LOUST-PRO/LLMmempipe

Source: LOUST · **Status:** Shipped · **Period:** 2025-08 (active)

Memory-pipeline cleaner for LLM chat exports: canonical normalization, deterministic PII scrubbing, and diff-friendly output.

The challenge

My chat history exports from Claude, ChatGPT, and Gemini each had their own schema, their own stale PII, and their own broken tool calls. When I fed them into long-running agents as long-term memory, the contamination propagated downstream and surfaced weeks later as confused behavior.

My role

I built a Rust-based cleaner that normalizes heterogeneous chat exports into a single canonical schema with deterministic, rule-based PII scrubbing and a diff-friendly output format.

What I did (3–5 decisions)

1. Single canonical schema for all vendors

Why: One schema means one audit chain. Per-vendor schemas mean per-vendor failures, and the agent's memory picks up whichever vendor's quirks leaked in.

Tradeoff: Some vendor-specific features get flattened during normalization. For a memory substrate the consistency wins; for a feature showcase it would not.

2. Deterministic, rule-based PII scrubbing

Why: Regex guessing leaks, and a reviewer cannot trace a redaction back to the rule that fired. A small, auditable rule book makes every redaction explainable.

Tradeoff: New PII patterns require explicit rule updates. The rule book stays small because the same dozen patterns keep appearing across vendors.

3. Diff-friendly output format

Why: Memory file drift has to be reviewable per line. Per-line diffs catch what file-level diffs miss — a single bad export no longer hides inside a hundred clean ones.

Tradeoff: Output files are larger than compact formats. The reviewability earns the extra bytes.

What changed

Schema drift across vendors

Baseline: 3 vendors, 3 schemas, 3 failure modes

1 canonical schema, 1 failure surface

PII scrub coverage

Baseline: About 70% (regex heuristics)

About 98% (rule-based + audit trail)

Evidence: estimate, not measured

Memory file reviewability

Baseline: File-level diff (single line buried in hundreds)

Per-line diff with stable IDs

Overall tradeoffs

I gave up some per-vendor feature richness for the canonical substrate. For long-running agent memory the substrate is the product; the vendor-specific extras would be a tax.

Lessons

Memory substrate quality is the limiting factor on long-running agents, not the agent's reasoning. A clean substrate lets a small model perform like a large one.

Stack

Rust | LLM tooling | normalization

Repository

github.com/LOUST-PRO/LLMmempipe

Evidence

- [repo] github.com/LOUST-PRO/LLMmempipe