust CLI tool — home audience. r/rust regularly features open source Rust projects with Show HN-style posts.	Allowed with "project" flair. Must be substantive (no drive-by links).
r/LocalLLaMA	~530K	Privacy-first local AI crowd. Strong overlap with brainjar's zero-cloud value prop.	Self-promo OK if you contribute, don't spam. Be the builder, not the marketer.
r/AI_Agents	~130K	Directly about AI agent building. Memory for agents is a core topic here.	Generally open to open-source tools. No spammy affiliate/saas pitches.
r/mcp	~40K	Dedicated Model Context Protocol community. brainjar is an MCP server — perfect fit.	Very open. Community is builder-oriented and new tools are celebrated.
r/ObsidianMD	~175K	PKM/second brain crowd. They care about local-first,	Self-promo OK 1x/month per rule. Be

Subreddit	~Members	Why It's Relevant	Self-Promo Rules
		markdown, and knowledge graph workflows.	genuinely helpful in comments first.
r/selfhosted	~430K	Self-hosters love: zero cloud dependencies, SQLite, "runs offline", "you own the data."	Self-promo allowed for FOSS. Mark clearly as your own project.
r/commandline	~110K	CLI power users who appreciate fast, composable tools. brainjar's <code>--json</code> output and CLI flags hit home.	Self-promo OK for open source tools.
r/devtools	~65K	Developer tooling community — broad but relevant. Agent memory as dev infrastructure.	Open to showcase posts.
r/localfirst	~15K	Small but highly targeted. These people are evangelists for local-first software.	Enthusiastically welcome new local-first tools.

2. Reddit Posts

r/rust

Title: I built an AI agent memory system in Rust – local-first, SQLite, hybrid search (FTS5 + graph + fuzzy)

Body:

Been building AI workflows that needed persistent memory, and the cloud KB options all had the same problems: API latency, billing surprises, your data leaving your machine. So I built brainjar.

It's a Rust CLI that gives AI agents searchable memory backed entirely by SQLite. Sync your markdown/code files, optionally extract entities into a knowledge graph, then search via multiple engines:

```
brainjar search "deployment workflow"          # FTS5 + graph, ~33ms
brainjar search --fuzzy "deploymnt workflow"    # typo-corrected via Levenshtein vocabul
brainjar search --graph "LocalPage"             # graph traversal only
brainjar search --json "query" | jq .           # pipe-friendly
```

The interesting parts (Rust-wise):

- FTS5 virtual table in SQLite with triggers for auto-update on upsert
- Vocabulary fuzzy correction: all tokens extracted from docs on sync, stored with frequency counts, Levenshtein distance at query time (max 2-3 depending on word length). No file scanning at query time — it's all a SQLite lookup
- GraphRAG: optional LLM-backed entity/relationship extraction (Gemini, OpenAI, Ollama), stored in a separate `_graph.db`, traversed and RRF-merged with FTS5 results
- MCP server mode via stdio transport — works with Claude Code, Cursor, Windsurf

Search runs fully offline. All data lives in `~/.brainjar/<kb_name>.db`. One file, portable forever.

Also ships as an MCP server so agents can call `search_memory`, `sync_memory`, `get_status` directly.

Still early — vector embeddings are Phase 3 (sqlite-vec). Watch mode and a web graph UI are on the roadmap.

FOSS, MIT: <https://github.com/Farad-Labs/brainjar>

Curious what the Rust community thinks about the architecture choices, especially the vocabulary fuzzy vs embedding approach.

r/LocalLLaMA

Title: Built a local-first memory layer for AI agents – SQLite, zero API calls at query time, works offline

Body:

I kept running into the same problem with agent workflows: there's no good local option for persistent, searchable memory. Cloud knowledge bases are either expensive, slow, or both. Embedding everything in a vector DB is powerful but overkill for a lot of use cases.

So I built **brainjar** — local-first AI memory backed by SQLite.

What it does: - Syncs your markdown, code, and text files into SQLite - Extracts entities into a knowledge graph (optional, via local Ollama or API-based LLMs) - Hybrid search: FTS5 BM25 + graph entity traversal, merged with RRF - Fuzzy search that corrects typos against a vocabulary built from your own docs - Zero cloud dependencies — everything is in a `.db` file on your machine

Query latency: - Default (FTS + graph): ~33ms - Fuzzy: ~100ms - Text-only FTS5: ~10ms - All zero API calls

Privacy: Your memory doesn't leave your machine. Even entity extraction can be fully local if you run Ollama.

Also ships as an MCP server, so if you're running Claude Code, Cursor, or Windsurf, you can plug brainjar in as a memory tool directly.

I think the vector hype is real but also overapplied. For agent memory, FTS + graph + fuzzy catches most retrieval cases without the embedding overhead.

MIT, open source: <https://github.com/Farad-Labs/brainjar>

Would love feedback from people who've built agent memory systems — what edge cases have you hit?

r/AI_Agents

Title: Show r/AI_Agents: brainjar – persistent searchable memory for agents, backed by SQLite, zero cloud deps

Body:

One of the most common gaps I see in agent workflows: no good local memory. Agents that can reason but can't remember. Every session starts from scratch unless you stuff context into the prompt or hit a cloud KB.

I built **brainjar** to fill that gap — local-first, persistent, searchable memory for AI agents.

Key features: - **Hybrid search:** FTS5 full-text + graph entity traversal, merged via Reciprocal Rank Fusion - **Fuzzy correction:** typo-tolerant queries via vocabulary table (Levenshtein, built from your docs) - **GraphRAG:** entity/relationship extraction via LLM → traversable knowledge graph - **MCP server:** plug into Claude Code, Cursor, Windsurf as `search_memory` / `sync_memory` tools - **Zero cloud:** all data in SQLite, works offline, ~33ms queries

Setup is just:

```
cargo install --path .
brainjar init
brainjar sync
brainjar mcp # starts the MCP server
```

Then in your Claude Code or Cursor config:

```
{
  "mcpServers": {
    "brainjar": {
      "command": "brainjar",
      "args": ["mcp"]
    }
  }
}
```

Your agent can now call `search_memory` to retrieve context, `sync_memory` to ingest new files, and `get_status` to inspect the knowledge bases.

You can isolate memory per project (`--kb personal` , `--kb myproject`), search with JSON output for piping into other tools, and use `.brainjarignore` to exclude noise.

Free, MIT: <https://github.com/Farad-Labs/brainjar>

Feedback welcome — especially from anyone who's tried other memory solutions and hit walls with them.

r/mcp

Title: `brainjar` - open source MCP server for local AI memory (FTS5 + graph search, SQLite, zero cloud)

Body:

Built an MCP server that gives AI agents real persistent memory, backed entirely by local SQLite.

Tools exposed: - `search_memory` — hybrid search (FTS5 + graph RRF, fuzzy, text-only, graph-only) - `sync_memory` — ingest/update files into knowledge bases - `get_status` — inspect KB stats and sync state

Config (Claude Code / Cursor / Windsurf):

```
{
  "mcpServers": {
    "brainjar": {
      "command": "brainjar",
      "args": ["mcp"]
    }
  }
}
```

What makes it different from other memory MCPs: - Zero cloud — all data lives in `~/.brainjar/*.db` (SQLite) - No embedding model required — FTS5 + graph handles most retrieval - Fuzzy search built from your own document vocabulary — tolerates typos without an API call - Optional GraphRAG via Ollama (fully local) or

Gemini/OpenAI - Multiple isolated knowledge bases (personal, per-project) - Ships as a CLI too — useful for debugging, scripting, and piping

Works offline. No API calls at query time. Query latency is 10–100ms depending on mode.

MIT, written in Rust: <https://github.com/Farad-Labs/brainjar>

r/ObsidianMD

Title: I built a local-first search layer for your Obsidian vault – works as MCP memory for AI agents

Body:

For anyone who uses Obsidian as an AI agent knowledge base (or just wants fast local search), I built something you might like.

brainjar is a CLI that syncs your markdown files into SQLite and gives you hybrid search — FTS5 full-text + entity graph traversal, with fuzzy correction for typos. No cloud, no API at query time.

```
# Point it at your vault
# brainjar.toml:
[knowledge_bases.vault]
watch_paths = ["~/Documents/Obsidian/MyVault"]
auto_sync = true

brainjar sync
brainjar search "fleeting notes GTD"
brainjar search --fuzzy "prjectmanagment" # corrects typos
brainjar search --graph "Zettelkasten"    # finds connected concepts
```

If you use Claude Code or Cursor, it also runs as an MCP server — so your AI coding assistant can directly search your vault for context.

All data stays local. One SQLite file per knowledge base. No telemetry, no accounts.

Entity extraction (optional) builds a knowledge graph from your notes using configurable LLMs — including local Ollama, so you can keep it fully offline.

Obsidian vaults are already plaintext markdown, so there's essentially zero friction to get started. Just point it at your vault directory.

MIT, open source: <https://github.com/Farad-Labs/brainjar>

r/selfhosted

Title: Show r/selfhosted: brainjar – local-first AI memory for agents. SQLite, zero cloud, offline capable

Body:

Self-hosters, this one's for you.

brainjar is a Rust CLI that gives AI agents persistent, searchable memory without any cloud dependencies.

The short version: - Syncs your docs/code into SQLite - Hybrid search: FTS5 + knowledge graph, merged via RRF - Fuzzy typo correction via local vocabulary table (Levenshtein) - Optional entity extraction via Ollama (fully local LLM) - MCP server for Claude Code, Cursor, Windsurf - All data in `~/.brainjar/*.db` — one file, yours forever - No accounts, no telemetry, no surprise bills

Costs: If you use Ollama for entity extraction, \$0.00/month. If you use Gemini Flash Lite, ~\$0.66/month for initial ingestion + daily sync of ~300 docs.

Search cost at query time: \$0.00. All local SQLite.

This is the kind of tool I wanted to exist before I built it — something that treats AI memory as infrastructure you own, not a service you rent.

MIT, written in Rust: <https://github.com/Farad-Labs/brainjar>

r/commandline

Title: brainjar – CLI for local AI memory. FTS5 + graph search, fuzzy, SQLite, MCP server. Written in Rust

Body:

Built a CLI tool that's been genuinely useful in my day-to-day — sharing in case it's useful for others.

brainjar gives AI agents (and humans) fast, local, searchable memory from their own files.

```
$ brainjar search "deployment workflow"
→ FTS5 + graph, ~33ms

$ brainjar search --fuzzy "deploymnt workflw"
🔍 corrected: deploymnt → deployment, workflw → workflow
→ results...

$ brainjar search --json "agent memory" | jq '.[0].score'
→ 0.94

$ brainjar status
→ KB: personal | 276 docs | last sync: 2m ago
```

Flags: - (default) — FTS5 BM25 + graph RRF (~33ms) - --fuzzy — vocabulary-corrected FTS + graph (~100ms) - --text — pure FTS5 BM25 (~10ms) - --graph — entity graph traversal only (~20ms) - --local — nucleo file scanner (unsynced files) - --exact — case-insensitive substring - --json — machine-readable output - --kb <name> — target specific knowledge base - --limit N — result count

Also runs as an MCP server (`brainjar mcp`) so AI tools can call it programmatically.

Zero cloud deps. SQLite. Rust. MIT: <https://github.com/Farad-Labs/brainjar>

r/localfirst

Title: brainjar – local-first AI memory with hybrid search. SQLite, offline-capable, you own the data

Body:

Finally built the thing I wanted but couldn't find: a local-first memory layer for AI agents.

brainjar treats AI memory the way local-first software should: - All data in a single SQLite `.db` file on your machine - No accounts, no cloud, no API calls at query time - Works offline — full functionality without internet access - Portable: move the `.db`, move your memory - Open source, MIT

How it works: 1. `brainjar sync` — indexes your files into SQLite (FTS5 virtual table, vocabulary, optional entity graph) 2. `brainjar search "query"` — hybrid FTS5 + graph traversal, ~33ms 3. `brainjar mcp` — runs as MCP server for Claude Code, Cursor, etc.

Even entity extraction (GraphRAG) can run fully locally if you point it at Ollama.

The philosophy: your memory is infrastructure you own, not a service you subscribe to. One `.db` file, backed up with your normal backup strategy, portable anywhere Rust runs.

For privacy-sensitive workflows, this is particularly useful — no data ever leaves your machine if you run Ollama for extraction.

MIT: <https://github.com/Farad-Labs/brainjar>

3. Hacker News Post

Title: Show HN: Brainjar – Local-first AI memory with hybrid search (FTS5 + graph + fuzzy) backed by SQLite

Body:

I built brainjar because I kept hitting the same problem: AI agents with no persistent memory, cloud KBs with latency and billing surprises, and vector DBs that are powerful but overkill for most agent memory use cases.

brainjar syncs your files into SQLite and gives you hybrid search: FTS5 BM25 full-text search merged with knowledge graph traversal via Reciprocal Rank Fusion. On top of that, a fuzzy layer corrects typos against a vocabulary table built from your own document corpus (Levenshtein, purely local, no API at query time).

Search modes: - Default (FTS + graph): ~33ms - Fuzzy: ~100ms
- Text-only: ~10ms - Graph-only: ~20ms

The fuzzy approach is deliberate. Vector search is great for semantic similarity but has real tradeoffs for agent memory: API costs, latency, and opacity. FTS + graph gives you exact precision and semantic breadth through entity relationships, with fuzzy for the typo/abbreviation cases. Vector embeddings are on the roadmap (sqlite-vec) as an additive layer, not a replacement.

Entity/relationship extraction is optional — configurable against Gemini, OpenAI, or Ollama. With Ollama you're fully offline.

Also ships as an MCP server (stdio transport), so Claude Code, Cursor, Windsurf can call `search_memory` directly.

Written in Rust. All data in `~/.brainjar/*.db`. MIT.

<https://github.com/Farad-Labs/brainjar>

Would particularly welcome feedback on the architecture tradeoffs — especially the FTS+graph vs pure-vector question for agent memory.

4. LinkedIn Post

Why AI agents keep forgetting things (and what I built about it)

Every AI agent workflow hits the same wall eventually: the agent is smart in the moment, but each session starts from scratch. You can stuff context into prompts, but that gets expensive and brittle fast. Cloud knowledge bases work, but you're paying for storage, paying for queries, and your data lives on someone else's infrastructure.

I've been building AI workflows long enough to find this friction genuinely annoying, so I built something to fix it.

brainjar is a local-first AI memory tool for agents and developers. It syncs your files into SQLite and gives you hybrid search — fast BM25 full-text search merged with graph entity traversal, plus typo-tolerant fuzzy search built from your own document vocabulary. All of it runs locally. Query time is 10-100ms. Zero API calls at search time.

A few things I'm proud of: - **Zero cloud deps.** All data in a single SQLite file. Works offline. You own it. - **MCP server.** Plug it into Claude Code, Cursor, or Windsurf as a native memory tool. - **GraphRAG.** Optional entity extraction builds a traversable knowledge graph. Can run fully local via Ollama. - **Cost.** ~\$0.66/month for entity extraction on ~300 docs (Gemini Flash Lite). Search: \$0.00.

The framing I keep coming back to: AI memory should be infrastructure you own, not a service you rent. One `.db` file. Backed up with your normal backups. Portable anywhere.

It's free, open source (MIT), written in Rust: <https://github.com/Farad-Labs/brainjar>

If you're building AI agent workflows and memory has been a pain point, I'd genuinely love to hear what you've tried and what's worked.

AI Agents #OpenSource #LocalFirst #MCP #Rust #DeveloperTools #LLM

5. skills.sh Findings

Relevant skill found: `coreyhaines31/marketingskills` — specifically the `launch-strategy` skill.

This is a comprehensive open-source marketing skills collection for Claude Code and AI agents. The most relevant skills for brainjar's launch:

Skill	Use Case
<code>launch-strategy</code>	Product launch planning — announcements, sequencing, channels
<code>social-content</code>	Twitter/X and LinkedIn content for the launch wave
<code>content-strategy</code>	Planning dev-focused blog/article content
<code>copywriting</code>	Homepage/README copy optimization
	SEO pages comparing brainjar to Mem0, Zep, etc.

Skill	Use Case
competitor-alternatives	
customer-research	Reddit/forum mining for agent memory pain points
marketing-ideas	140 SaaS marketing ideas to mine for open-source tactics
ai-seo	Optimization for AI-generated search results (AEO/ GEO)

Install:

```
npx skills add coreyhaines31/marketingskills --skill launch-strategy social-content
```

Also notable: The skills.sh leaderboard shows `inferen-sh/skills` (1.1M total downloads) — could be worth submitting brainjar as a skill to that collection once the MCP server is polished.

6. Additional GTM Ideas

Dev.to Article

Write a technical deep-dive: **"Why I built AI agent memory with SQLite instead of a vector database"**

Angle: The FTS5 + graph + fuzzy vs. pure-vector tradeoff. Show benchmarks (33ms local vs 500ms+ cloud), the vocabulary fuzzy architecture, the RRF merge. This is the kind of "contrarian technical take" that gets traction on Dev.to and gets crossposted to HN.

Product Hunt Launch

Yes, absolutely launch here. Timing: coordinate with the HN Show HN post on the same day.

- Category: Developer Tools
- Tagline: *"Local-first AI memory. SQLite, hybrid search, MCP server. No cloud."*
- Hunter: ideally someone with followers in the AI tools space (reach out to known hunters in dev tools)
- Gallery assets: search demo GIF, architecture diagram, benchmark comparison table
- First comment: personal founder story ("I built this because cloud KBs kept failing me at 2am...")

Discord Communities

Community	Notes
Claude Discord (Anthropic)	Post in #tools or #mcp-servers — perfect fit
Cursor Discord	MCP integrations channel
Windsurf Discord	Similar to Cursor
Continue.dev Discord	Open-source VS Code AI extension with MCP support
LangChain Discord	Large agent-builder community
AI Tinkerers (Slack/ Discord)	Local chapters, builder-focused
Obsidian Discord	#plugins and #share-showcase channels
Rust Community Discord	#showcase channel

Twitter/X Strategy

Account handle suggestion: @brainjar_ai or @brainjarmem

Content strategy:

1. **Launch thread** — technical walkthrough with code snippets. Show the fuzzy correction in action ("🐞 corrected: deploymnt → deployment"). Genuine reaction bait.
2. **Benchmark thread** — "I ran 1000 agent memory queries locally vs cloud KB. Here's what I found." Show latency, cost, privacy tradeoffs.
3. **Build in public** — weekly updates on what shipped (vector embeddings, watch mode, web UI). Tag @AnthropicAI, @cursor_ai, @windsurf_ai when MCP integrations are relevant.
4. **Response marketing** — search Twitter for "AI agent memory", "Claude Code memory", "MCP server memory" and reply with genuine context about brainjar where relevant. Don't spam.
5. **Demo video** — 60-second screen recording showing sync + search + MCP integration in Claude Code. Repost as Reels/Shorts.

MCP Directory Listings

Submit to these directories immediately — low effort, permanent inbound:

Directory	URL	Notes
PulseMCP	https://pulsemcp.com	11,820+ servers, well-indexed. Submit form.
MCP.so	https://mcp.so	Submit via GitHub PR or form
Glama.ai	https://glama.ai/mcp/servers	Curated MCP directory
Awesome MCP Servers (GitHub)	Search GitHub for this repo	Open PR to add brainjar
Claude.ai MCP docs	Anthropic's own docs	Get listed as an example server

README Badges + GitHub Signal Amplification

- Add shields.io badges: Crates.io version, license, stars
- Post in GitHub Discussions for Claude Code / Cursor asking for feedback (stay within ToS)
- Consider a `llms.txt` or `agents.md` file in the repo (trending per r/rust threads — helps LLMs understand how to use the tool)
- Submit to **crates.io** with good keywords: `mcp` , `agent-memory` , `knowledge-graph` , `local-first` , `sqlite` , `fuzzy-search`

Timing Recommendation

Day	Action
D-3	Submit to MCP directories (PulseMCP, Glama, MCP.so)
D-2	Post r/mcp and r/localfirst (smaller, builds early engagement)
D-1	Post r/AI_Agents and r/selfhosted
D0 (Launch Day)	HN Show HN + Product Hunt simultaneously, LinkedIn post
D0 afternoon	r/rust, r/commandline, r/devtools
D+1	r/LocalLLaMA (after HN traction screenshot is available)
D+2	r/ObsidianMD
D+3	Dev.to article published, crosspost thread on Twitter/X
D+7	Discord community posts (Cursor, Windsurf, Claude, Rust)

All content written for developer audiences. No hype. Show don't tell.