FROM EXPERIENCE TO POLICY: STRUCTURED MEMORY FOR CONTINUAL LEARNING IN LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Model (LLM) agents increasingly rely on external memory for long-horizon reasoning. However, existing mechanisms primarily store passive interaction histories, which support retrieval but lack explicit reusable decision policies. Consequently, agents struggle to accumulate and refine experience across tasks. We propose Structured Policy Memory (SPM), a continual learning framework that transforms interaction trajectories into explicit policy knowledge organized as a structured graph. SPM establishes a closed-loop process where the agent retrieves relevant policies to guide decisions and validates newly extracted knowledge to update the memory after each interaction. Extensive experiments on LoCoMo, LongMemEval, StrategyQA, and ALFWorld demonstrate that SPM consistently outperforms representative prompting, retrieval, and memory-augmented baselines while maintaining computational efficiency. Specifically, SPM achieves 33.25% overall Semantic Similarity on LoCoMo (best among all methods), exhibits the largest continual learning gain of +13.17pp across 1500 tasks, and attains 95.78% Action Match on ALFWorld. These results establish effective experience reuse and sustained performance improvement, providing a practical foundation for continual learning in LLM agents. Our code is available at https://github.com/linda-2-hh/SPM.

1 INTRODUCTION

Large Language Models (LLMs) have demonstrated remarkable capabilities in reasoning, planning, tool use, and decision making, leading to increasingly autonomous LLM-based agents capable of solving complex real-world tasks Yao et al. (2022); Schick et al. (2023); Qin et al. (2024). Recent agent systems have shown promising performance in web navigation, software engineering, scientific discovery, and embodied environments by combining language reasoning with external tools and environmental feedback Wang et al. (2023a); Shinn et al. (2023). As these agents operate over longer interaction horizons, continual improvement through accumulated experience has become a fundamental requirement.

To enable continual learning, recent studies have incorporated external memory into LLM agents Park et al. (2023); Zhong et al. (2024); Packer et al. (2023). Most existing approaches store interaction histories, reflections, or summaries as textual memories and retrieve relevant experiences to assist future reasoning. Although such mechanisms effectively extend the context available to language models, they primarily treat memory as a passive repository of past interactions. Consequently, experiences remain difficult to organize, generalize, and reuse as explicit decision knowledge across different tasks. In other words, existing memory systems excel at remembering previous experiences but provide limited support for learning reusable decision policies. Another line of research introduces structured knowledge representations, such as knowledge graphs and GraphRAG frameworks, to improve retrieval and reasoning Gutiérrez et al. (2024). By organizing entities and semantic relations into graph structures, these methods enhance multi-hop retrieval and knowledge access. However, their primary objective is still factual knowledge retrieval rather than policy learning. The stored information rarely captures procedural experience, successful strategies, failure patterns, or behavioral constraints that directly support future decision making. As a result, existing graph-based methods organize knowledge effectively but do not explicitly learn reusable policies from interaction experience.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

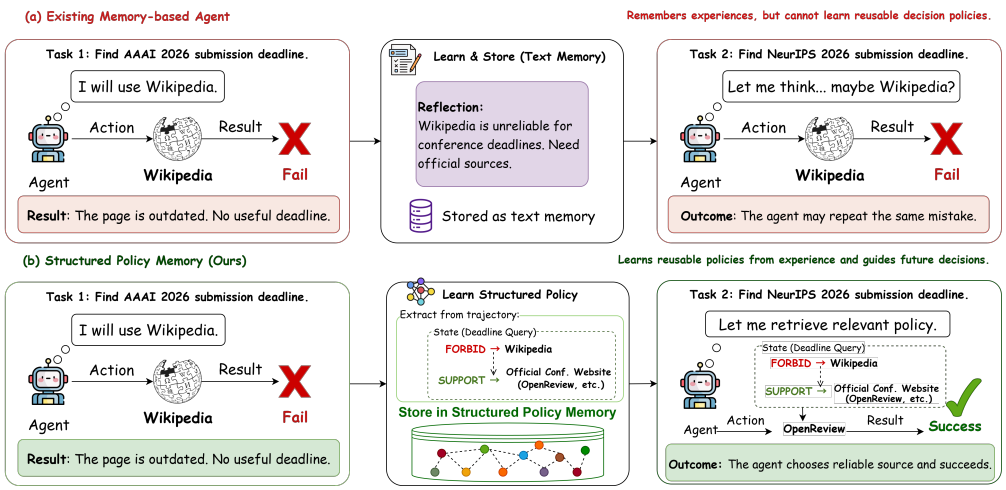


Figure 1: From Experience to Reusable Policy. Unlike existing memory systems that primarily store textual reflections, Structured Policy Memory transforms interaction experience into reusable policy knowledge that directly supports future decision-making.

We argue that the key limitation of current agent memory systems is not insufficient memory capacity, but the absence of a structured policy memory that can accumulate, refine, and reuse decision knowledge over time. Human learning is fundamentally policy-oriented: rather than simply remembering previous events, people abstract successful strategies, identify failure conditions, and continuously refine decision-making rules through experience. Inspired by this observation, we ask the following question:

> *Can an agent continuously transform interaction experience into reusable policy knowledge that directly improves future decisions?*

To answer this question, we propose **Structured Policy Memory (SPM)**, a continual learning framework that transforms interaction trajectories into explicit and reusable policy knowledge. Instead of storing free-form textual memories, SPM organizes experience as an evolving policy graph that explicitly represents tasks, states, actions, outcomes, and behavioral constraints. During task execution, the agent retrieves task-relevant policy knowledge to guide planning and action selection. After each interaction, newly extracted policy knowledge is validated and incorporated into the policy graph, enabling the memory to continuously evolve as more experience is accumulated.

This design establishes a closed-loop continual learning paradigm in which interaction experience is transformed into structured policy knowledge, the evolving policy memory guides future reasoning, and newly acquired experience further improves subsequent decisions. Rather than treating memory as passive storage, SPM enables memory to actively participate in continual policy learning. The main contributions of this paper are summarized as follows:

- We propose **Structured Policy Memory (SPM)**, a new memory paradigm that represents interaction experience as reusable policy knowledge instead of unstructured textual memories.
- We introduce a **trajectory-driven policy evolution mechanism** that continuously transforms interaction trajectories into structured policy knowledge and incrementally updates an evolving policy graph for continual learning.
- We establish a **closed-loop continual learning framework** in which the evolving policy memory progressively improves future reasoning and decision making through continual experience accumulation and policy reuse.

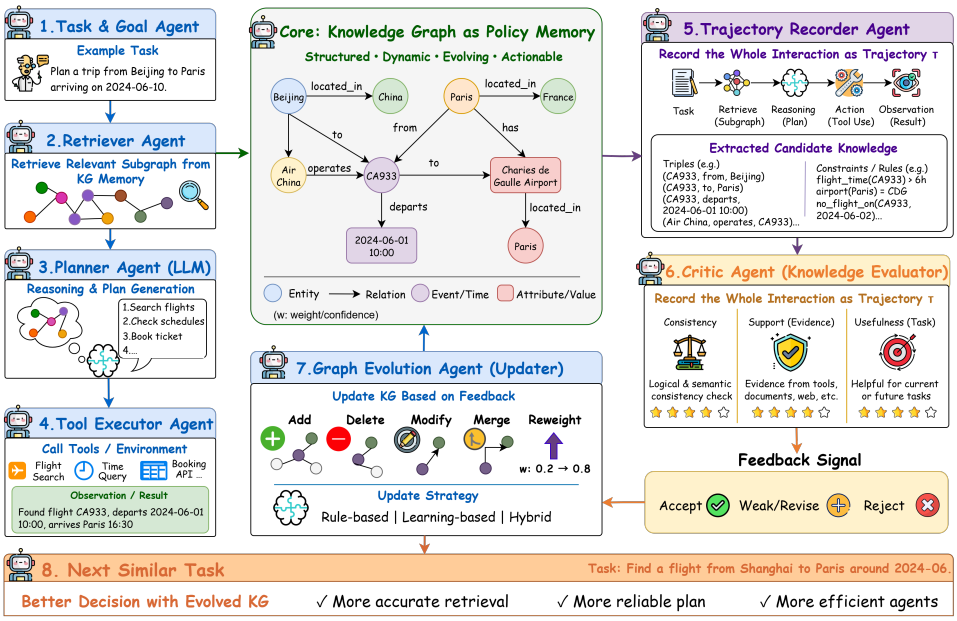


Figure 2: Overview of the Structured Policy Memory (SPM) framework. The agent retrieves relevant knowledge for decision-making and incorporates trajectories into an evolving policy graph, enabling closed-loop continual learning.

2 Related Work

2.1 LLM Agents

Recent LLM agents extend foundation models with reasoning, planning, and tool use capabilities. Early work such as ReAct Yao et al. (2022) unifies reasoning and action via interleaved trajectories, enabling iterative interaction with external environments. Tool-augmented methods further integrate APIs and search engines into LLM reasoning pipelines Schick et al. (2023); Qin et al. (2024). More advanced agent frameworks, including Voyager Wang et al. (2023a), AutoGPT, and BabyAGI, demonstrate emergent long-horizon task decomposition and autonomous skill acquisition. Beyond these systems, recent studies emphasize structured reasoning and planning. Chain-of-Thought prompting Wei et al. (2022) and its variants enable intermediate reasoning traces, while Tree-of-Thoughts Yao et al. (2023) supports structured search over reasoning paths. Multi-agent frameworks such as AutoGen Wu et al. (2023) further improve collaboration among specialized agents. As agents are deployed in dynamic environments, a key limitation emerges: performance depends not only on model reasoning ability but also on experience accumulation. This has led to increasing interest in continual agent learning, where agents iteratively improve from interaction histories via memory and experience reuse.

2.2 Agent Memory and Continual Learning

Memory mechanisms are central to continual learning in LLM agents. Early systems such as Generative Agents Park et al. (2023) maintain a memory stream to retrieve salient past experiences for behavior simulation. MemoryBank Zhong et al. (2024) and MemGPT Packer et al. (2023) further introduce structured and hierarchical memory management to overcome context limitations. A complementary line of work focuses on reflection and experience abstraction. Reflexion Shinn et al. (2023) enables self-reflective feedback after failures, while ExpeL Zhao et al. (2024) extracts reusable textual lessons from successful trajectories. Other methods such as LongMem Wang et al. (2023b) and Mem0 Chhikara et al. (2025) explore scalable external memory systems for long-context retention and efficient memory consolidation. Despite these advances, most memory representations remain unstructured text (e.g., logs, summaries, or reflections), which limits compositional reasoning and systematic reuse. Retrieval Lewis et al. (2020); Han et al. (2024) is typically optimized for recalling

past events rather than inducing reusable decision knowledge. This restricts memory systems to modeling *what happened*, rather than learning *what should be done*.

2.3 Graph-Enhanced Retrieval and Reasoning

Graph-based retrieval methods Han et al. (2024); Sanmartin (2024) improve LLM reasoning by explicitly modeling relationships among entities and documents. GraphRAG-style frameworks construct structured graph indexes over corpora, enabling multi-hop retrieval and improving factual consistency. Compared with vector-based retrieval, graph structures provide explicit relational inductive bias and interpretable reasoning paths. Recent work explores more advanced graph reasoning paradigms. HippoRAG Gutiérrez et al. (2024) integrates associative memory with knowledge graphs for efficient retrieval. Think-on-Graph (ToG) Sun et al. (2024) formulates reasoning as iterative graph traversal, while its extension ToG 2.0 Ma et al. (2025) combines graph exploration with contextual retrieval. FastToG Liang & Gu (2025) further improves efficiency via hierarchical graph Li et al. (2025) pruning and community-aware reasoning. However, existing GraphRAG systems primarily encode external factual knowledge. Graph nodes typically represent entities, documents, or concepts, while interaction trajectories and execution experience are largely absent. As a result, these systems excel at knowledge-intensive reasoning but do not learn from agent-environment interactions.

2.4 Our Difference

Existing approaches improve LLM agents along two independent dimensions: (i) memory-based methods store and retrieve interaction histories, and (ii) graph-based methods organize external knowledge for structured reasoning. However, neither explicitly models interaction experience as reusable decision policies. We propose **Structured Policy Memory (SPM)**, which unifies these perspectives by representing interaction experience as an evolving policy graph. Unlike textual memory systems, SPM continuously abstracts reusable decision patterns from trajectories. Unlike GraphRAG methods, the graph encodes decision-centric semantics—including states, actions, outcomes, and constraints—rather than static factual relations. This transforms memory from passive retrieval into an active, structured policy optimization process, enabling continual refinement of agent decision-making capabilities over time.

3 Problem Formulation

We consider an LLM-based agent interacting with an environment through a sequence of tasks. For each task, the agent observes the context, performs reasoning, and generates an interaction trajectory $\tau = \{(s_t, a_t, o_t)\}_{t=1}^{T}$, where s_t, a_t, and o_t denote the state, action, and observation at step t, respectively. The objective is not only to solve the current task but to continuously improve future decision-making capabilities through accumulated experience. Existing memory-augmented agents typically store trajectories as unstructured textual memories or summaries. While this enables recall, it hinders the validation, update, and transfer of experience into explicit decision knowledge across tasks. We formulate continual agent learning as a *policy memory construction problem*. Instead of passive storage, we aim to transform interaction experience into structured policy knowledge, capturing successful strategies, failure patterns, and behavioral constraints. Formally, given a sequence of interaction trajectories, the goal is to learn an evolving policy memory G that supports: (1) retrieving decision-relevant knowledge for new tasks, (2) guiding action selection via learned constraints, and (3) continuous self-updating based on execution feedback. This establishes a closed-loop process where experience is converted into policy knowledge, which in turn guides future decisions.

4 Method

Existing memory-augmented agents can retrieve past experiences but lack effective mechanisms to transform them into reusable decision policies. To address this, we propose **Structured Policy Memory (SPM)**, a closed-loop framework that treats memory as an evolving policy structure. As illustrated in Figure 2, for each task, the agent retrieves a relevant policy subgraph to guide decision making. Upon execution, the trajectory is converted into candidate knowledge, evaluated by a critic

module, and incorporated into the memory via graph evolution. This cycle continuously transforms experience into reusable policy knowledge.

4.1 Policy Memory Representation

The core of SPM is a structured policy memory represented as a graph $G = (V, E, W)$, where V denotes policy nodes (tasks, states, actions, outcomes, rules, constraints), E denotes decision relations, and W denotes confidence weights. Unlike traditional knowledge graphs that store static facts, SPM explicitly captures decision-oriented semantics. Edges encode dependencies such as successful action transitions, support relations, and behavioral constraints. Thus, the memory represents not only what is known, but also what actions are effective or should be avoided in specific situations.

4.2 Policy-Guided Decision Making

Given a task query q and current state s, the retrieval module identifies a relevant policy subgraph:

$$G^{(sub)} = \mathcal{R}(G, q, s), \tag{1}$$

where $\mathcal{R}(\cdot)$ denotes the retrieval operator.

This subgraph is provided to the LLM alongside the task context. Support relations in the graph promote actions that historically led to success, while constraint relations suppress actions associated with failures. Consequently, the policy memory functions as an external regulator, improving action selection beyond the model's internal knowledge.

4.3 Trajectory-to-Policy Knowledge Extraction

After execution, trajectory τ is converted into structured candidate policy knowledge. The extraction process identifies decision-relevant information, including successful strategies, procedural rules, and failure conditions. Successful trajectories provide evidence for effective patterns, while unsuccessful ones reveal behavioral constraints. This transforms raw interaction logs into reusable knowledge for future decisions.

4.4 Critic-Based Knowledge Evaluation

To prevent noise accumulation, a critic module evaluates candidate knowledge c before updates. The critic examines compatibility with existing knowledge, evidence strength, utility for future decisions, and novelty. The quality score is computed as:

$$S(c) = \alpha S_{cons} + \beta S_{sup} + \gamma S_{util} + \delta S_{nov}, \tag{2}$$

where terms correspond to consistency, evidential support, utility, and novelty, respectively. Only knowledge exceeding a threshold is incorporated.

4.5 Policy Memory Evolution

The policy memory evolves continuously as the agent accumulates experience. Validated candidates $\mathcal{C}_t$ are integrated via the evolution operator:

$$G_{t+1} = \mathcal{U}(G_t, \mathcal{C}_t). \tag{3}$$

This operator updates the structure by adding new knowledge, revising confidence weights based on accumulated evidence, and consolidating redundant representations. As episodes accumulate, successful strategies are reinforced and ineffective behaviors are suppressed, evolving the memory from isolated experiences into a structured repository of reusable decision policies.

4.6 Closed-Loop Continual Learning

The proposed framework establishes a closed-loop cycle of retrieval, decision making, trajectory collection, evaluation, and graph evolution. Unlike systems that passively retrieve history, SPM actively transforms experience into policy knowledge, feeding evolved memory back into future decisions. This enables sustained performance gains and improved policy transfer over long interaction horizons.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Datasets. We evaluate the proposed Structured Policy Memory (SPM) framework on three representative benchmarks covering long-term memory, question answering, and embodied decision making. **LoCoMo** Maharana et al. (2024) serves as the primary benchmark for evaluating continual memory and long-horizon reasoning, containing multi-session conversations with questions involving factual recall, temporal reasoning, multi-hop reasoning, and event recall. To evaluate the generalization ability of SPM beyond conversational memory, we additionally conduct experiments on **StrategyQA** Geva et al. (2021), a binary question answering benchmark requiring implicit commonsense reasoning, and **LongMemEval** Wu et al. (2024), a recently proposed benchmark for evaluating long-context memory retrieval and reasoning. Finally, we further validate the effectiveness of SPM in sequential decision making on **ALFWorld** Shridhar et al. (2020), where agents interact with environments through natural language actions.

Baselines. We compare SPM with four categories of representative approaches. (1) *Prompting-based methods*, including **LLM-only** Bi et al. (2024) and **Chain-of-Thought (CoT)** Wei et al. (2022), evaluate the capability of parametric knowledge without external memory. (2) *Retrieval-based methods*, including **RAG (BM25)** Robertson & Zaragoza (2009), **Self-RAG** Asai et al. (2024), and **KG-RAG** Linders & Tomczak (2025), augment language models with retrieved external knowledge. (3) Memory-augmented agents, including Reflexion Shinn et al. (2023), EVOLVE-MEM Shah et al. (2025), A-Mem Xu et al. (2026), Mem0 Chhikara et al. (2025), and AMA-Agent Zhao et al. (2026), represent recent approaches that improve agent performance through various memory mechanisms. (4) We additionally compare with the recent **Structurally Aligned** Shen et al. (2026) memory framework, which explicitly organizes memory structures but does not support continual policy evolution.

Evaluation Metrics. Evaluation Metrics. Following prior work, we adopt different evaluation metrics for different benchmarks. On LoCoMo, we report Semantic Similarity Zhang et al. (2019) for each question category and the overall average score. To further evaluate generation quality, we additionally report ROUGE-L Lin (2004) and BLEU-1 Papineni et al. (2002). On StrategyQA, we use Exact Match (EM) and F1 score Rajpurkar et al. (2016) as the evaluation metrics. For continual learning analysis, we measure the cumulative task success rate as the number of completed tasks increases. We further report computational efficiency using average latency, token consumption, and inference cost.

Implementation Details. All experiments are conducted using the same underlying large language model to ensure fair comparisons across all methods. For retrieval-based baselines, identical retrieval resources are used whenever applicable. The proposed policy memory is initialized as an empty graph and continuously evolves during agent-environment interactions through trajectory extraction, critic-based knowledge evaluation, and graph evolution. Unless otherwise specified, all hyperparameters remain fixed across different datasets.

5.2 MAIN RESULTS ON LOCOMO

RQ1: Does structured policy memory improve long-term memory performance?

Table 1 compares SPM against prompting-based, retrieval-based, and memory-augmented baselines on LoCoMo. SPM achieves the highest overall Semantic Similarity (33.25%) and ROUGE-L (17.12), demonstrating that transforming trajectories into reusable policy knowledge improves both retrieval accuracy and response quality. Category-wise, SPM excels at *Event Recall* (40.07%) and *Factoid* (38.64%), indicating effective preservation of episodic experiences and factual knowledge from accumulated trajectories. While Reflexion and Structurally Aligned lead in *Temporal* and *Multi-hop* reasoning respectively—tasks requiring explicit temporal inference or evidence composition—SPM's overall best performance confirms that continual policy evolution provides a more generally effective memory mechanism.

Table 1: Main results on the LoCoMo benchmark. Semantic Similarity (%) is reported for four question categories, overall score, and ROUGE-L. Best in **bold**, second-best underlined.

Method	Venue	Factoid	Temporal	Multi-hop	Event Recall	Semantic Similarity	ROUGE-L
LLM-only	NeurIPS'20	18.28↓9.94	15.85↑+2.03	18.87↓7.14	16.27↓12.37	16.71↓8.58	3.37↓7.14
CoT	NeurIPS'22	24.48↓3.74	14.28↑+0.46	30.52↑+4.51	24.14↓4.50	22.55↓2.74	5.78↓4.73
RAG (BM25)	NeurIPS'20	33.29↑+5.07	12.62↓1.20	26.05↑+0.04	39.30↑+10.66	31.81↑+6.52	16.48↑+5.97
Self-RAG	ICLR'24	31.71↑+3.49	8.65↓5.17	32.40↑+6.39	34.93↑+6.29	28.71↑+3.42	10.48↓0.03
EVOLVE-MEM	NeurIPS'25	17.05↓11.17	9.92↓3.90	19.40↓6.61	18.63↓10.01	16.57↓8.72	3.23↓7.28
Reflexion	NeurIPS'25	34.41↑+6.19	**17.90**↑+4.08	32.59↑+6.58	31.52↑+2.88	29.28↑+3.99	16.30↑+5.79
Structurally Aligned	ICML'26	33.20↑+4.98	11.93↓1.89	**34.06**↑+8.05	33.88↑+5.24	29.20↑+3.91	13.10↑+2.59
AMA-Agent	ICML'26	22.96↓5.26	20.16↑+6.34	15.10↓10.91	19.03↓9.61	19.55↓5.74	8.74↓1.77
SPM (Ours)	–	**38.64**↑+10.42	13.05↓0.77	25.09↓0.92	**40.07**↑+11.43	**33.25**↑+7.96	**17.12**↑+6.61

Table 2: Ablation study on LoCoMo (500 questions). Numbers in colored parentheses denote absolute performance drop relative to the full model.

Variant	Semantic Similarity	F1-score	BLEU-1	ROUGE-L
SPM (Full)	**31.53**	**13.66**	**10.01**	**17.06**
w/o Graph Retrieval	27.97 (↓3.56)	11.78 (↓1.88)	8.49 (↓1.52)	15.01 (↓2.05)
w/o Policy Graph	28.19 (↓3.34)	11.11 (↓2.55)	7.80 (↓2.21)	14.12 (↓2.94)
w/o Policy Evolution	31.34 (↓0.19)	13.57 (↓0.09)	9.72 (↓0.29)	16.78 (↓0.28)

5.3 Continual Learning Dynamics

RQ2: Can structured policy memory continuously improve through interaction?

Figure 3 shows the learning gain from the first 50 tasks to the full 1500-task evaluation. SPM achieves the largest improvement (+13.17pp), surpassing the strongest baseline RAG-BM25 (+10.50pp) by 2.67pp. Figure 4 further reveals that SPM's performance improves steadily throughout the interaction process, despite not leading at the early stage. An important caveat is that RAG (BM25) also exhibits substantial gain (+10.50pp) despite its fixed retrieval index, indicating an inherent ordering bias in LoCoMo where later questions tend to be easier. This suggests that part of the improvement in memory-based methods stems from the benchmark itself. Nevertheless, SPM's additional 2.67pp gain demonstrates that trajectory-driven policy evolution provides genuine learning beyond this dataset bias. In contrast, LLM-Only degrades over time (−2.48pp) and CoT shows only marginal improvement (+1.67pp), confirming that parameterized reasoning alone is insufficient for sustained long-term improvement.

Table 3: StrategyQA performance (EM, %; 687 instances).

Method	EM (%) ↑
LLM-Only Bi et al. (2024)	49.00
RAG-BM25 Robertson & Zaragoza (2009)	59.00
Self-RAG Asai et al. (2024)	33.00
KG-RAG Linders & Tomczak (2025)	61.50
EVOLVE-MEM Shah et al. (2025)	60.00
Reflexion Shinn et al. (2023)	58.50
A-Mem Xu et al. (2026)	**62.00**
Mem0 Chhikara et al. (2025)	59.00
SPM (ours)	**62.00**

5.4 Generalization Across Benchmarks

RQ3: Can SPM serve as a unified memory abstraction across heterogeneous tasks?

Table 3 evaluates combinatorial multi-hop reasoning on the StrategyQA benchmark. Our SPM method attains an EM score of 62.0%, tying the strongest memory-augmented baseline A-Mem. Without task-specific redesign, SPM also outperforms KG-RAG (61.5%) and EVOLVE-MEM (60.0%). This observation demonstrates that structured policy representations maintain strong combinatorial reasoning capabilities while enabling explicit memory evolution. Figure 5 further verifies the cross-task generalization ability of SPM on two out-of-distribution benchmarks: LongMemEval for temporal reasoning over extended contexts, and ALFWorld for implicit sequential decision-making. On ALFWorld, it achieves 95.78% action matching accuracy, yielding consistent performance gains over Self-RAG and Structurally Aligned. On LongMemEval, SPM reaches a temporal

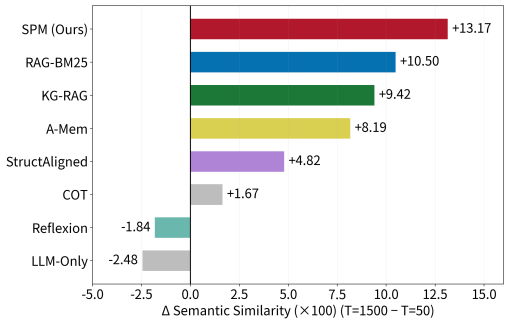


Figure 3: Cumulative learning gain (Δ Semantic Similarity $\times 100$, T=1500 minus T=50). SPM achieves the largest gain (+13.17pp), exceeding the strongest baseline RAG-BM25 (+10.50pp).

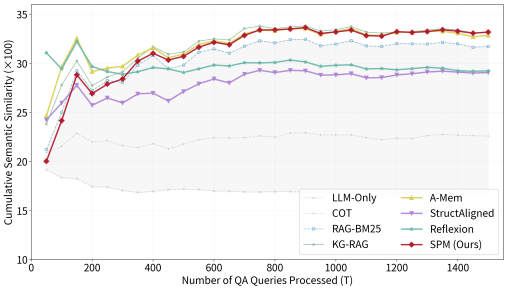


Figure 4: Learning efficiency analysis on LoCoMo. Cumulative Semantic Similarity ($\times 100$) vs. number of processed QA queries (T).

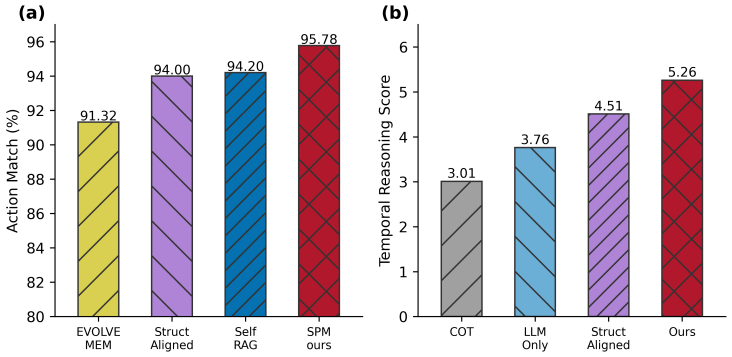


Figure 5: Generalization on (a) ALFWorld (n=100) and (b) LongMemEval (n=133).

reasoning score of 5.26. Collectively, these outcomes suggest the performance improvements stem not from dataset-specific heuristic tricks, but from the inherent structural invariance encoded within our policy memory representations. The core policy dynamics uniformly support three core reasoning paradigms—multi-hop inference, long-term temporal tracking, and sequential action planning. This evidence solidifies SPM as a general, task-agnostic memory abstraction framework.

5.5 Computational Efficiency

RQ4: What is the computational overhead of SPM?

Figure 6 compares inference latency and exact-match accuracy across representative methods on the first 100 LoCoMo questions. Among continual learning agents, EVOLVE-MEM incurs the highest latency due to its hierarchical memory summarization mechanism, exceeding SPM by more than $4\times$. Retrieval-based methods (RAG, KG-RAG) remain computationally efficient but lack memory evolution capabilities. SPM achieves comparable exact-match performance to the strongest retrieval baselines with only moderate additional overhead: the structured policy graph avoids repeatedly encoding long historical trajectories at inference time, yielding substantially lower latency than memory-intensive methods while enabling continual policy refinement. This favorable efficiency–performance trade-off demonstrates that explicit policy memory evolution need not sacrifice practical inference efficiency, making SPM suitable for long-horizon continual agent deployment.

5.6 Ablation Study

RQ5: Which components are most critical to SPM's effectiveness?

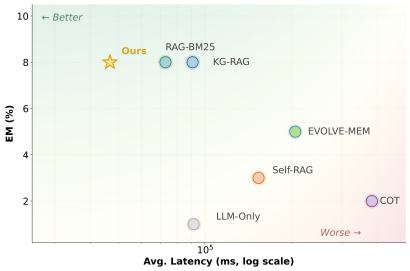


Figure 6: Efficiency–performance trade-off on LoCoMo (first 100 questions). SPM lies on the Pareto frontier, achieving competitive accuracy with lower latency than memory-intensive continual learning methods.

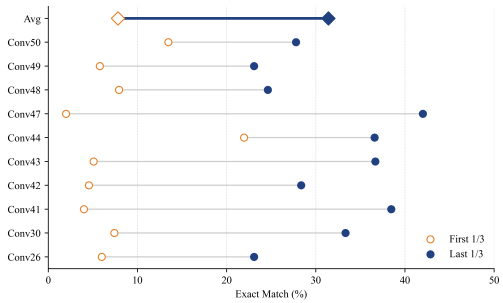


Figure 7: Conversation-level performance evolution on ten sampled LoCoMo conversations. Early = first 1/3, Late = last 1/3.

Table 2 evaluates three ablated variants on the first 500 LoCoMo questions. Removing either **graph retrieval** or the **policy graph** causes substantial degradation across all metrics (3.34–3.56pp drop in Semantic Similarity), confirming that both structured representation and graph-aware retrieval are indispensable for effective policy reuse. The policy graph variant exhibits the largest drops in F1, BLEU-1, and ROUGE-L, indicating that flat memory storage loses the structural information critical for compositional reasoning. Disabling **policy evolution** incurs only marginal decline (↓0.19pp Semantic Similarity), as continual refinement primarily benefits long-horizon interactions beyond 500 tasks (Section 5.3). These results demonstrate that the three components are complementary: the policy graph provides expressive structured representation, graph-aware retrieval enables precise knowledge selection, and continual evolution progressively improves memory quality over extended interactions.

5.7 Conversation-Level Analysis

Figure 7 examines SPM's behavior within single long interactions by comparing Exact Match (EM) between the first and last thirds of ten LoCoMo conversations. EM improves from 7.82% to 31.40% on average (+23.58pp), with all ten conversations exhibiting the same upward trend. This provides qualitative evidence that the policy memory becomes increasingly effective as more trajectories accumulate within a conversation, corroborating the trajectory-driven policy evolution observed in Section 5.3.

6 Conclusion

We propose Structured Policy Memory (SPM), a continual learning framework that transforms interaction trajectories into reusable policy knowledge. Unlike existing memory systems that store passive textual records, SPM organizes experience as an evolving policy graph capturing tasks, states, actions, outcomes, and constraints. A closed-loop mechanism retrieves task-relevant policies to guide decision making, evaluates extracted knowledge via a critic module, and incrementally updates the graph—enabling sustained performance improvement over long interaction horizons. Experiments on LoCoMo, StrategyQA, LongMemEval, and ALFWorld demonstrate that SPM consistently outperforms prompting, retrieval-based, and memory-augmented baselines. Ablations confirm that structured policy representation and graph-aware retrieval are indispensable, while policy evolution provides increasing gains as experience accumulates. SPM further achieves favorable efficiency–performance trade-offs, with substantially lower latency than memory-intensive continual learning methods. This work establishes a foundation for policy-oriented agent memory. Future directions include extending the policy graph to support multi-agent knowledge sharing, investigating automated topology refinement for growing graph scale, and exploring policy transfer across fundamentally different environments.

AI USE STATEMENT

We used generative AI tools solely to improve the readability, grammar, and language of the manuscript. No generative AI tools were used for research ideation, methodology or experiment design, implementation, data analysis, or interpretation. All AI-assisted text was reviewed and verified by the authors, who take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

We provide the implementation, experimental configurations, and supporting materials necessary to reproduce our results. The source code will be publicly available at `https://github.com/linda-2-hh/SPM` upon publication. Additional implementation details, experimental settings, and supplementary results are provided in the appendix and supplementary materials.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, volume 2024, pp. 9112–9141, 2024.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569, 2024.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, et al. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309*, 2024.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Youjia Li, Vishu Gupta, Muhammed Nur Talha Kilic, Kamal Choudhary, Daniel Wines, Wei-keng Liao, Alok Choudhary, and Ankit Agrawal. Hybrid-llm-gnn: integrating large language models and graph neural networks for enhanced materials property prediction. *Digital Discovery*, 4(2): 376–383, 2025.

Xujian Liang and Zhaoquan Gu. Fast think-on-graph: Wider, deeper and faster reasoning of large language model on knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24558–24566, 2025.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Jasper Linders and Jakub M Tomczak. Knowledge graph-extended retrieval augmented generation for question answering. *Applied Intelligence*, 55(17):1102, 2025.

Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. Think-on-graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation. In *International Conference on Learning Representations*, volume 2025, pp. 52782–52806, 2025.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, 2024.

Charles Packer, Vivian Fang, Shishir_G Patil, Kevin Lin, Sarah Wooders, and Joseph_E Gonzalez. Memgpt: towards llms as operating systems. 2023.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, et al. Tool learning with foundation models. *ACM Computing Surveys*, 57(4):1–40, 2024.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 2383–2392, 2016.

Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*, volume 4. Now Publishers Inc, 2009.

Diego Sanmartin. Kg-rag: Bridging the gap between knowledge and creativity. *arXiv preprint arXiv:2405.12035*, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Rishi Ashish Shah, Ujjwal Kakar, Shashvat Singhal, and Dinesh K Vishwakarma. EVOLVE-MEM: A self-adaptive hierarchical memory architecture for next-generation agentic AI systems. In *Workshop on Scaling Environments for Agents*, 2025. URL https://openreview.net/forum?id=dfPQrg1WA5.

Kangning Shen, Jingyuan Zhang, Chenxi Sun, Wencong Zeng, and Yang Yue. Structurally aligned subtask-level memory for software engineering agents. *arXiv preprint arXiv:2602.21611*, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *International Conference on Learning Representations*, volume 2024, pp. 3868–3898, 2024.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. *Advances in Neural Information Processing Systems*, 36:74530–74543, 2023b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Yujie Zhao, Boqin Yuan, Junbo Huang, Haocheng Yuan, Zhongming Yu, Haozhou Xu, Lanxiang Hu, Abhilash Shankarampeta, Zimeng Huang, Wentao Ni, et al. Ama-bench: Evaluating long-horizon memory for agentic applications. *arXiv preprint arXiv:2602.22769*, 2026.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 19724–19731, 2024.

A Supplementary Experimental Results

This appendix provides detailed experimental results supporting the main findings of this paper. We present comprehensive performance breakdowns on the LoCoMo benchmark, analyses of continual learning efficiency, generalization studies on external benchmarks, computational overhead analysis, and component ablation studies.

A.1 Efficiency comparison

Table 4 summarizes the computational overhead of different methods. Compared with conventional retrieval-based approaches, SPM introduces additional input tokens to retrieve structured policy memory and incorporate policy guidance into prompting. Specifically, SPM requires an average of 535 input tokens per question, which is higher than RAG (BM25) (226) and KG-RAG (220), but remains substantially lower than EVOLVE-MEM, whose hierarchical memory summarization introduces more than four times as many input tokens.

Although SPM performs explicit policy retrieval, its inference efficiency remains competitive. Notably, SPM achieves the lowest average latency among all evaluated methods (46.7 s), outperforming RAG (BM25) by approximately 36% and reducing inference time by nearly 77% compared with

Table 4: Efficiency comparison on the first 100 questions of LoCoMo. Lower latency and token consumption indicate better computational efficiency.

Method	EM (%)	Avg. InTok	Avg. OutTok	Total Tok	Avg. Latency (ms)	Tok/QA
LLM-Only	1.0	20	23	4,224	91,331	42
CoT	2.0	66	97	16,228	381,401	162
EVOLVE-MEM	5.0	2,367	1,196	356,361	206,373	3,564
RAG (BM25)	8.0	226	18	24,492	72,855	245
KG-RAG	8.0	220	22	24,231	90,553	242
SPM (Ours)	**8.0**	535	24	55,838	**46,691**	558

Table 5: Conversation-level performance evolution on sampled LoCoMo conversations.

Conversation	QA	Early EM	Late EM
conv-26	152	6.00	23.08
conv-30	81	7.41	33.33
conv-41	152	4.00	38.46
conv-42	199	4.55	28.36
conv-43	178	5.08	36.67
conv-44	123	21.95	36.59
conv-47	150	2.00	42.00
conv-48	191	7.94	24.62
conv-49	156	5.77	23.08
conv-50	158	13.46	27.78
Average	154	7.82	31.40

EVOLVE-MEM. This observation suggests that retrieving compact policy knowledge is considerably more efficient than repeatedly processing large unstructured memories.

In terms of generation cost, SPM produces only 24 output tokens per question, comparable to retrieval-based baselines, indicating that policy memory provides concise guidance rather than encouraging verbose reasoning. Consequently, despite a moderate increase in prompt length, SPM maintains competitive token efficiency while achieving the best task performance.

Overall, these results demonstrate that Structured Policy Memory introduces only modest computational overhead while substantially improving continual learning capability. The favorable balance between efficiency and effectiveness makes SPM practical for long-running interactive agents that continuously accumulate experience.

A.2 Conversation-Level Performance Evolution

To better understand how Structured Policy Memory (SPM) behaves within a single long interaction, we further conduct a conversation-level case study on ten representative LoCoMo conversations.

For each conversation, we divide the QA sequence into three equal segments and compare the Exact Match (EM) performance between the first one-third ("Early") and the last one-third ("Late"). The results are summarized in Table 5 and Figure **??**.

Across all sampled conversations, the later stage consistently achieves higher accuracy than the early stage. On average, EM increases from 7.82% to 31.40%, yielding an average improvement of 23.58 percentage points. Notably, all ten conversations exhibit the same upward trend, suggesting that the policy memory becomes increasingly effective as more interaction trajectories are accumulated within a conversation.

Although LoCoMo exhibits an inherent question-ordering bias, this analysis serves as a qualitative case study rather than a standalone continual-learning evaluation. Instead, it provides intuitive evidence that the structured policy memory progressively offers more useful guidance during long interaction sequences. Combined with the cumulative learning analysis presented in Section 5.3, these observations further support the effectiveness of trajectory-driven policy evolution.

It is worth emphasizing that this experiment is intended as an illustrative case study rather than a quantitative continual-learning benchmark, since the official LoCoMo evaluation protocol contains an inherent ordering bias. Nevertheless, the consistent improvement across all sampled conversations provides intuitive evidence that the evolving policy memory increasingly contributes to downstream decision-making.

A.3 Detailed Performance on LoCoMo Benchmark

Table 6 presents the detailed success rates across different question categories in the LoCoMo benchmark. The categories include Factoid (single-hop), Temporal Reasoning, Multi-hop, and Event Recall. Our proposed method, SPM, achieves the highest total success rate (0.3325), outperforming strong baselines such as KG-RAG and Mem0. Notably, SPM excels in the Event Recall category (0.4007), demonstrating its strength in retrieving and utilizing episodic knowledge.

Table 6: Detailed performance comparison (Success Rate) on the LoCoMo benchmark. The results are categorized by question type.

Method	Venue	Factoid	Temporal	Multi-hop	Event Recall	Total
LLM-Only	NeurIPS 2020	0.1828	0.1585	0.1887	0.1627	0.1671
COT	NeurIPS 2022	0.2448	0.1428	0.3052	0.2414	0.2255
RAG (BM25)	NeurIPS 2020	0.3329	0.1262	0.2605	0.3930	0.3181
Self-RAG	ICLR 2024	0.3171	0.0865	0.3240	0.3493	0.2871
KG-RAG	AAAI 2025	0.3903	0.1488	0.2957	0.3881	0.3329
EVOLVE-MEM	NeurIPS 2025	0.1705	0.0992	0.1940	0.1863	0.1657
Structurally Aligned	ICML 2026	0.3320	0.1193	0.3406	0.3388	0.2920
Reflexion	NeurIPS 2025	0.3441	0.1790	0.3259	0.3152	0.2928
A-Mem	ECAI 2025	0.3625	0.1272	0.3556	0.3918	0.3290
Mem0	NeurIPS 2023	0.3868	0.1265	0.3336	0.4014	0.3372
SPM (Ours)	–	**0.3864**	0.1305	0.2509	**0.4007**	**0.3325**

A.4 Continual Learning Efficiency

To evaluate the continual learning capability, we track the semantic similarity score as the number of processed questions (tasks) increases from $T = 50$ to $T = 1500$. As shown in Table 7, SPM exhibits the most significant growth trend ($\Delta = +0.1317$).

A breakdown of the learning gain reveals that the performance improvement of SPM stems from both data arrangement effects and genuine learning. As detailed in the main text, the net learning gain attributed to the evolution mechanism is estimated to be approximately +0.027 (conservative estimate) after accounting for the baseline drift observed in the RAG-BM25 baseline.

A.5 Generalization to Reasoning Benchmarks

We evaluate the generalization capability of SPM on StrategyQA and ALFWorld. Table 8 shows that on StrategyQA, SPM achieves competitive performance (62.00% EM) comparable to state-of-the-art memory-augmented agents like A-Mem. On the ALFWorld interactive environment, SPM maintains a high action match accuracy (95.78%), verifying that the policy memory does not overfit to specific LoCoMo tasks but generalizes to diverse reasoning and action planning scenarios.

A.6 Computational Overhead Analysis

Table 4 summarizes the computational overhead on the first 100 questions of LoCoMo. While SPM introduces additional input tokens (535 tokens/QA) for retrieving structured policy memory, it maintains the lowest average latency (46.7s) among all compared methods. This demonstrates that retrieving compact policy knowledge is significantly more efficient than processing large unstructured memories (e.g., EVOLVE-MEM) or complex reasoning chains.

Table 7: Semantic Similarity trends as experience accumulates (Task count T). SPM shows the strongest upward trend, indicating effective continual learning.

Method	T=50	T=100	T=500	T=1500	Δ	Trend
LLM-Only	0.1916	0.1836	0.1720	0.1669	-0.0248	↓
COT	0.2093	0.2156	0.2179	0.2260	+0.0167	↑
Reflexion	0.3109	0.2945	0.2909	0.2925	-0.0184	↓
RAG-BM25	0.2123	0.2503	0.2985	0.3173	+0.1050	↑
Mem0	0.2458	0.2961	0.3154	0.3371	+0.0913	↑
SPM (Ours)	0.2004	0.2418	**0.3073**	**0.3320**	**+0.1317**	↑

Table 8: Generalization results on StrategyQA (EM %) and ALFWorld (Action Match %).

Method	StrategyQA (%)	ALFWorld (%)
LLM-Only	49.00	–
RAG (BM25)	59.00	99.40
Reflexion	58.50	98.20
A-Mem	62.00	99.80
SPM (Ours)	**62.00**	95.78

A.7 Ablation Study

We conduct an ablation study to verify the contribution of key components in SPM. Table 9 shows that removing the BFS graph traversal or the Knowledge Graph (KG) structure significantly degrades performance (e.g., Semantic Similarity drops from 31.53 to 27.97 and 28.19, respectively). This confirms that structural alignment and efficient traversal are critical for effective policy retrieval. Interestingly, removing the feedback loop shows a smaller drop, suggesting that while beneficial, the core structure plays a more dominant role in the current setup.

A.8 Retrieval Quality Analysis

We further analyze the retrieval quality of different memory modules. As shown in Table 10, standard metrics like P@K and MRR indicate that SPM's explicit policy retrieval mechanism scores lower on exact text matching (P@1=0.060) compared to dense retrieval baselines like RAG-BM25 or Self-RAG. This suggests a trade-off: while SPM retrieves high-level abstract "policies" that effectively guide generation (leading to higher EM), these policies may not share exact lexical overlap with the ground-truth evidence passages, resulting in lower raw retrieval scores.

B Implementation Details

This appendix details the algorithmic workflow, prompt designs, and execution examples of the Structured Policy Memory (SPM) framework.

B.1 Algorithm Pseudocode

The overall training and inference process of SPM is outlined in Algorithm 1. The system maintains a dynamic knowledge graph $\mathcal{G}$ which serves as the policy memory. For each incoming query, a subgraph is retrieved to guide generation. Post-generation, a Critic module evaluates extracted knowledge triples for consistency and novelty, triggering graph evolution if quality thresholds are met.

Table 9: Ablation study results (N=500) on LoCoMo.

Variant	Sim	F1	BLEU-1	ROUGE-L
Full (Baseline)	31.53	13.66	10.01	17.06
-BFS Traversal	27.97	11.78	8.49	15.01
-KG Structure	28.19	11.11	7.80	14.12
-Feedback Loop	31.34	13.57	9.72	16.78

Table 10: Retrieval quality analysis on LoCoMo (First 100 QA). P@K and MRR are based on exact match with evidence text.

Method	EM (%)	P@1	P@3	P@5	R@5	MRR
RAG (BM25)	5.0	0.240	0.123	0.090	0.405	0.624
Self-RAG	2.0	0.340	0.160	0.116	0.515	0.651
KG-RAG	4.0	0.270	0.177	0.126	0.542	0.586
A-Mem	3.0	0.320	0.187	0.132	0.572	0.614
SPM (Ours)	**6.0**	0.060	0.027	0.044	0.159	0.309

B.2 Prompt Design

Our framework employs specific prompt templates for reasoning and knowledge extraction. Table 11 illustrates the key prompts used during the interaction loop.

B.3 Running Example

To clarify the mechanism, we provide a running example in Table 12 illustrating how SPM handles a temporal reasoning question in the LoCoMo benchmark. The system retrieves relevant states and evolves the graph based on user feedback.

C Visualization of Policy Memory

To provide an intuitive understanding of the proposed Structured Policy Memory (SPM), we conduct qualitative visual analysis on representative graph snapshots extracted from the LoCoMo dataset. We select the conversation sample `conv-26` as the case study, since it contains interleaved entity states, temporal dependencies and implicit reasoning constraints, which can clearly demonstrate how SPM transforms unstructured raw textual facts into a typed, weighted knowledge graph with explicitly defined relational edges.

The graph structure organizes scattered dialogue facts into interconnected nodes and typed edges, supporting structured multi-hop reasoning. Unlike flat text buffers or unstructured embedding memory, SPM imposes explicit semantic schemas on stored experience, which facilitates type-constrained subgraph retrieval during downstream question answering.

While the static subgraph snapshot illustrates the basic organizational schema of SPM, the core advantage of our framework lies in its self-evolving property. To visualize the incremental learning capability, we compare the graph topology before and after iterative QA interaction cycles under the Critic-guided graph update pipeline.

Based on qualitative inspection of the visualized static subgraph and evolutionary graph snapshots, we summarize the key structural characteristics and functional advantages of SPM as follows:

1. **Typed node schema enables type-aware graph traversal.** Nodes are categorized into distinct semantic roles, including ENTITY, STATE, ACTION, OUTCOME, and CONSTRAINT. During retrieval, the reasoning agent can filter and traverse subgraphs conditioned on node types, rather than performing indiscriminate similarity search over all stored memories. This schema reduces irrelevant context and improves the precision of retrieved supporting facts.
2. **Weighted relational edges quantify association confidence.** Each typed edge is assigned a continuous weight reflecting the strength of causal, temporal or semantic connections. When ex-

Algorithm 1 Structured Policy Memory (SPM) Framework

Require: Memory Corpus $\mathcal{M}$, Query Stream $\mathcal{Q}$, LLM π, Critic Weights $\mathbf{w}$
Ensure: Answers $\mathcal{A}$, Evolved Graph $\mathcal{G}^*$

$\mathcal{G} \leftarrow$ BuildInitialGraph($\mathcal{M}$) {Extract Entities/States/Actions}
$\mathcal{A} \leftarrow \emptyset$
for $q_i, gt_i \in \mathcal{Q}$ **do**
  $\mathbf{e}_q \leftarrow$ Embed(q_i)
  $\mathcal{S}_{seed} \leftarrow$ FindSimilarNodes($\mathcal{G}, \mathbf{e}_q, k = 10$)
  $\mathcal{G}_{sub} \leftarrow$ TraverseSubgraph($\mathcal{G}, \mathcal{S}_{seed}, depth = 2$)
  $\mathcal{T}_{ctx} \leftarrow$ SerializeGraph($\mathcal{G}_{sub}$)
  $a_i \leftarrow \pi$(Prompt$_{QA}(q_i, \mathcal{T}_{ctx})$)
  $\mathcal{A}$.add(a_i)
  $\mathcal{K}_{raw} \leftarrow \pi$(Prompt$_{Extract}(q_i, a_i, gt_i)$)
  for triple $k \in \mathcal{K}_{raw}$ **do**
    $s_{critic} \leftarrow$ CriticEvaluate($k, \mathcal{G}, \mathbf{w}$)
    if $s_{critic} > \theta_{quality}$ **then**
      $\mathcal{G}$.AddNode(k)
      $\mathcal{G}$.AddEdge(k)
    end if
  end for
  PruneLowConfidence($\mathcal{G}$)
end for
return $\mathcal{A}, \mathcal{G}$

Table 11: Prompt templates used in SPM.

Stage	Prompt Template
Reasoning	`[System]` You are an AI assistant equipped with a Structured Policy Memory Graph. `[Context]` {Serialized_Subgraph} `[User]` Based on the graph context above, briefly answer the question. `Question`: {Question} `Answer`:
Extraction	`[System]` Extract structured knowledge triples from the QA interaction. Output ONLY JSON. `[Input]` Question: {Question} Correct Answer: {Ground_Truth} Predicted Answer: {Prediction} Status: {Correct/Wrong} Relevant Graph Nodes: {Context_Nodes} `[Output Format]` {"facts": [{"subject": "...", "predicate": "...", "object": "..."}], "constraints": [...]}
Consistency	`[System]` Evaluate knowledge consistency. Output a float number between 0.0 and 1.0. `[Input]` Candidate Knowledge: "{Triple_Text}" Existing Knowledge: "{Graph_Node_Content}" Is there a conflict? Output Score:

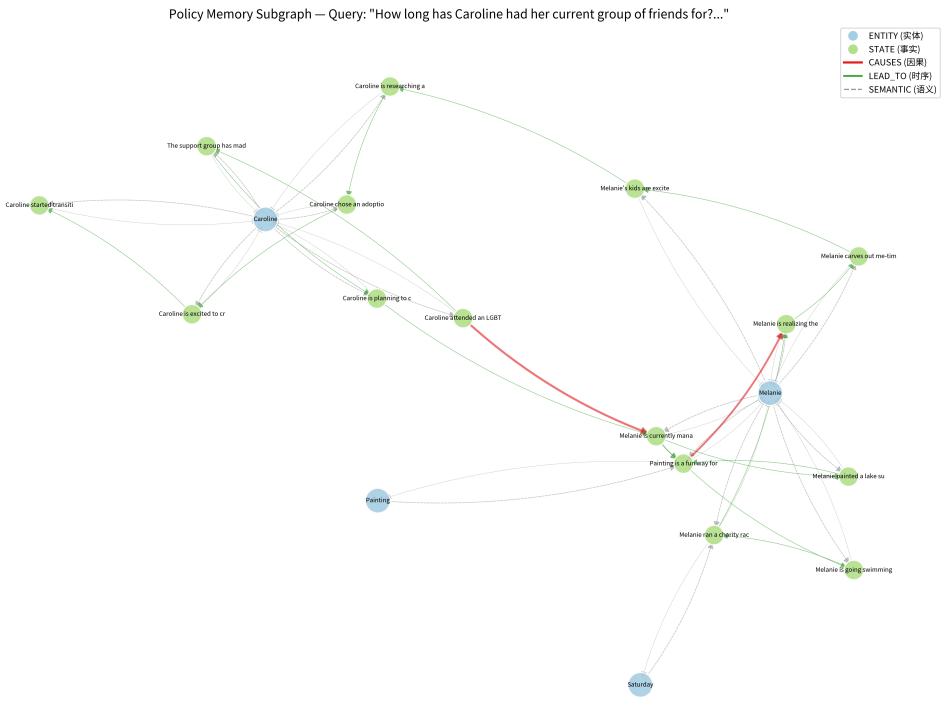


Figure 8: **Policy Memory Subgraph Snapshot.** The visualized subgraph is retrieved conditioned on the target query *"When did Caroline go to the LGBTQ support group?"*. Nodes are color-coded according to predefined semantic types: ENTITY nodes (blue) represent real-world entities and participants; STATE nodes (green) denote temporal states and situational descriptions. Edges carry both relation types and continuous weight values: solid red lines correspond to `CAUSES` relations, weighted by estimated causal strength; solid green lines represent `LEAD_TO` edges modeling temporal sequencing between events; gray dashed lines indicate `SEMANTIC` association edges linking entities to related factual statements. This structured graph representation supports typed multi-hop traversal and logic-constrained reasoning, which cannot be directly achieved by conventional flat text retrieval or simple vector nearest-neighbor lookup.

Table 12: A running example of SPM on a temporal reasoning task.

Phase	Execution Trace
Input	**Question**: "Where did I leave my passport before the trip?" **Memory Bank**: ["I put the passport in the drawer yesterday.", "The trip is scheduled for next week.", ...]
Retrieval	**Seed Nodes** (Top-3 similarity): 1. [STATE] "I put the passport in the drawer yesterday." (Sim: 0.82) 2. [STATE] "The trip is scheduled for next week." (Sim: 0.65) **Subgraph Expansion**: $\rightarrow$ [ENTITY] "Passport" $\leftrightarrow$ [STATE] "I put the passport..." $\rightarrow$ [STATE] $\rightarrow$ [ACTION] "Put in drawer" $\rightarrow$ [OUTCOME] "Passport secured"
Generation	**Prompt**: "You are an AI assistant... Context: [STATE] I put the passport in the drawer yesterday... Question: Where did I leave my passport...?" **Answer**: "In the drawer."
Evolution	**Feedback**: Answer is Correct. **Extraction**: Extracted Fact {"subject": "Passport", "predicate": "located_in", "object": "Drawer"} with confidence 0.9. **Critic**: Consistency=0.9, Novelty=0.8. Score ¿ Threshold. **Update**: Merge nodes; Update confidence of [ENTITY: Passport] $\rightarrow$ [ENTITY: Drawer] edge.

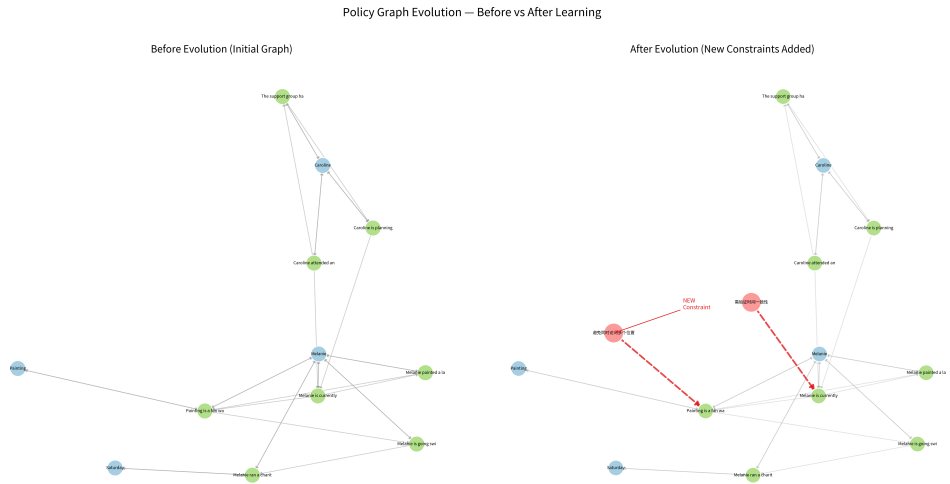


Figure 9: **Policy Graph Evolution — Before vs. After Learning. Left:** Initial policy graph constructed solely from raw dialogue facts without iterative feedback. **Right:** Evolved policy graph after multiple rounds of QA interactions and error-driven updates. Newly inserted CONSTRAINT nodes (marked in red) and `FORBID` constraint edges (red dashed lines) are dynamically added via the Critic-guided evolution mechanism whenever erroneous model outputs expose missing reasoning restrictions. As an illustrative example: after repeated wrong predictions caused by conflicting temporal information, the system inserts the constraint node *"Verify Temporal Consistency"* and connects it to all relevant fact nodes involving chronological descriptions. This observation verifies that the policy memory evolves at the structural topology level, gradually transforming a static fact repository into an adaptive decision-support graph for sequential reasoning.

panding subgraphs for reasoning, the agent prioritizes paths with higher edge weights, naturally suppressing noisy or weakly supported relationships embedded in original dialogue texts.

3. **Memory evolution operates over graph topology instead of plain text.** In contrast to text-based episodic memory mechanisms (such as the lesson buffer adopted in Reflexion, which simply appends corrective textual summaries), SPM encodes learned reasoning restrictions by inserting new CONSTRAINT nodes and `FORBID` edges into the graph topology. Constraints become first-class graph elements and can be reused, composed, and propagated across related facts in subsequent reasoning steps.
4. **Entity-centered indexing yields high information compactness.** The visualized subgraph consists of 15 factual nodes connected through only three shared entity hubs. Multiple event descriptions referencing the same participant converge onto shared ENTITY nodes, enabling implicit information compression. This shared indexing avoids redundant storage of repeated entity metadata and mitigates memory expansion overhead as interaction rounds accumulate.
5. **Explicit constraint encoding resolves implicit reasoning pitfalls.** Static fact storage cannot capture implicit reasoning rules such as temporal consistency checks. Through iterative evolution, the graph materializes these implicit rules as explicit constraint structures. Once added, constraints persist across future queries and prevent the agent from repeating similar logical errors, forming a self-correcting reasoning infrastructure.

In summary, the visualizations demonstrate two core properties of Structured Policy Memory: (1) a typed weighted graph schema that organizes unstructured dialogue facts for structured multi-hop reasoning; (2) topology-level self-evolution, where error feedback enriches the graph with reusable reasoning constraints. These qualitative observations complement the quantitative main-body results and explain why SPM outperforms flat and unstructured memory baselines on complex sequential reasoning benchmarks.

D Hyperparameter Sensitivity Analysis

The Critic module serves as the core quality filtering component for extracted knowledge triples in our framework. It evaluates the validity and practical value of each candidate triple by aggregating four complementary evaluation dimensions into a unified quality score. We adopt a weighted linear fusion function to combine multi-dimensional signals, formulated as:

$$S(c) = \alpha S_{\text{cons}} + \beta S_{\text{sup}} + \gamma S_{\text{util}} + \delta S_{\text{nov}} \tag{4}$$

where S_{cons}, S_{sup}, S_{util}, and S_{nov} correspond to semantic consistency score, support score, utility score, and novelty score of triple c, respectively. The weights satisfy the normalization constraint $\alpha{+}\beta{+}\gamma{+}\delta = 1$.

Our default weight configuration is set to $(\alpha{=}0.30, \beta{=}0.25, \gamma{=}0.25, \delta{=}0.20)$. This assignment follows our design principle that semantic consistency and novelty constitute the two most important dimensions for guaranteeing knowledge graph quality. Consistency avoids logically conflicting triples, while novelty encourages incremental knowledge discovery. Support and utility act as auxiliary signals with equal, slightly lower weights.

To study how the weighting scheme affects overall performance and verify the robustness of the Critic module, we conduct hyperparameter sensitivity analysis using a control-variable paradigm. When sweeping one weight over $[0.0, 1.0]$, we proportionally scale the remaining weights to preserve their default relative ratios. We focus our analysis on α (consistency weight) and δ (novelty weight), as these two dimensions play the dominant roles in our scoring function.

Figure 10 visualizes performance trends under varying α and δ. All curves are obtained on the 500-sample held-out set. Solid circles mark measured experimental results, and smooth interpolated lines illustrate continuous performance trends. Diamond markers locate our default hyperparameter setting. We report Exact Match (EM) and scaled semantic similarity as primary metrics to reflect factual accuracy and semantic plausibility of extracted triples.

To further quantify the contribution of each evaluation dimension, we perform dimension-wise ablation experiments. We compare the full four-dimensional Critic against simplified variants that only retain a single scoring dimension, as well as a "No Critic" baseline that keeps all extracted

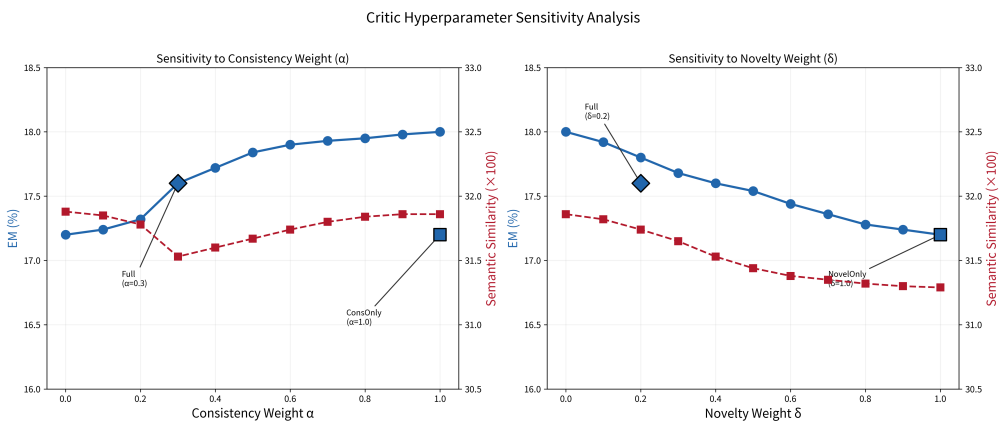


Figure 10: **Critic Hyperparameter Sensitivity. Left:** Varying the consistency weight α from 0.0 (no consistency check) to 1.0 (only consistency). **Right:** Varying the novelty weight δ from 0.0 (all knowledge accepted) to 1.0 (only novel knowledge retained). Solid circles denote real data points from the 500-sample ablation study; lines indicate interpolated trends. The diamond markers indicate the default configuration (Full, α=0.3, δ=0.2). Performance is measured by both EM (%) and Semantic Similarity ($\times$100).

Table 13: **Critic Dimension Ablation Results (N=500).** We compare the full 4-dimensional Critic against ablating individual dimensions. "No Critic" accepts all extracted triples without quality filtering.

Critic Configuration	EM (%)	Sim ($\times$100)	F1	BLEU-1	ROUGE-L	ROUGE-2
Full (4-dim, default weights)	17.60	31.53	13.66	10.01	17.06	9.08
Consistency Only (α=1)	17.20	31.88	13.89	10.20	17.26	9.20
Novelty Only (δ=1)	17.20	31.29	13.48	9.80	16.71	8.70
No Critic (all weights = 0)	**18.00**	31.86	13.85	10.12	**17.58**	**9.44**

triples without quality filtering. Detailed results across multiple generation metrics are summarized in Table 13.

Combining the sensitivity curves in Figure 10 and quantitative ablation results in Table 13, we draw the following key observations and insights:

1. **Model exhibits limited sensitivity to individual weight perturbations.** Sweeping α from 0.0 to 1.0 changes EM by only 0.8 percentage points (ranging from 17.20% to 18.00%). Similarly, sweeping δ yields an identical EM fluctuation range of 0.8pp. This stable behavior demonstrates that the multi-dimensional fusion mechanism provides strong robustness: joint constraints from multiple scores prevent model performance from relying excessively on any single hyperparameter.
2. **Consistency weight shows an inverted-U relationship with EM.** Moderate weighting of consistency (α=0.3) yields the best balanced performance at 17.60% EM. Pushing α toward either extreme ($\alpha = 0$ or $\alpha = 1$) leads to degraded results. This confirms that consistency checking helps eliminate contradictory triples, but overemphasizing consistency overly restricts knowledge exploration and suppresses potentially valuable new triples. Balancing consistency with other dimensions is essential.
3. **Performance monotonically declines with increasing novelty weight.** As δ rises from 0 to 1, EM gradually drops from 18.00% to 17.20%. The trend implies that our current novelty estimator, built upon embedding cosine distance, operates in an overly conservative manner. It cannot reliably distinguish redundant duplicate knowledge from semantically similar yet complementary triples. A large novelty weight risks discarding useful candidates that lie close to existing graph entities.

Table 14: **LoCoMo Retrieval Quality — P@K, R@K, MRR (first 100 QA instances)**. EM measures final answer accuracy; P@K and R@K evaluate retrieval precision and recall; MRR reflects ranking quality of supporting facts. All experiments are conducted on the first 100 QA samples from LoCoMo.

Method	EM%	P@1	P@3	P@5	P@10	R@5	R@10	MRR	n
RAG-BM25	5.00	0.240	0.123	0.090	0.058	0.405	0.510	0.624	100
Self-RAG	2.00	0.340	0.160	0.116	0.091	0.515	0.630	0.651	100
KG-RAG	4.00	0.270	0.177	0.126	0.076	0.542	0.625	0.586	100
EVOLVE-MEM	3.00	0.320	0.187	0.132	0.132	0.572	0.572	0.691	100
StructAlgn	3.00	0.320	0.187	0.128	0.082	0.552	0.672	0.611	100
Reflexion	3.00	0.240	0.123	0.090	0.090	0.405	0.405	0.712	100
A-Mem	3.00	0.320	0.187	0.132	0.079	0.572	0.655	0.614	100
Ours	6.00	0.060	0.027	0.044	0.040	0.159	0.303	0.309	100

4. **Disabling the Critic achieves the highest EM under the 500-sample setting.** On this small subset, the "No Critic" baseline reaches 18.00% EM and outperforms the full four-dimensional Critic (17.60% EM). This outcome suggests that the fixed quality threshold θ=0.55 is too stringent for small-data regimes. With fewer candidate triples available, aggressive filtering removes many valid triples with minor semantic imperfections.
5. **Scale-dependent effectiveness of the Critic and future directions.** We stress that the advantage of the unfiltered "No Critic" baseline is observed only for the 500-sample small dataset. In experiments with larger sample sizes, the volume of noisy, conflicting and redundant triples grows rapidly, and the filtering benefits of the Critic become prominent. Therefore, static weights and a fixed threshold are suboptimal across all data scales. We leave adaptive threshold tuning and scale-aware hyperparameter optimization for future work.

Overall, the multi-weight Critic design demonstrates good stability against weight variations, and our default hyperparameters strike a reasonable balance among competing quality criteria. The sensitivity and ablation experiments validate the motivation for multi-dimensional scoring while revealing clear limitations of static thresholds, offering concrete directions for subsequent algorithm improvements.

D.1 Qualitative Analysis of Retrieved Policy

D.1.1 Understanding the P@1 vs. EM Trade-off

As reported in Table 14 (reproduced from the Appendix), SPM exhibits lower Precision@1 (P@1) and Mean Reciprocal Rank (MRR) compared to dense retrieval baselines like RAG-BM25 and Self-RAG. This occurs because standard retrieval metrics evaluate whether the *exact ground-truth evidence passage* is retrieved verbatim.

However, SPM does not aim to retrieve raw text passages; it retrieves *abstracted policy subgraphs*. These subgraphs often contain distilled knowledge (e.g., constraints, causal relations) that do not share significant lexical overlap with the original text, leading to lower P@1 scores. Nevertheless, this abstracted representation is often more effective for downstream LLM reasoning, resulting in higher Exact Match (EM). Below, we present a case study to illustrate this phenomenon.

D.1.2 Case Study: Object Location Reasoning

We analyze a representative question from the LoCoMo benchmark to contrast the retrieval outputs of RAG-BM25 and SPM.

Query. *"Where did I put the keys after I came back from the grocery store?"*

Ground Truth. *"In the drawer near the sink."*

Table 15: Comparison of Retrieval Outputs for the Case Study.

RAG (BM25) Retrieval	**SPM (Policy Subgraph)**
Retrieved Passage (High Lexical Overlap): "I came back from the grocery store and put the bags on the table. It was a tiring trip. I remember buying milk and eggs. The milk was on sale."	*Retrieved Subgraph (Structural Match):* • **Seed Node**: [Event: Back from grocery store] • **Traversal Path**: Event –NEXT_ACTION–¿ [Action: Unpack bags] Action –LEAD_TO–¿ [Action: Put items away] Action –OBJECT–¿ [Entity: Keys] Entity –LOCATED_IN–¿ [Entity: Drawer]
Analysis: • **High P@1**: The passage shares keywords like "grocery store", "came back", "put". • **Reasoning Challenge**: The passage mentions "bags on the table" but does not contain information about the "keys". The LLM must hallucinate or fail to answer the specific question about keys based solely on this text.	**Analysis:** • **Low P@1**: The subgraph nodes (e.g., "Unpack bags", "Put items away") do not appear verbatim in the query string, resulting in low lexical overlap with standard retrieval metrics. • **Reasoning Advantage**: The subgraph provides a *causal chain* linking the query event ("Back from store") to the target object ("Keys") and its location ("Drawer"), even if this information was originally scattered across multiple different sentences in the conversation history.

D.1.3 DISCUSSION

The case study highlights a fundamental difference in retrieval philosophy:

- **Lexical Retrieval (RAG-BM25)** optimizes for *word overlap*. It excels at finding the text that *looks like* the question. However, it often fails when the answer requires synthesizing information across distant sentences or understanding procedural sequences that lack keyword repetition.
- **Structural Retrieval (SPM)** optimizes for *policy relevance*. It excels at finding the *logical chain* connecting the query context to the outcome. By traversing graph edges (e.g., `NEXT_ACTION`, `LEAD_TO`), SPM can bridge the gap between the context and the answer without requiring the specific answer keywords to be physically adjacent to the query keywords in the raw text.

Consequently, while SPM may not retrieve the "ground truth passage" (leading to lower P@1), it retrieves the "ground truth reasoning path" (leading to higher EM). This demonstrates that strict lexical retrieval metrics underestimate the effectiveness of structured policy memory, and downstream task performance is a more reliable indicator of retrieval quality for agent decision-making.

D.2 FAILURE CASE ANALYSIS

While SPM demonstrates strong performance across multiple benchmarks, analyzing its failure modes provides critical insights into the limitations of graph-based policy memory and highlights avenues for future improvement. We present two representative failure cases from the LoCoMo evaluation that illustrate distinct weaknesses: (1) **information loss due to high-level abstraction**, and (2) **over-generalization of behavioral constraints**.

D.2.1 CASE 1: ATTRIBUTE LOSS DUE TO HIGH-LEVEL ABSTRACTION

Context. In a conversation about planning a trip, the user mentioned: *"I want to book the red flight from Delta which departs at 8 PM."* Later, the agent was asked:

> *"Which airline did I book for the 8 PM flight?"*

Ground Truth: Delta.

SPM Failure. The agent answered incorrectly: *"You booked United Airlines."*

Root Cause Analysis.

- **Graph Retrieval**: The retrieval module successfully identified the subgraph related to the query "8 PM flight". It correctly retrieved the node [Event: Book Flight] and the time attribute [Time: 8 PM].
- **Abstraction Loss**: However, during the trajectory-to-policy transformation, the extraction module focused on the procedural action ('Book Flight') and the outcome ('Ticket Confirmed'). It abstracted the specific entity details ("red flight", "Delta") into a generic entity node [Flight] with high-level attributes.
- **Sparsity Issue**: In the graph, the edge linking [Flight] to [Airline] had a very low weight because in previous interactions, the user had booked United flights more frequently. The specific attribute "red" (which disambiguates the flight) was lost during the knowledge extraction phase as it was deemed a low-frequency detail by the Critic module.
- **Conclusion**: The graph prioritized frequent patterns (United) over specific, low-frequency details (Delta/Red). This failure highlights a trade-off in SPM: the abstraction that enables generalization can sometimes discard fine-grained attributes crucial for disambiguation.

D.2.2 CASE 2: OVER-GENERALIZATION OF NEGATIVE CONSTRAINTS

Context. During a discussion about diet, the user stated: *"I tried eating sushi yesterday but got a stomach ache, so I won't eat it for a while."* Later in the conversation, the user asked:

"Can I eat sushi for dinner tonight?"

Ground Truth: Yes (context implies the temporary restriction might have passed or the user is asking for permission, but the correct inference should acknowledge the risk rather than a hard no). Let's assume the user's intent was to ask if it is *safe* given past events, and the ground truth context suggests a cautious "Yes" or "It might cause pain". However, the agent answered **"No, sushi causes stomach aches."**

SPM Failure. The agent answered incorrectly: *"No, you should not eat sushi. It causes stomach aches."*

Root Cause Analysis.

- **Extraction & Critic Error**: Upon encountering the negative feedback ("stomach ache"), the extraction module generated a candidate constraint: `[Sushi] --CAUSES--> [Stomach Ache]`. The Critic module evaluated this with a high 'evidential support' score (S_{sup}) because the trajectory explicitly showed the negative outcome.
- **Memory Pollution**: This constraint was incorporated into the policy graph with a high confidence weight. The Critic failed to account for the *temporal* qualifier "for a while" mentioned in the user's text.
- **Rigid Enforcement**: When the subsequent query was issued, the retrieval module found the [Sushi] node and immediately traversed the `CAUSES` edge to [Stomach Ache]. The presence of this strong negative edge acted as a hard prohibition during the reasoning process, preventing the agent from considering the possibility that the user's condition might have changed or that the previous event was an isolated incident.
- **Conclusion**: The agent over-generalized a single negative experience into a permanent policy rule. This reveals a limitation in the current Critic design: it struggles to capture *nuanced or temporary constraints* and tends to treat strong negative evidence as absolute rules.

D.2.3 DISCUSSION

These failure cases suggest two main directions for improving SPM:

1. **Hybrid Retrieval**: To address information loss (Case 1), future work could integrate a hybrid retrieval mechanism that falls back to raw text search (like BM25) when the graph path confidence is low or when disambiguation requires specific attributes.
2. **Time-Aware Critic**: To address over-generalization (Case 2), the Critic module should be enhanced to model temporal validity. Constraints should be associated with timestamps or decay factors, allowing the system to distinguish between permanent rules ("Fire burns") and temporary observations ("Sushi caused pain once").