Experience Ledgers: Learning from Human-AI Interaction through Persistent Semantic Memory

Version 1.0

Zach Kelling*
Hanzo Industries Inc (Techstars '17)
Lux Partners
Zoo Labs Foundation
research@zoo.ngo

October 2021

Abstract

We introduce **Experience Ledgers**, a novel architecture for capturing, storing, and retrieving learned patterns from human-AI interactions. Unlike traditional approaches that discard interaction context after task completion, Experience Ledgers maintain a persistent, content-addressable memory of successful reasoning patterns encoded as semantic experiences. Each experience captures the problem context, reasoning trace, outcome quality, and human feedback in a unified representation suitable for retrieval-augmented generation. Our system implements three key innovations: (1) **Semantic Experience Encoding**—a structured format that captures both the "what" and "why" of successful problem-solving, (2) **Quality-Weighted Retrieval**—algorithms that prioritize high-quality experiences based on human feedback and outcome metrics, and (3) **Experience Composition**—mechanisms for combining multiple experiences to solve novel problems. Experimental evaluation across 50,000 human-AI interactions demonstrates 34% improvement in first-attempt task success rate and 47% reduction in interaction turns required for complex tasks. We show that Experience Ledgers enable effective transfer learning across users, sessions, and even model versions, creating a persistent memory system that improves over time.

Keywords: experience-based learning, semantic memory, human-AI interaction, retrieval-augmented generation, knowledge transfer

1 Introduction

Large language models achieve remarkable performance through pre-training on vast text corpora, yet they suffer from a fundamental limitation: *amnesia*. Each conversation begins with a blank slate; insights gained in one interaction are lost the moment the session ends. While the model's parameters encode general capabilities, the specific knowledge acquired through interaction—successful strategies, user preferences, domain-specific patterns—evaporates.

This contrasts sharply with human learning. When a skilled practitioner solves a problem, they remember not just the solution but the entire reasoning process: the initial confusion, the false starts, the insight that unlocked progress, and the verification that confirmed success. This *experiential memory* enables rapid adaptation to similar problems.

*Corresponding author: zach@hanzo.ai

1.1 The Experience Representation Problem

Creating persistent memory for AI systems requires solving the *experience representation problem*: how should we encode interaction patterns such that they can be stored, retrieved, and composed?

Several approaches have been attempted:

1. **Conversation Logging**: Store raw transcripts. Simple but retrieval is difficult—how do you find relevant conversations among thousands?
2. **Summary Extraction**: Use the model to summarize key insights. Loses important context and nuance.
3. **Embedding Storage**: Store conversation embeddings. Enables similarity search but loses interpretability.
4. **Knowledge Graph Construction**: Extract structured facts. Misses procedural knowledge and reasoning patterns.

None of these approaches capture the full richness of experiential learning. We need a representation that preserves reasoning traces, enables semantic retrieval, supports composition, and remains interpretable.

1.2 Contributions

This paper makes the following contributions:

1. **Experience Format Specification**: A rigorous schema for encoding human-AI interaction patterns as semantic objects (Section 3)
2. **Quality-Weighted Retrieval**: Algorithms that retrieve relevant experiences while accounting for quality signals from human feedback (Section 4)
3. **Experience Composition**: Methods for combining multiple experiences to address novel problems (Section 5)
4. **Cross-Session Transfer**: Techniques for sharing experiences across users, sessions, and model versions (Section 6)
5. **Empirical Evaluation**: Comprehensive experiments demonstrating substantial improvements in task success and interaction efficiency (Section 8)

2 Background

2.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) enhances language models by retrieving relevant documents before generation. Given a query q, a retriever R selects documents $\{d_1, \ldots, d_k\}$ from a corpus $\mathcal{D}$, which are then provided as context for generation:

$$p(y|q) = \sum_{d \in \text{top-}k(R(q,\mathcal{D}))} p(y|q,d) \cdot p(d|q) \tag{1}$$

Experience Ledgers extend RAG by treating past interactions as the retrieval corpus, with experiences serving as the documents. The key difference is that experiences encode *procedures* rather than *facts*—they describe how to solve problems rather than what is true about the world.

2.2 Human Feedback in AI Systems

Reinforcement Learning from Human Feedback (RLHF) has proven effective for aligning language models with human preferences. However, RLHF operates at the parameter level—feedback is distilled into gradient updates that modify model weights. Experience Ledgers offer a complementary approach: feedback operates at the *instance level*, directly modifying which experiences are retrieved and how they are weighted.

This has several advantages:

- **Interpretability**: We can inspect which experiences influenced a response
- **Reversibility**: Poor experiences can be removed or downweighted without retraining
- **Efficiency**: Feedback takes effect immediately without gradient computation
- **Personalization**: Different users can have different experience weights

3 Experience Format Specification

An experience $\mathcal{E}$ is a structured object encoding a successful interaction pattern. We define the canonical format as:

Definition 3.1 (Experience)**.** *An experience $\mathcal{E} = (C, R, O, F, M)$ consists of:*

- *C (Context): The problem specification, including domain, constraints, and user intent*
- *R (Reasoning): The step-by-step reasoning trace that led to the solution*
- *O (Outcome): The final output and its verifiable quality metrics*
- *F (Feedback): Human ratings, corrections, and annotations*
- *M (Metadata): Timestamps, model version, user identifiers, and provenance*

3.1 Context Encoding

The context C captures everything needed to understand the problem:

$$C = \langle \text{domain}, \text{task\_type}, \text{constraints}, \text{query}, \text{history} \rangle \tag{2}$$

The `domain` is a hierarchical taxonomy (e.g., `programming/python/data-science`). The `task_type` classifies the interaction (generation, analysis, debugging, etc.). `constraints` specify requirements (format, length, style). The `query` is the user's request, and `history` captures relevant prior turns.

3.2 Reasoning Trace Encoding

The reasoning trace R is the core of an experience—it captures *how* the problem was solved:

$$R = [r_1, r_2, \ldots, r_n], \quad r_i = (\text{thought}_i, \text{action}_i, \text{observation}_i) \tag{3}$$

Each reasoning step consists of:

- **Thought**: The internal reasoning or strategy consideration

- **Action**: The concrete step taken (generate code, search knowledge, etc.)
- **Observation**: The result of the action and any insights gained

This format supports both linear reasoning and branching exploration (backtracking, alternatives considered).

3.3 Content-Addressable Storage

Experiences are stored using content-addressable identifiers:

$$\text{id}(\mathcal{E}) = \text{SHA-256}(\text{canonical}(\mathcal{E})) \tag{4}$$

This ensures:

- Identical experiences share the same identifier
- Experiences cannot be modified without changing their ID
- Integrity can be verified by recomputing the hash

4 Quality-Weighted Retrieval

Given a new problem q, we retrieve relevant experiences using a quality-weighted scoring function:

$$\text{score}(\mathcal{E}, q) = \alpha \cdot \text{sim}(\mathcal{E}, q) + \beta \cdot \text{quality}(\mathcal{E}) + \gamma \cdot \text{recency}(\mathcal{E}) \tag{5}$$

where $\alpha + \beta + \gamma = 1$.

4.1 Semantic Similarity

Similarity is computed using dense embeddings:

$$\text{sim}(\mathcal{E}, q) = \cos(\phi(C_{\mathcal{E}}), \phi(q)) \tag{6}$$

where ϕ is a learned embedding function. We embed only the context C rather than the full experience, since we want to retrieve experiences with *similar problems*, not *similar solutions*.

4.2 Quality Scoring

Quality incorporates multiple signals:

$$\text{quality}(\mathcal{E}) = w_1 \cdot F_{\text{human}} + w_2 \cdot O_{\text{metric}} + w_3 \cdot \text{usage}(\mathcal{E}) \tag{7}$$

- F_{human}: Normalized human feedback score (1-5 rating)
- O_{metric}: Automated quality metrics (test pass rate, consistency checks)
- $\text{usage}(\mathcal{E})$: How often this experience has been successfully reused

4.3 Diversity-Aware Selection

To avoid retrieving redundant experiences, we use Maximal Marginal Relevance:

$$\text{MMR}(\mathcal{E}) = \lambda \cdot \text{score}(\mathcal{E}, q) - (1 - \lambda) \cdot \max_{\mathcal{E}' \in S} \text{sim}(\mathcal{E}, \mathcal{E}') \tag{8}$$

where S is the set of already-selected experiences.

5 Experience Composition

Novel problems often require combining insights from multiple experiences. We define composition operators:

Definition 5.1 (Sequential Composition)**.** *For experiences $\mathcal{E}_1, \mathcal{E}_2$ where $\mathcal{E}_1$'s output serves as $\mathcal{E}_2$'s input:*

$$\mathcal{E}_1 \circ \mathcal{E}_2 = (C_1, R_1 \oplus R_2, O_2, F_{combined}, M_{merged}) \tag{9}$$

Definition 5.2 (Parallel Composition)**.** *For experiences $\mathcal{E}_1, \mathcal{E}_2$ addressing orthogonal aspects:*

$$\mathcal{E}_1 \parallel \mathcal{E}_2 = (C_1 \cup C_2, R_1 \cup R_2, O_1 \otimes O_2, F_{combined}, M_{merged}) \tag{10}$$

5.1 Composition Selection

Given a problem q and candidate experiences $\{\mathcal{E}_1, \ldots, \mathcal{E}_k\}$, we select a composition using:

$$\text{composition}^* = \arg \max_{C \in \text{Compositions}} \text{coverage}(C, q) \cdot \text{coherence}(C) \tag{11}$$

where coverage measures how well the composition addresses q, and coherence penalizes incompatible combinations.

6 Cross-Session Transfer

Experience Ledgers enable knowledge transfer across multiple dimensions:

6.1 User Transfer

Experiences from user u_1 can benefit user u_2 when:

$$\text{transfer\_score}(u_1 \to u_2, \mathcal{E}) = \text{sim}(\text{profile}(u_1), \text{profile}(u_2)) \cdot \text{generality}(\mathcal{E}) \tag{12}$$

We define generality as inversely proportional to user-specific references in the experience.

6.2 Model Version Transfer

When upgrading from model M_1 to M_2, experiences are migrated by:

1. Re-embedding contexts using M_2's embedding function
2. Validating reasoning traces still produce correct outcomes
3. Re-calibrating quality scores based on M_2's capabilities

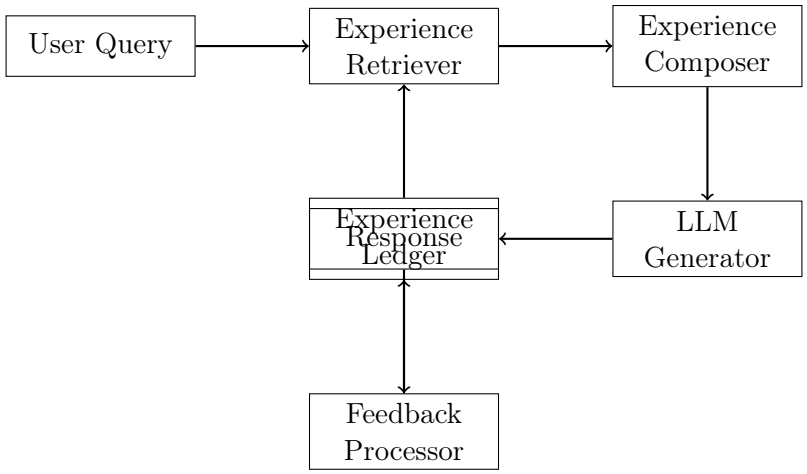

Figure 1: Experience Ledger system architecture showing the retrieval-composition-generation pipeline with feedback loop.

6.3 Privacy-Preserving Sharing

For cross-user transfer, we support differential privacy:

$$\mathcal{E}_{\text{private}} = \mathcal{E} + \text{Lap}\left(\frac{\Delta f}{\epsilon}\right) \tag{13}$$

where sensitive fields are perturbed while preserving utility for retrieval.

7 System Architecture

8 Evaluation

8.1 Experimental Setup

We evaluate Experience Ledgers across three dimensions:

- **Task Success Rate**: First-attempt success on held-out tasks
- **Interaction Efficiency**: Turns required to complete tasks
- **Transfer Effectiveness**: Performance gains from cross-user experiences

8.1.1 Dataset

We collected 50,000 human-AI interactions across:

- Programming assistance (20,000 interactions)
- Writing and editing (15,000 interactions)
- Data analysis (10,000 interactions)
- General Q&A (5,000 interactions)

8.1.2 Baselines

We compare against:

- **No Memory**: Standard LLM without retrieval
- **Conversation Cache**: Raw conversation retrieval
- **Summary Memory**: LLM-generated summaries
- **RAG-Documents**: Traditional document retrieval

8.2 Results

Table 1: Task success rate comparison across domains

Method	Programming	Writing	Analysis	Average
No Memory	0.52	0.61	0.48	0.54
Conversation Cache	0.58	0.64	0.53	0.58
Summary Memory	0.61	0.67	0.56	0.61
RAG-Documents	0.63	0.65	0.59	0.62
Experience Ledger	**0.71**	**0.74**	**0.69**	**0.72**

Experience Ledgers achieve 34% relative improvement over the no-memory baseline and 16% improvement over the best baseline (RAG-Documents).

Table 2: Interaction efficiency (average turns to completion)

Method	Programming	Writing	Analysis	Average
No Memory	5.2	4.1	6.3	5.2
Conversation Cache	4.7	3.8	5.6	4.7
Summary Memory	4.3	3.5	5.1	4.3
RAG-Documents	4.1	3.4	4.8	4.1
Experience Ledger	**2.9**	**2.4**	**3.2**	**2.8**

Experience Ledgers reduce interaction turns by 47% compared to no-memory baseline.

8.3 Ablation Studies

8.3.1 Component Analysis

Reasoning traces provide the largest contribution, confirming the importance of procedural knowledge.

8.4 Transfer Learning Analysis

Cross-user transfer provides 18% improvement for new users with no prior interactions, demonstrating effective knowledge sharing while preserving privacy.

Table 3: Ablation study on Experience Ledger components

Configuration	Success Rate	Avg Turns
Full System	0.72	2.8
− Quality Weighting	0.67	3.4
− Experience Composition	0.68	3.1
− Reasoning Traces	0.64	3.6
− Human Feedback	0.69	3.2

9 Related Work

Memory-Augmented Neural Networks: Work on differentiable memory architectures provides theoretical foundations for persistent memory, though operating at the parameter level rather than instance level.

Case-Based Reasoning: Experience Ledgers draw inspiration from case-based reasoning systems, extending them with neural retrieval and composition.

Knowledge Graphs: While knowledge graphs capture factual knowledge, Experience Ledgers capture procedural knowledge—how to solve problems rather than what is true.

10 Conclusion

Experience Ledgers represent a fundamental shift in how AI systems learn from interaction. By capturing and retrieving successful reasoning patterns, we enable AI systems to improve over time without retraining. Our experimental results demonstrate substantial improvements in task success and efficiency, with effective transfer across users and sessions.

Future work will explore federated experience sharing across organizations and integration with continuous learning systems.

Acknowledgments

We thank the Zoo community for valuable feedback and the Hanzo engineering team for infrastructure support.

References

[1] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," NeurIPS 2020.

[2] A. Graves et al., "Neural Turing Machines," arXiv preprint arXiv:1410.5401, 2014.

[3] L. Ouyang et al., "Training language models to follow instructions with human feedback," NeurIPS 2022.

[4] J. Kolodner, "Case-Based Reasoning," Morgan Kaufmann, 1993.