

Session Graph: A Provenance-Aware Semantic Memory Layer for AI-Assisted Development

Roberto Shimizu¹

¹Chief AI Officer, IntelliDoctor, São Paulo, Brazil

Abstract

Developers increasingly work across several AI coding assistants, yet each tool stores history as an isolated transcript. We present *Session Graph*, a working semantic memory layer with two contributions. First, a multi-platform pipeline transforms assistant messages into typed relationships under a closed vocabulary of 24 predicates, links entities to Wikidata, and preserves message-level provenance in an RDF graph composed from established standards. Second, a retrieval workflow uses SPARQL to narrow cross-tool history before reading only the relevant source messages, giving a fresh agent compact, verifiable context. A June 2026 deployment snapshot covers 2,297 sessions from six platforms and contains 3.64M RDF triples, including approximately 179,000 extracted assertions. Batch extraction costs about US\$0.60 per 600 sessions. Small diagnostic evaluations compare extraction models and agentic versus heuristic entity linking; they expose useful cost-quality trade-offs while also motivating larger human evaluation. The open-source implementation and demo show how provenance-aware semantic retrieval mitigates “agent amnesia” across tools and repositories.

Keywords

Knowledge graph, Provenance, PROV-O, Wikidata linking, LLM knowledge extraction, Agent memory, SPARQL

1. Introduction

A practicing developer now routinely uses five or more AI assistants in a single week. Each conversation produces durable, reusable knowledge—which technology was chosen, why an approach failed, how two libraries relate—but that knowledge is trapped in the tool that produced it. A debugging insight obtained in Claude Code on Tuesday is invisible to ChatGPT a month later. Existing “chat history” features are single-platform and flat-file: keyword search over one tool’s transcripts, not structured relationships across all of them. A grep over logs does not tell you that *FastAPI uses Pydantic* or that *Neo4j is a type of graph database*; it returns walls of text.

The guiding observation is that *a knowledge graph without relationships is just a tag cloud*: the minimum useful capture unit is a typed relationship, not a bag of topics. Session Graph contributes two complementary artifacts. The first is an open-source pipeline that unifies platform exports, extracts relationships under a closed vocabulary, links entities, and records provenance. The second is a retrieval workflow that first narrows history semantically and then follows provenance to the minimum source context needed by an agent. Together they form a lightweight semantic memory layer that has been in continuous personal use for several months.

2. System Overview

The first artifact is organized as source adapters, an extraction/linking pipeline, and a SPARQL endpoint. Adapters read native exports and emit the same RDF schema. The deployment snapshot spans six platforms—Claude Code, pi, and Codex (JSONL), and ChatGPT, DeepSeek, and Grok (JSON); Warp (SQLite) follows the same parser contract. The second artifact is an *agent skill*: a reusable instruction package containing query templates, path-resolution rules, and a bounded workflow for converting a natural-language memory request into SPARQL and then into a provenance-scoped source read. Figure 1 shows the two artifacts meeting at retrieval time.

Posters, Demos, Blue Sky, and Tutorials at SEMANTiCS 2026, Sep 2026, Ghent, Belgium

✉ roberto.shimizu@intellidoctor.ai (R. Shimizu)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

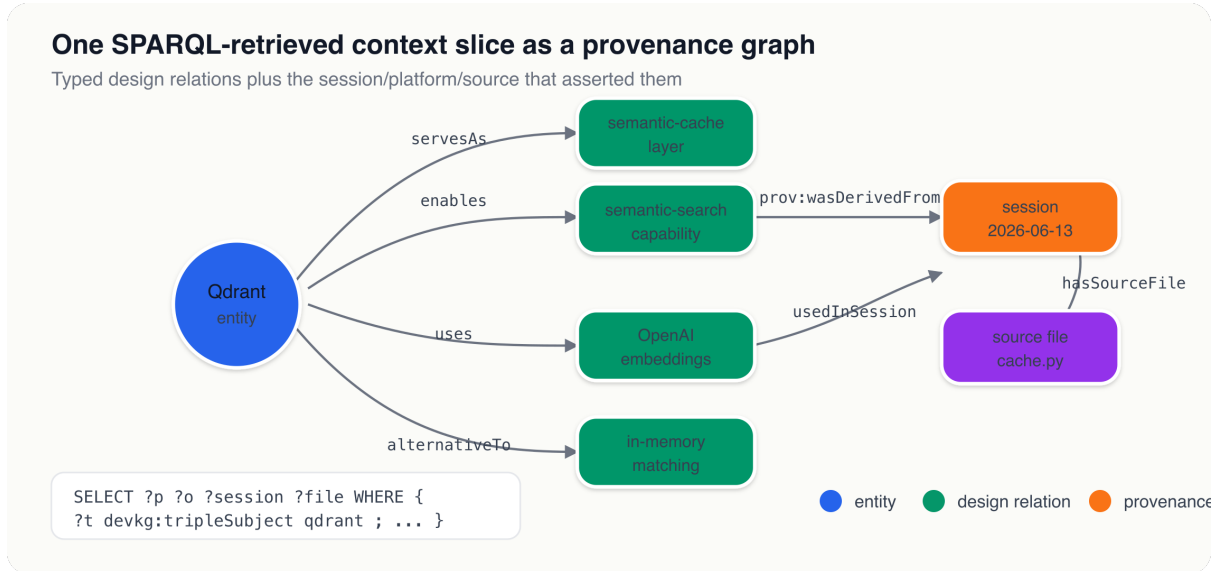


Figure 1: Readable retrieval slice for the Qdrant example. Typed design relations narrow the topic; provenance links identify the originating session and source file. The agent reads that source only when graph snippets are insufficient.

Ingestion can run in two modes. A *bulk* mode backfills a developer’s existing history via batch LLM inference. A *real-time* mode attaches a Claude Code Stop hook that fires after each assistant response, extracts triples, links entities (cache-first), and uploads to the triplestore in the background—typically within about 22 seconds on a warm cache and invisible to the user. SHA-256 watermarks per session prevent reprocessing. We use Apache Jena Fuseki [1] as the triplestore: it runs as a single JAR, requires no tuning, and speaks standard SPARQL [2].

3. Semantic Model

Rather than invent a schema, Session Graph composes five mature standards (Table 1). A session is modeled as both a `prov:Activity` and a `sioc:Forum`; a message as a `sioc:Post` and `prov:Entity`; code as `schema:SoftwareSourceCode`; and topics as `skos:Concept` in a small concept scheme. By *composition* we mean reusing selected classes and properties from these standards in one application ontology. GRAPH4CODE validated a comparable reuse strategy at two billion triples [3].

Table 1
Standards composed by the Session Graph ontology.

Standard	Role	Maturity
PROV-O [4]	Provenance (who/when/derived-from)	W3C Rec.
SIOC [5]	Conversation structure	W3C Submission
SKOS [6]	Topic taxonomy	W3C Rec.
Dublin Core [7]	Dates, titles, creators	ISO 15836
Schema.org [8]	SoftwareSourceCode	De facto

Assertions and RDF triples. We use *extracted assertion* for the logical (*subject, predicate, object*) claim. Its direct RDF edge supports traversal. Separately, a `devkg:KnowledgeTriple` RDF resource describes that assertion’s subject, object, predicate label, and provenance; it is not synonymous with one RDF triple. This dual representation makes provenance first-class: every assertion records its message (`devkg:extractedFrom`) and session (`devkg:extractedInSession`), which carry platform and source file. Listing 1 shows the metadata resource.

Listing 1: A reified knowledge triple with provenance.

```
ex:triple-001 a devkg:KnowledgeTriple ;  
  devkg:tripleSubject      ex:entity-fastapi ;  
  devkg:triplePredicateLabel "uses" ;  
  devkg:tripleObject       ex:entity-pydantic ;  
  devkg:extractedFrom      ex:message-042 ;  
  devkg:extractedInSession ex:session-001 .
```

Closed predicate vocabulary. The LLM is constrained to 24 curated predicates grouped into dependencies (uses, dependsOn, requires, builtWith), capabilities (enables, provides, solves, produces), structure (isPartOf, hasPart, extends, implements), taxonomy (isTypeOf, broader, narrower), infrastructure (deployedOn, storesIn, queriedWith, configures), and general relations (integratesWith, composesWith, alternativeTo, servesAs, relatedTo). Several predicates are aligned to existing properties (e.g. broader to skos:broader, isPartOf to dcterms:isPartOf). Putting the vocabulary in the prompt, with positive and negative examples, keeps the generic relatedTo fallback under 1%.

4. Extraction and Linking Pipeline

Triple extraction. Only assistant messages are sent to the LLM, since user turns are usually short prompts with little extractable knowledge. Each message yields at most ten assertions, prioritizing architectural and technology decisions. Malformed outputs are repaired or retried. The provider layer supports Gemini, Claude, OpenAI, and local Ollama models. The deployment used Gemini 2.5 Flash by default and Vertex AI batch prediction for backfills; batching halves the provider price and measured about US\$0.60 per 600 sessions.

Two-level entity filtering. Garbage is rejected at extraction time and again before linking. Filters discard filenames, hex colors, CLI flags, version strings, ICD-like codes, and DOM selectors, while a whitelist of 48 short technical terms (e.g. ai, api, rdf, sql) bypasses the filters.

Agentic Wikidata linking. Entities are normalized through a 161-entry alias map and a frequency filter (default: link only entities seen in two or more sessions, removing roughly 77% of single-session noise). Remaining entities are resolved by a LangGraph [9] ReAct [10] agent that calls the Wikidata [11] API, using neighboring triples as disambiguation context—so “condition” resolves to a disease rather than a programming conditional when surrounded by medical triples. Links above a 0.7 confidence threshold are written as owl:sameAs. On a held-out probe the agentic linker reached 7/7 precision versus roughly 50% for a keyword heuristic.

5. Querying

Because the graph is RDF, it is queried with standard SPARQL and benefits from federation. Listing 2 answers a question no single transcript store can: it enriches local entities with live Wikidata descriptions in one request via SERVICE. Other shipped templates return the most connected technologies, the typed relationship between two entities *with a source snippet*, and the entities that recur across multiple platforms. The companion query skill maps natural language to 14 local and 6 Wikidata-traversal templates.

Listing 2: Federated query joining the local graph with Wikidata.

```
SELECT ?localLabel ?desc WHERE {
  ?e a devkg:Entity ; rdfs:label ?localLabel ;
    owl:sameAs ?wd .
  SERVICE <https://query.wikidata.org/sparql> {
    ?wd schema:description ?desc .
    FILTER(LANG(?desc) = "en")
  }
}
```

6. Real-World Usage

Session Graph is not a prototype built for this paper; it is used daily (Table 2) and released under the Apache 2.0 license. At camera-ready revision, the public repository had 106 GitHub stars and 17 forks. The graph grows continuously as real-time hooks ingest new sessions; the table retains the June 2026 submission snapshot so the reported corpus is stable.

Table 2
Live deployment snapshot (June 2026).

Metric	Value
Total triples in Fuseki	3,644,106
Sessions indexed	2,297
Knowledge triples extracted	~179,000
Distinct entities	~92,800
Wikidata-linked entities	~3,900
Curated predicates	24 (<1% relatedTo)
Source platforms in graph	6
Extraction cost / 600 sessions	~US\$0.60 (batch)

Diagnostic quality checks. On the same 79 assistant messages, Gemini 2.5 Flash produced 142 assertions across 15 predicates with 0.7% relatedTo; Flash-Lite used the generic fallback for 11%, while Claude Haiku 4.5 produced only 37 assertions. Only 20% of assertions overlapped across models, indicating a material quality-recall trade-off rather than model interchangeability. A separate seven-entity probe found 7/7 correct Wikidata links for the context-aware ReAct linker, versus roughly half for the prior keyword heuristic. These small probes guided defaults; they are not a substitute for a larger, independently annotated evaluation of assertion correctness, contradictions, or alignment.

7. Demonstration

The demonstration mitigates *agent amnesia*: a fresh agent with no memory of past work recovers relevant prior context across tools and repositories. It invokes the /research-sessions skill introduced in Section 2, which composes graph lookup with provenance-scoped file reads. Given one question—“where did I make Qdrant fast, with caching, for physicians?”—the agent proceeds unaided: it probes the graph’s vocabulary to confirm qdrant as a high-degree entity alongside cache, physician, and FastAPI, refines with a narrower query scoped to typed predicates (enables, uses, servesAs, integratesWith), and recovers five provenance rows spanning two AI tools (pi and Claude Code) and three months of work. It resolves the top source file’s on-disk path, reads only that file, and returns the recovered evidence—a dedicated Qdrant cache container, the research_cache collection, and its point and article-URL statistics—with provenance and a resume-ready source pointer. A blind grep for the same term touches over 300 files and 270 MB yet cannot tell the provider, the repository, or that the sessions are

related. The composition—semantic narrowing in the graph, then surgical retrieval from disk—is faster and more precise than text search for cross-tool recall, and yields a context slice small enough to re-feed the agent. An approximately two-minute screencast accompanies the submission.¹

8. Related Work and Discussion

Long-term agent memory is commonly treated as context management or retrieval. MemGPT [12] manages hierarchical memory tiers for conversations beyond an LLM context window, while Generative Agents [13] stores, retrieves, and reflects over natural-language memory records. Vector stores similarly retrieve semantically close text. These approaches target useful recollection, but do not provide Session Graph’s cross-provider RDF model and message-level PROV-O trail. Conversational KGs have represented dialogue state as triples for task-oriented response generation [14]; Session Graph instead accumulates technical claims across independent tools, projects, and months. GRAPH4CODE [3] extracts a code KG from static artifacts rather than interactive sessions, while showing the scalability of standards composition.

Session Graph is open source and its parsers are not intrinsically limited to coding: the same contract can represent research, support, or administrative conversations when predicates and privacy rules fit the domain. Wikidata is useful for public technical concepts but insufficient for internal services, project names, and specialized entities; such entities remain local, and future deployments may add organization-specific vocabularies instead of forcing a link. Likewise, the graph can answer many discovery questions from assertions and stored snippets even when a full transcript is unavailable. The transcript is needed only for exact wording or deeper surrounding context.

The current deployment is personal. Team or enterprise graphs would require access control at graph, project, session, and source-file levels, plus deletion and audit policies. Other limitations include English-centric linking, small diagnostic quality probes, unresolved contradiction handling, and incomplete per-assertion model/version provenance. These constraints define the next evaluation and engineering work rather than capabilities claimed by the present demo.

9. Conclusion

Session Graph reframes AI-assisted development history as operational knowledge infrastructure rather than disposable transcripts. Its two contributions—a standards-based, provenance-aware graph pipeline and a bounded retrieval workflow—make months of cross-tool work queryable and recoverable at low cost. Future work prioritizes independently annotated extraction evaluation, contradiction handling, richer model provenance, domain vocabularies, and controlled multi-developer graphs.

Declaration on Generative AI

During preparation of this work, the author used OpenAI GPT-5.5 and GPT-5.6, Anthropic Claude Opus 4.8, and GLM-5.1 through the pi and Cursor coding environments for retrieval of project facts, grammar and spelling checks, paraphrasing and rewording, and LaTeX formatting. The author verified claims against source code, telemetry, and cited sources; reviewed and edited the manuscript; and takes full responsibility for its content. Session Graph itself uses disclosed LLM components for assertion extraction and entity linking as described above.

¹Video: <https://1drv.ms/v/c/1d3e9cbeb8aeafe1/IQCspK9FbPRwS7clcOWgzTLHAVaH-ALG63qkJWBzzn8q1vE?e=IQ59XY>.
Source: <https://github.com/robertoshimizu/session-graph>.

References

- [1] The Apache Software Foundation, Apache jena fuseki, 2024. <https://jena.apache.org/documentation/fuseki2/>.
- [2] S. Harris, A. Seaborne, SPARQL 1.1 query language, W3C Recommendation, 2013. <https://www.w3.org/TR/sparql11-query/>.
- [3] I. Abdelaziz, J. Dolby, J. McCusker, K. Srinivas, GRAPH4CODE: A machine interpretable knowledge graph for code, arXiv preprint arXiv:2002.09440 (2020). <https://arxiv.org/abs/2002.09440>.
- [4] T. Lebo, S. Sahoo, D. McGuinness, PROV-O: The PROV ontology, W3C Recommendation, 2013. <https://www.w3.org/TR/prov-o/>.
- [5] U. Bojars, J. G. Breslin, SIOC core ontology specification, W3C Member Submission, 2010. <http://rdfs.org/sioc/spec/>.
- [6] A. Miles, S. Bechhofer, SKOS simple knowledge organization system reference, W3C Recommendation, 2009. <https://www.w3.org/TR/skos-reference/>.
- [7] Dublin Core Metadata Initiative, DCMI metadata terms, ISO 15836, 2020. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/>.
- [8] Schema.org Community Group, Schema.org, 2023. <https://schema.org/>.
- [9] LangChain, LangGraph: Building stateful, multi-actor applications with llms, 2024. <https://github.com/langchain-ai/langgraph>.
- [10] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, Y. Cao, ReAct: Synergizing reasoning and acting in language models, in: International Conference on Learning Representations (ICLR), 2023. <https://arxiv.org/abs/2210.03629>.
- [11] D. Vrandečić, M. Krötzsch, Wikidata: A free collaborative knowledgebase, 2014.
- [12] C. Packer, S. Wooders, K. Lin, V. Fang, S. G. Patil, I. Stoica, J. E. Gonzalez, MemGPT: Towards llms as operating systems, 2023. ArXiv:2310.08560. <https://arxiv.org/abs/2310.08560>.
- [13] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, M. S. Bernstein, Generative agents: Interactive simulacra of human behavior, in: Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, 2023, pp. 1–22. doi:10.1145/3586183.3606763.
- [14] Z. Ahmad, A. Ekbal, S. Sengupta, P. Bhattacharyya, Neural response generation for task completion using conversational knowledge graph, PLOS ONE 18 (2023) e0269856. doi:10.1371/journal.pone.0269856.