LEARNING WHAT MATTERS: COUNTERFACTUAL POLICY LEARNING FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents increasingly improve through accumulated interaction experience, yet it remains unclear which decisions within a trajectory are worth learning and reusing. Successful trajectories may contain many incidental actions, so trajectory-level memorization cannot reliably distinguish essential decisions from irrelevant details. We propose Counterfactual Policy Learning (CPL), a decision-level experience learning framework that separates candidate identification from policy validation. CPL contrasts successful and failed trajectories to identify candidate decision differences, treating this observational signal only as relevance rather than causal evidence. It then abstracts candidates into intervention-testable policies and performs same-task interventions that replace policy-consistent decisions with alternative actions. Only policies with positive intervention effects and sufficient admission confidence are admitted into policy memory; unseen-task performance is used solely for downstream evaluation. On ALFWorld, policy-guided interventions recover 10.8% of failed episodes versus 0% for random replacements; on LoCoMo, CPL achieves 32.65% overall Semantic Similarity. Ablations show that type-matched retrieval acts as a firewall against negative transfer, reducing ReusePerf from 53.86% to 43.81% when replaced by random retrieval. CPL thus reframes experience learning from "what happened?" to "which decision is worth learning from?"Our code is available at https://github.com/linda-2-hh/Counterfactual-Policy-Learning.

1 INTRODUCTION

Large language model (LLM) agents have recently demonstrated strong capabilities in reasoning, planning, and interacting with complex environments. However, their performance often remains myopic: an agent may repeatedly make similar decisions across tasks despite having encountered related successes or failures. This has motivated a growing body of work on agent memory and experience learning, where previous interactions are stored, reflected upon, or transformed into reusable guidance Shinn et al. (2023); Zhao et al. (2024). Recent systems further improve memory organization and management through dynamic memory structures Xu et al. (2025); Chhikara et al. (2025); Yu et al. (2026). These studies demonstrate that experience can serve as an important source of continual improvement.

Yet, an important question remains largely unresolved: *which parts of an experience are actually worth learning from?* A successful trajectory may contain many incidental actions, while a failed trajectory may differ from a successful one at only a few consequential decisions. Simply storing trajectories, extracting reflections, or assigning step-level rewards does not necessarily determine whether a particular decision is responsible for the observed outcome. Recent approaches have begun to exploit failure trajectories and step-level signals. ETO Song et al. (2024) constructs contrastive pairs from successful and failed trajectories for policy optimization, while STeCa Wang et al. (2025) identifies suboptimal actions through step-level reward comparison and calibrates trajectories. These methods establish the value of negative experience and fine-grained decision signals, but primarily use them to optimize agent behavior rather than to determine which decisions should become persistent, reusable policies.

This distinction is important for continual agents. An experience can be *correlated* with success without being sufficiently reliable to guide future behavior. We therefore propose **Counterfactual**

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.

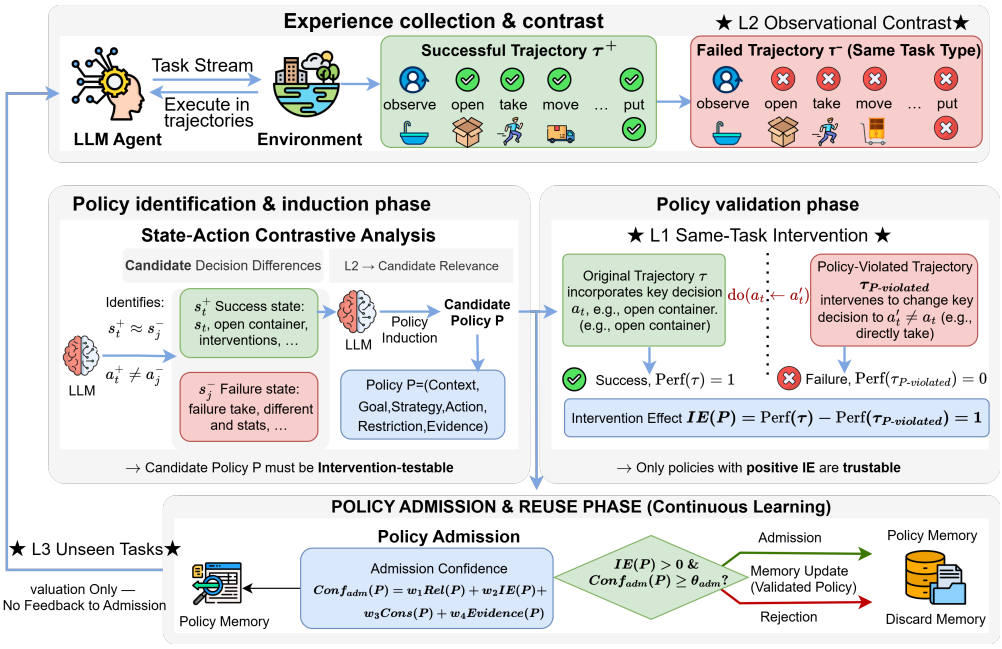

Figure 1: Overview of Counterfactual Policy Learning (CPL). CPL contrasts successful and failed trajectories to identify candidate decision patterns and induces intervention-testable policies. Candidate policies are subsequently validated through same-task counterfactual intervention, where the key decision is replaced by a policy-violating alternative and the intervention effect $\mathrm{IE}(P)$ is measured. Only intervention-supported policies with sufficient admission confidence are retained in policy memory, while reuse on unseen tasks is evaluated independently.

Policy Learning (CPL), a decision-level experience learning framework that separates *candidate identification* from *policy validation*. CPL first contrasts successful and failed experiences of similar task types to identify candidate decision differences. These observations are explicitly treated as relevance signals rather than causal evidence. For each candidate policy, CPL then performs a targeted intervention on the same task by replacing the relevant decision with a policy-violating alternative and measures the resulting performance change. Only policies supported by positive intervention effects are admitted into persistent policy memory and subsequently evaluated on unseen tasks.

The key principle is therefore:

> *Use observational contrast to identify what may matter, and intervention to verify what is worth remembering.*

This design also distinguishes CPL from recent counterfactual and feedback-based approaches. Counterfactual reasoning has been explored for analyzing language-model behavior Liu et al. (2025); Bynum & Cho (2025); Pona et al. (2025), while feedback-based agent learning methods such as LEAP Choudhury & Sodhi (2025) use external feedback to improve model policies. CPL instead uses intervention as an *admission criterion* for an external policy memory, without requiring parameter updates or claiming causal discovery.

We evaluate CPL across interactive and reasoning-oriented settings, including ALFWorld, StrategyQA, and LoCoMo. Our experiments examine not only end-task performance but also intervention effects, policy survival, and cross-task reuse. We summarize our contributions as follows:

- We introduce **CPL**, a decision-level experience learning framework that separates observational candidate identification from intervention-based policy validation.
- We propose an **intervention-based policy admission mechanism**, retaining a policy only when violating it produces measurable performance degradation on held-out experiences.

- We provide a **three-level evaluation protocol** distinguishing observational relevance, intervention evidence, and unseen-task reuse, enabling more precise analysis of what makes an experience worth learning from.
- Experiments across three benchmarks demonstrate that intervention-validated policies provide more reliable and reusable experience than conventional trajectory-level memory and reflection approaches.

2 Related Work

2.1 Agent Memory and Experience Learning

Memory-augmented LLM agents increasingly learn from prior interactions rather than treating each task independently. Reflexion Shinn et al. (2023) stores verbal feedback from previous trials, while ExpeL Zhao et al. (2024) extracts reusable experiential knowledge from past interactions. MetaReflection Gupta et al. (2024) further learns instructions from historical reflections through an experiential semantic memory, and Experiential Co-Learning Qian et al. (2024) extracts shortcut-oriented experiences from historical trajectories for solving unseen tasks. These approaches demonstrate that past experience can provide reusable guidance beyond the current trajectory.

Recent work has also focused on evolving agent policies and memory management. AgentPro Zhang et al. (2024) performs policy-level reflection and optimization over interactive experiences, progressively improving behavioral policies. A-MEM Xu et al. (2025) introduces an adaptive memory organization mechanism, while Mem0 Chhikara et al. (2025) provides a production-oriented memory layer for persistent agent memory. More recently, AgeMem Yu et al. (2026) integrates long- and short-term memory operations into the agent policy and learns when to store, retrieve, update, summarize, or discard memories. Despite these advances, most approaches focus on *how to store, organize, or optimize experience*, leaving open the question of *which decisions within an experience are sufficiently reliable to retain*.

2.2 Learning from Failures and Decision-Level Experience

A complementary line of work exploits unsuccessful trajectories or fine-grained decision signals to improve agent learning. ETO Song et al. (2024) constructs contrastive preference pairs from successful and failed trajectories, demonstrating the value of negative experience for policy optimization. STeCa Wang et al. (2025) further moves to the step level by comparing rewards across trajectory steps to identify suboptimal actions and calibrate trajectories for agent learning. These methods are closely related to CPL in exploiting outcome contrasts and decision-level signals, but their primary goal is to *optimize or calibrate agent behavior*. In contrast, CPL uses contrastive experience only to identify *candidate* decisions and does not treat observational differences as causal evidence. A candidate policy is retained only after its relevant decision is subjected to a same-task intervention that tests whether violating the policy changes task performance.

2.3 Counterfactual and Intervention-Based Reasoning

Counterfactual reasoning has also been explored as a means of understanding and improving language-model behavior. CREST Liu et al. (2025) employs counterfactual reasoning to mitigate shortcut learning, while recent work on causal language models investigates the generation of observational, interventional, and counterfactual samples Bynum & Cho (2025). Abstract Counterfactuals Pona et al. (2025) further studies higher-level interventions over agent actions rather than relying solely on token-level perturbations. CPL is complementary to these efforts but adopts a narrower and more operational role for intervention: it does not attempt to discover a causal structure or identify a complete causal model. Instead, intervention serves as an empirical *policy validation and admission criterion*. Observational contrast identifies potentially relevant decisions, whereas same-task intervention provides the evidence required before a policy is retained.

Overall, existing work has progressed from storing trajectories and reflections to extracting reusable experiences, learning from failures, and dynamically managing memory. CPL addresses a different bottleneck: *determining which decision-level experience is worth remembering*. It therefore sep-

arates experience learning into two stages—contrastive candidate identification and intervention-based policy validation—and evaluates the resulting policies through subsequent cross-task reuse.

3 COUNTERFACTUAL POLICY LEARNING

3.1 PROBLEM FORMULATION AND FRAMEWORK OVERVIEW

We consider an LLM agent that sequentially solves a stream of tasks $\mathcal{T} = \{T_1, T_2, \ldots\}$. For a task T, the agent interacts with the environment through a trajectory $\tau = \{(s_t, a_t, r_t)\}_{t=1}^{H}$, where s_t denotes the observed state, a_t the selected action, and r_t the resulting feedback. The trajectory is evaluated by a binary outcome $y(\tau) \in \{0, 1\}$ indicating task success or failure. The agent is conditioned on a policy memory $\mathcal{M}$ that stores reusable decision policies.

The central challenge is to determine *which decisions are worth learning from* and, more importantly, *how to verify that a candidate decision actually matters*. CPL addresses this problem by separating decision identification from intervention-based validation. Given successful and failed experiences of the same task type, CPL first identifies candidate decision differences using observational contrast. These candidates are then abstracted into intervention-testable policies and validated through same-task interventions. Only policies supported by positive intervention effects and sufficient admission confidence are added to $\mathcal{M}$. Policies stored in memory are subsequently retrieved for unseen tasks, where their transfer utility is evaluated without feeding the evaluation outcome back into the admission process.

The resulting learning process follows four stages: *contrastive experience construction*, *candidate decision identification and policy induction*, *same-task intervention validation*, and *policy admission and reuse*. This design explicitly separates three levels of evidence: L2 observational contrasts for candidate relevance, L1 same-task interventions for intervention evidence, and L3 unseen-task evaluation for transfer performance. In particular, CPL does not treat observational relevance as causal evidence.

3.2 CONTRASTIVE DECISION IDENTIFICATION

CPL exploits two complementary types of experience. First, successful and failed trajectories of the same task type are paired to form L2 *observational contrasts*. The trajectories are aligned according to their semantic states, and decision points with similar states but different actions are extracted as candidate decision differences:

$$\mathcal{D}_{\text{cand}} = \left\{ (d, \text{Rel}(d)) \,\middle|\, s_t^+ \approx s_j^-,\ a_t^+ \neq a_j^-,\ \text{Rel}(d) \geq \theta_{\text{rel}} \right\}.$$

Here, $\text{Rel}(d)$ measures the relevance of a decision difference for explaining the observed outcome contrast. It is obtained from state-action alignment, cross-experience frequency, and LLM-based structured judgment.

Importantly, $\text{Rel}(d)$ is deliberately treated as a *candidate relevance score*, rather than a causal effect. A decision may appear frequently in successful trajectories and differ from its counterpart in failed trajectories while still being confounded with other differences in the trajectories. Therefore, L2 evidence only determines which decisions are sufficiently informative to proceed to policy induction; it does not justify a causal claim.

This distinction is central to CPL. Instead of directly converting successful experience into memory, CPL uses failure experience as a contrastive signal to identify where the agent's decisions diverged. The result is a substantially smaller set of potentially meaningful decisions, which reduces the cost of subsequent intervention validation.

3.3 POLICY INDUCTION AND SAME-TASK INTERVENTION VALIDATION

The candidate decision differences are converted into reusable, intervention-testable policies by an LLM. Each policy is represented as

$$P = (C, G, S, A, R, E),$$

where C, G, S, A, R, and E denote the decision context, goal, strategy, concrete action, restriction, and supporting evidence, respectively. The key requirement is that the action and restriction fields must specify an executable alternative, so that the policy can later be tested by changing the corresponding decision in an otherwise comparable trajectory. The evidence field records the originating contrastive experiences, making the policy traceable to its empirical support.

The induced policy is then evaluated using L1 *same-task intervention*. For a validation trajectory τ that contains the decision specified by P, we construct a policy-violated trajectory $\tau_{P\text{-violated}}$ by intervening on the corresponding action while keeping the task and preceding trajectory fixed. When the environment supports replay, the modified action is executed in the environment rather than merely simulated by the LLM. The intervention effect of the policy is therefore measured as

$$\mathrm{IE}(P) = \frac{1}{|\mathcal{V}_P|} \sum_{\tau \in \mathcal{V}_P} \left[\mathrm{Perf}(\tau) - \mathrm{Perf}(\tau_{P\text{-violated}}) \right],$$

where $\mathcal{V}_P$ is a held-out validation set that is not used for policy induction.

A positive $\mathrm{IE}(P)$ provides intervention-based evidence that violating the policy changes the task outcome under the tested conditions. This is the only evidence in CPL that supports an intervention-based causal interpretation. In environments without executable intervention, such as reasoning-only benchmarks, the same procedure is interpreted strictly as a *reasoning-level intervention effect*, rather than an environmental causal effect.

This validation stage is therefore fundamentally different from simply asking an LLM whether a policy is useful. The LLM proposes the policy and constructs the intervention, but the evidence for its usefulness comes from the resulting intervention outcome. Random decision interventions are used as an additional control to examine whether the observed effect is concentrated on the candidate decision rather than arising from arbitrary trajectory perturbations.

3.4 Policy Admission and Continual Reuse

Intervention evidence alone is not sufficient to determine whether a policy should be stored. A policy may exhibit a positive effect in a small number of cases while having weak observational support or poor consistency across experiences. CPL therefore combines intervention evidence with the non-causal evidence available before admission. Specifically, we compute an admission confidence as

$$\mathrm{Conf}_{\mathrm{adm}}(P) = w_1 \overline{\mathrm{Rel}}(P) + w_2 \widehat{\mathrm{IE}}(P) + w_3 \mathrm{Cons}(P) + w_4 \mathrm{Evidence}(P),$$

where $\overline{\mathrm{Rel}}(P)$ denotes the average candidate relevance, $\widehat{\mathrm{IE}}(P)$ the normalized intervention effect, $\mathrm{Cons}(P)$ the consistency of the decision direction across supporting experiences, and $\mathrm{Evidence}(P)$ the amount of supporting evidence.

A candidate policy is admitted into the policy memory only when both conditions are satisfied: $\mathrm{IE}(P) > 0$ and $\mathrm{Conf}_{\mathrm{adm}}(P) \geq \theta_{\mathrm{adm}}$. Otherwise, the policy is discarded. This two-stage admission mechanism makes the intervention result an explicit evidence gate rather than merely another feature used to rank policies.

After admission, the validated policy becomes available to the agent through policy retrieval when solving subsequent tasks. Its transfer utility is measured on temporally later, unseen tasks. We define the reuse performance as the fraction of retrieved policies that lead to improved task outcomes relative to the corresponding agent without the policy:

$$\mathrm{ReusePerf}(P) = \frac{\#\{T_{\mathrm{new}} : y(\pi(\cdot \mid \mathcal{M} \cup P)) > y(\pi(\cdot \mid \mathcal{M}))\}}{\#\{T_{\mathrm{new}} : P \text{ is retrieved}\}}.$$

Crucially, $\mathrm{ReusePerf}$ is an evaluation metric rather than an admission signal. Outcomes on unseen tasks are therefore never fed back into the admission decision, preventing temporal leakage and preserving a clean separation between policy validation and policy transfer.

Overall, CPL forms a closed learning cycle of *observational contrast* $\rightarrow$ *candidate policy* $\rightarrow$ *same-task intervention* $\rightarrow$ *evidence-based admission* $\rightarrow$ *policy reuse*. The key distinction is that experience is not accepted merely because it is successful or judged useful by the LLM; it must first survive an intervention-based validation step.

Table 1: Main results on LoCoMo. Semantic Similarity (%) across four question categories and overall. Best performance is bold. Superscripts denote difference from column mean: red ↑ above mean, blue ↓ below mean.

Method	Factoid	Temporal	Multi-hop	Event	Overall
LLM-Only Bi et al. (2024)	18.28↓9.93	15.85↑+1.95	18.87↓7.25	16.27↓12.21	16.71↓8.52
CoT Wei et al. (2022)	24.48↓3.73	14.28↑+0.38	30.52↑+4.40	24.14↓4.34	22.55↓2.68
RAG-BM25 Robertson & Zaragoza (2009)	33.29↑+5.08	12.62↓1.28	26.05↓0.07	39.30↑+10.82	31.81↑+6.58
Self-RAG Asai et al. (2024)	31.71↑+3.50	8.65↓5.25	32.40↑+6.28	34.93↑+6.45	28.71↑+3.48
EVOLVE-MEM Shah et al. (2025)	17.05↓11.16	9.92↓3.98	19.40↓6.72	18.63↓9.85	16.57↓8.66
Reflexion Shinn et al. (2023)	34.41↑+6.20	17.90↑+4.00	32.59↑+6.47	31.52↑+3.04	29.28↑+4.05
Structurally Aligned Shen et al. (2026)	33.20↑+4.99	11.93↓1.97	**34.06**↑+7.94	33.88↑+5.40	29.20↑+3.97
AMA-Agent Zhao et al. (2026)	22.96↓5.25	**20.16**↑+6.26	15.10↓11.02	19.03↓9.45	19.55↓5.68
CPL (Ours)	**38.47**↑+10.26	13.79↓0.11	26.11↓0.01	38.63↑+10.15	**32.65**↑+7.42

4 EXPERIMENTS

We evaluate CPL along three dimensions: *end-task performance*, *intervention-based validation*, and *evidence-based admission*. The experiments follow the same principle: candidate decisions are identified from experience, validated by intervention when possible, and retained only if they pass the admission gate. End-task results measure practical utility, while intervention and ablation studies test whether the proposed admission mechanism prevents unsupported experience from propagating.

4.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate CPL on three complementary benchmarks. **LoCoMo** (Maharana et al., 2024) evaluates long-term memory over multi-session conversations, including factoid, temporal, multi-hop, and event reasoning. **StrategyQA** (Geva et al., 2021) evaluates reasoning-level policy reuse at the answer level. **ALFWorld** (Shridhar et al., 2020) provides an interactive embodied environment in which alternative actions can be executed and their outcomes observed.

Baselines. We compare against non-memory, retrieval-based, reflection-based, and agentic memory methods: LLM-Only (Bi et al., 2024), CoT (Wei et al., 2022), RAG-BM25 (Robertson & Zaragoza, 2009), Self-RAG (Asai et al., 2024), Reflexion (Shinn et al., 2023), Mem0 (Chhikara et al., 2025), EVOLVE-MEM (Shah et al., 2025), A-Mem (Xu et al., 2026), and Structurally Aligned (Shen et al., 2026).

Metrics. We report Semantic Similarity (Sim) on LoCoMo and Exact Match (EM) on StrategyQA. For intervention experiments, we report intervention effect (IE), outcome-flip rate, and recovery rate. For policy admission, we report the number of admitted policies, coverage, and ReusePerf. Performance on future tasks is used only for evaluation and never for policy admission.

4.2 END-TASK PERFORMANCE

Table 2: ALFWorld end-task performance with DeepSeek-7B under the same 26-task evaluation setting.

Method	Success (Rate %)	Method	Success (Rate %)	Method	Success (Rate %)
A-Mem	3/26 (11.5)	ExpeL	0/26 (0.0)	RAG-BM25	0/26 (0.0)
Agent-Pro	1/26 (3.8)	LLM-Only	0/26 (0.0)	Reflexion	0/26 (0.0)
CPL (Ours)	1/26 (3.8)	Mem0	0/26 (0.0)		

LoCoMo. Table 1 shows that CPL remains competitive on long-term memory, achieving 32.65(38.47a small number of consequential decisions can determine the answer. Temporal reasoning is weaker (13.79(26.11boundary condition of decision-level learning: CPL is most suitable when task success is relatively localized to reusable decisions, rather than diffusely distributed across a long reasoning chain.

Table 3: Intervention-based validation. (a) LoCoMo compares policy removal with random removal. (b) ALFWorld compares policy-guided fixes with random valid replacements in failed trajectories.

(a) LoCoMo

Intervention	N	IE	Flip (%)
Policy-removal	132	0.167	16.7
Random-removal	132	0.197	19.7
Difference	–	-0.030	-3.0

(b) ALFWorld

Intervention	Fixes	Wins	Win Rate (%)
Policy Fix	74	8	**10.8**
Random Fix	74	0	0.0
Difference	–	+8	**+10.8**

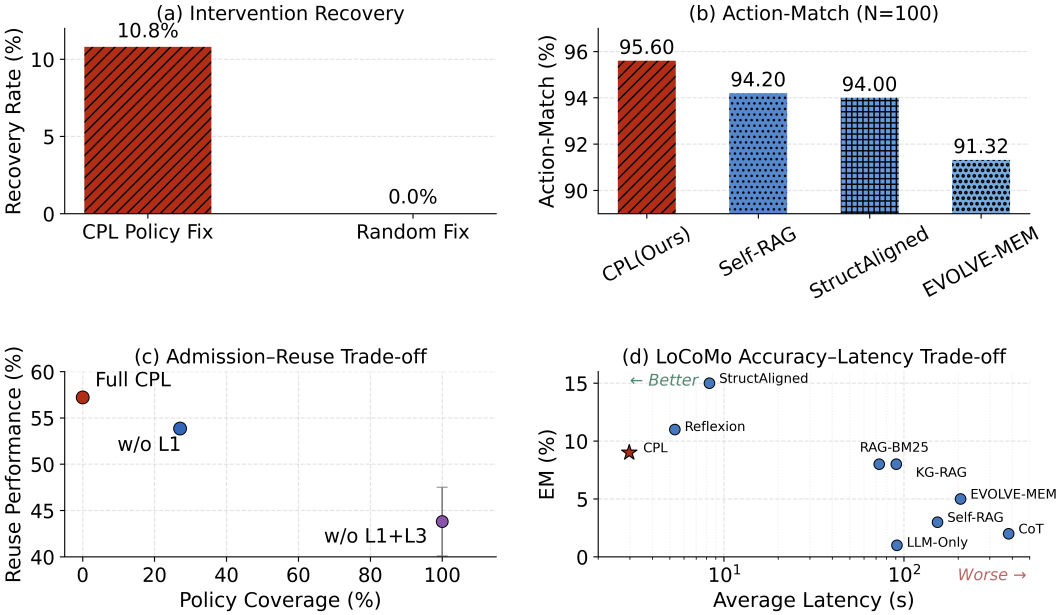

Figure 2: Experimental evidence for CPL. (a) Policy-guided interventions recover 8/74 failed ALFWorld episodes, versus 0/74 for random replacements. (b) CPL achieves 95.60(c) Removing L1 increases policy admission but reduces reuse performance; additionally removing type-matched L3 retrieval further reduces ReusePerf to 43.81(d) CPL achieves 9.0

ALFWorld. On the 26-task ALFWorld setting, CPL solves 1/26 tasks (3.8Agent-Pro but remaining below A-Mem. We do not interpret this small difference as evidence of an end-task advantage. With at most three successful tasks across methods, the benchmark resolution is approximately 3.8 percentage points per task, making end-task success a coarse measure of the memory mechanism. We therefore use ALFWorld primarily for the intervention analysis in Section 4.3, where the effect of individual policy decisions can be tested directly.

Efficiency. Figure 2(d) reports results on the first 100 LoCoMo QA pairs. CPL obtains 9.0latency, compared with 72.86s for RAG-BM25, 90.55s for KG-RAG, 153.71s for Self-RAG, and 206.37s for EVOLVE-MEM. The low latency follows from CPL's compact policy memory: only admitted policies are retrieved at inference time, avoiding repeated search over large trajectory or document stores.

4.3 Intervention-based Policy Validation

We directly intervene on policy-supported decisions and compare them with random valid interventions. The key test is whether policy-guided changes produce more useful outcome changes than arbitrary perturbations. This distinguishes actionable policy decisions from merely content-bearing decisions.

Table 4: Ablation of L2 candidate selection, L1 intervention validation, and L3 policy retrieval on 687 held-out instances. Admission is the number of admitted policies; Coverage is the fraction of instances retrieving a policy; Δ is the change in ReusePerf relative to full CPL. The w/o L3 row is a structural null because full CPL admits no policy. The w/o L1+L3 row fixes the six-policy memory produced by w/o L1 and replaces type-matched retrieval with random retrieval.

Variant	Admission	Coverage (%)	ReusePerf (%)	Δ
CPL (Full)	0/17	0.00	57.21	0.00
w/o L1	6/17	27.07	53.86	-3.35
w/o L2	0/17	0.00	57.21	0.00
w/o L3 (on full admission)	0/17	0.00	57.21	0.00
w/o L1+L3 (random retrieval)	6/17	100.00	43.81	-13.40

LoCoMo. The intervention result is negative. Policy removal produces IE=0.167 and a 16.7(IE=0.197, 19.7heavy setting, many content-bearing decisions are individually influential, so observational contrast does not reliably localize a uniquely consequential decision. We therefore treat LoCoMo as a reasoning-level intervention stress test rather than causal evidence. This result also motivates the interactive ALFWorld evaluation.

ALFWorld. In contrast, real-environment intervention provides direct behavioral evidence. As shown in Table 3(b) and Figure 2(a), 8/74 policy-guided fixes recover failed episodes (10.8succeed. Because both interventions are executable in the environment and the random arm controls for valid perturbations, the gap indicates that CPL-identified policy violations are more actionable than arbitrary valid replacements. Figure 2(b) further shows 95.60policy retrieval.

4.4 Ablation: Evidence-based Policy Admission

CPL separates contrastive candidate generation (L2), intervention validation (L1), and policy retrieval (L3). We ablate these components on 687 held-out instances to identify their distinct failure modes.

L1 prevents unsupported policies from entering memory. Removing L1 admits 6/17 policies and increases coverage to 27.07reduces ReusePerf from 57.21performance falls to 46.77the six admitted policies have negative intervention effects. Thus, L1 does more than filter weak candidates: it prevents observationally plausible but intervention-unsupported policies from being retained. Full CPL admits no policy under the conservative gate and consequently introduces no additional negative transfer. This conservative behavior is desirable in continual learning because unsupported experience is rejected rather than propagated.

L3 limits where harmful policies are applied. The full-admission w/o L3 row is a structural null: because full CPL admits zero policies, all 687 instances use the base agent and L3 is never active. The informative comparison is therefore the nested w/o L1+L3 condition. Holding the same six-policy memory fixed, type-matched retrieval obtains 53.86(301/687, 95make harmful policies correct; it limits their exposure by conditioning retrieval on task type. This ablation should be interpreted as a retrieval control given a non-empty memory, not as an independent removal from full CPL.

L2 reduces the candidate search space. Removing L2 and using random candidate generation causes all 17 induced candidates to fail validation. Random candidates have low relevance (≈ 0.05), indicating that L2 primarily improves candidate localization rather than providing evidence for admission. Its role is to concentrate the expensive L1 intervention test on decisions that differ between successful and failed experiences; L1 then determines whether those candidates are actually actionable.

L2 candidate quality. Table 5 confirms this division of labor. L2 increases mean relevance from 0.373 to 0.472 and the proportion of high-relevance candidates ($\mathrm{Rel} \geq 0.6$) from 8.4not translate into a higher positive IE rate: 1.01.6Among L2 candidates, 389/534 are intervention-testable. Counting

Table 5: Candidate quality under random and L2 contrastive selection. Rel. is the M2 relevance score; high relevance denotes $\mathrm{Rel} \geq 0.6$. Positive IE indicates a successful outcome change after intervention.

Candidate Source	Mean Rel. ↑	Rel.≥0.6 (%) ↑	Positive IE (%)	Valid Policy (%) ↑
Random Selection	0.373	8.4	1.6 (8/512)	1.6
L2 Contrastive Selection	**0.472**	**27.2**	1.0 (4/389)	1.0

Table 6: Sensitivity to the admission threshold θ_{adm} over 10 candidate policies. ReusePerf is measured at policy firings on the test split; Survival is the proportion of successfully completed test episodes.

θ_{adm}	Admission	Coverage (%)	ReusePerf (%)	Neg. Transfer (%)	Survival (%)
0.1	10/10	100.0	96.2	3.8	30.0
0.2	10/10	100.0	96.2	3.8	30.0
0.3	8/10	80.0	96.2	3.8	37.5
0.5	7/10	70.0	**100.0**	**0.0**	**41.2**
0.6	1/10	10.0	100.0	0.0	11.5
0.7	0/10	0.0	–	–	–
0.8	0/10	0.0	–	–	–
0.9	0/10	0.0	–	–	–

the 145 non-testable candidates conservatively as zero-IE cases gives 4/534=0.75($p = 0.26$). Hence, observational relevance and intervention effectiveness remain distinct properties: L2 improves the former, while L1 verifies the latter.

Admission threshold sensitivity. We study θ_{adm} on 10 candidate policies induced from 145 high-relevance candidates. The admission confidence is

$$\mathrm{Conf}_{\mathrm{adm}}(P) = 0.25\,\mathrm{Rel}(P) + 0.40\,\mathrm{IE}(P) + 0.20\,\mathrm{Cons}(P) + 0.15\,\mathrm{Evidence}(P),$$

where IE is obtained from real-environment fix-direction interventions.

Table 6 shows the coverage–reliability trade-off. At $\theta_{\mathrm{adm}} \leq 0.2$, all policies are admitted, but negative transfer remains. At 0.5, seven policies are retained, ReusePerf reaches 100.0At 0.6, only one policy remains without further ReusePerf improvement, whereas thresholds ≥ 0.7 reject all candidates. We therefore use 0.5 as the operating point. The threshold was fixed before this sensitivity analysis rather than tuned on the evaluation split.

Intervention signal. We use fix-direction IE because it provides a non-degenerate signal for candidate discrimination. At $\theta_{\mathrm{adm}} = 0.6$, the surviving policy is `P_USE_TOOL_look_at_obj_in_light`, with IE=0.20, the only candidate with positive fix-direction IE. Break-direction IE is near saturated for seven policies and therefore provides little discrimination. This result highlights a practical requirement of the admission gate: the intervention signal must contain sufficient variation to distinguish candidate policies.

5 Conclusion

We presented **Counterfactual Policy Learning (CPL)**, a framework for learning reusable decision policies from agent experience through contrastive decision identification and intervention-based validation. CPL separates candidate discovery from evidence-based policy admission by identifying decision-level candidates from success–failure contrasts and validating them through executable interventions when available. Experiments show that this separation improves the reliability of policy reuse, while task-type-conditioned retrieval helps mitigate negative transfer. Rather than pursuing causal discovery, CPL provides an operational approach for determining which observed decisions are sufficiently supported to become reusable policies.

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/Counterfactual-Policy-Learning` 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.

Lucius E. J. Bynum and Kyunghyun Cho. Language models as causal effect generators. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2025.

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.

Sanjiban Choudhury and Paloma Sodhi. Better than your teacher: LLM agents that learn from privileged AI feedback. In *The Thirteenth International Conference on Learning Representations*, 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.

Priyanshu Gupta, Shashank Kirtania, Ananya Singha, Sumit Gulwani, Arjun Radhakrishna, Gustavo Soares, and Sherry Shi. MetaReflection: Learning instructions for language agents using past reflections. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8369–8385, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.477. URL `https://aclanthology.org/2024.emnlp-main.477/`.

Zhonghua Liu, Wei Shao, Shaolong E., and Xia Cao. CREST: A causal framework for mitigating shortcut learning in language models through counterfactual reasoning. *Information Processing & Management*, 62(5):104418, 2025. doi: 10.1016/j.ipm.2025.104418.

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.

Edoardo Pona, Milad Kazemi, Yali Du, David Watson, and Nicola Paoletti. Abstract counterfactuals for language model agents. *arXiv preprint arXiv:2506.02946*, 2025.

Chen Qian, Yufan Dang, Jiahao Li, Wei Liu, Zihao Xie, YiFei Wang, Weize Chen, Cheng Yang, Xin Cong, Xiaoyin Che, Zhiyuan Liu, and Maosong Sun. Experiential co-learning of software-developing agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5628–5640, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.305. URL https://aclanthology.org/2024.acl-long.305/.

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

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. In *Advances in Neural Information Processing Systems*, 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.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7584–7600, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.409.

Hanlin Wang, Jian Wang, Chak Tou Leong, and Wenjie Li. STeCa: Step-level trajectory calibration for LLM agent learning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11597–11614, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.604. URL https://aclanthology.org/2025.findings-acl.604/.

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.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic Memory for LLM Agents. *arXiv preprint arXiv:2502.12110*, 2025.

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.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21457–21483, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.981. URL https://aclanthology.org/2026.acl-long.981/.

Wenqi Zhang, Ke Tang, Hai Wu, Mengna Wang, Yongliang Shen, Guiyang Hou, Zeqi Tan, Peng Li, Yueting Zhuang, and Weiming Lu. Agent-Pro: Learning to evolve via policy-level reflection and optimization. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings*

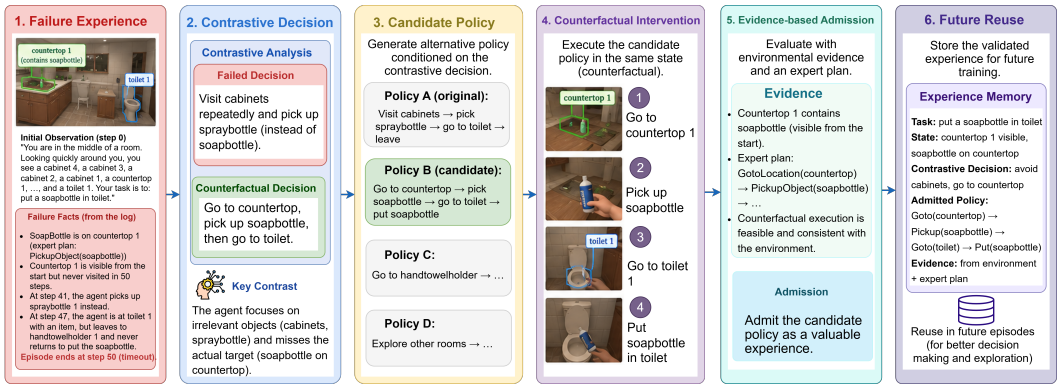

Figure 3: A real ALFWorld failure episode and the CPL walkthrough. The task is *put a soapbottle in toilet*, and the agent (`cpl`) fails by timeout after 50 steps. The initial observation already reveals *countertop 1* containing a soapbottle and *toilet 1*, but the agent repeatedly visits cabinets and eventually picks up a spray bottle instead of the soapbottle. CPL contrasts the failed trajectory with the expert plan, identifies the key decision difference, induces an intervention-testable candidate policy, validates it through a same-task counterfactual intervention, and admits the evidence-supported policy into memory for future reuse.

of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 5348–5375, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.292. URL https://aclanthology.org/2024.acl-long.292/.

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. doi: 10.1609/aaai.v38i17.29936.

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.

A Appendix

A.1 ALFWorld Failure Walkthrough

Figure 3 provides a concrete, step-by-step illustration of how CPL operates on a real ALFWorld failure episode. The task is *put a soapbottle in toilet*, and the agent (denoted `cpl`) fails by timeout after 50 steps. The episode exposes a characteristic failure mode: the agent repeatedly opens cabinets and eventually picks up a spray bottle, while the actual target—a soapbottle on countertop 1—is visible from the very first observation and is never visited.

(1) Failure experience. The initial observation at step 0 already reveals the relevant object: "a countertop 1" that contains a soapbottle, alongside a toilet 1. The expert plan is simple: `PickupObject(soapbottle)` followed by `PutObject(soapbottle, toilet)`. Instead, the agent spends its entire step budget exploring irrelevant locations. At step 41 it picks up a spray bottle, and at step 47 it reaches the toilet but with the wrong object, eventually timing out.

(2) Contrastive decision. CPL contrasts the failed trajectory with the successful (expert) plan. The key contrast is that the failed agent focuses on irrelevant objects (cabinets, spray bottle) and misses the actual target (soapbottle on countertop 1). This contrast yields a candidate decision difference: "avoid cabinets; go directly to countertop 1 and pick up the soapbottle."

(3) Candidate policy. The candidate difference is abstracted into an intervention-testable policy. Policy A is the original (failed) behavior: visit cabinets, pick up spray bottle, go to toilet, leave.

Policy B is the candidate: go to countertop 1, pick up the soapbottle, go to toilet, put the soapbottle. Policies C and D are alternative explorations (e.g., handtowels, other rooms).

(4) Counterfactual intervention. CPL executes the candidate policy in the same task state. The intervention replaces the failed decision with the policy-consistent action: go to countertop 1, pick up the soapbottle, go to toilet 1, and put the soapbottle in the toilet. When the environment supports replay, this intervention is executed in the actual environment rather than simulated by the LLM.

(5) Evidence-based admission. The intervention outcome is evaluated against environmental evidence and the expert plan. The evidence confirms that countertop 1 contains a soapbottle (visible from the start), that the expert plan is `GotoLocation(countertop)`→`PickupObject(soapbottle)`→ $\cdots$, and that the counterfactual execution is feasible and consistent with the environment. Because the policy passes the intervention gate, it is admitted as a valuable experience.

(6) Future reuse. The validated experience is stored in policy memory. On future tasks with similar decision contexts, the policy `Goto(countertop)`→`Pickup(soapbottle)`→`Goto(toilet)`→`Put(soapbottle)` can be retrieved, enabling better decision-making and exploration. This walkthrough illustrates the core CPL principle: observational contrast identifies what may matter, and same-task intervention verifies what is worth remembering.

A.2 LoCoMo Efficiency Comparison

To comprehensively analyze the overall performance of our proposed CPL framework, we conduct a systematic efficiency and effectiveness evaluation on the challenging LoCoMo long-term memory benchmark. Different from traditional memory benchmarks that only focus on prediction accuracy, LoCoMo requires models to balance answer quality, token overhead, and inference latency, which better reflects the practical deployment demands of real-world LLM memory systems. In this section, we report and analyze key quantitative metrics including exact-match (EM) accuracy, average input/output token consumption per query, total token usage, average inference latency, and per-query token cost, covering the core effectiveness and efficiency dimensions of existing memory-enhanced LLM methods.

As shown in Table 7, existing baseline methods exhibit obvious performance trade-offs between accuracy and efficiency. The iterative reasoning-based methods, including COT and EVOLVE-MEM, suffer from extreme inference latency overhead. Specifically, COT relies on multi-step chain-of-thought reasoning, resulting in a latency of up to 381,401 ms per query, which is three orders of magnitude higher than lightweight memory methods. Similarly, EVOLVE-MEM continuously evolves and optimizes memory content through iterative iteration, bringing massive token consumption (356,361 total tokens) and ultra-high latency, which severely limits its practical applicability. Most retrieval-based RAG variants, such as KG-RAG, Self-RAG, and RAG-BM25, maintain moderate token overhead but still retain high inference latency due to complex retrieval matching and post-processing logic, while their EM accuracy remains at a mediocre level of around 8%. In comparison, our CPL (ours) achieves an excellent balance between accuracy and practical efficiency. Although CPL introduces policy retrieval and decision-level memory processing mechanisms, leading to a relatively higher average input token count than simple baseline methods, its average inference latency is only 2,975 ms, which is comparable to the lightweight A-Mem method and far lower than all iterative reasoning and memory evolution baselines. Meanwhile, CPL obtains a competitive EM accuracy of 9.0%, outperforming LLM-Only, COT, EVOLVE-MEM, and Self-RAG. This fully demonstrates that the hierarchical policy learning and decision optimization mechanism of CPL can effectively improve long-term memory reasoning capability without introducing unbearable inference overhead, verifying the practical superiority of our method in real-time interactive scenarios.

A.3 ALFWorld Action-Match Evaluation

In real-world agent task scenarios, accurate memory reasoning is not only reflected in textual answer matching, but more importantly, determines whether the LLM agent can generate valid, environment-executable actions. To verify the practical usability and policy alignment capability of CPL beyond textual matching tasks, we further conduct action-match evaluation on 100 standard-

Table 7: Efficiency comparison on LoCoMo. AvgIn and AvgOut denote average input and output tokens per query, respectively. Tok/QA denotes total tokens per query.

Method	EM (%)	AvgIn	AvgOut	TotTok	AvgLat (ms)	Tok/QA
A-Mem	12.0	212	18	22977	2627	230
COT	2.0	66	97	16228	381401	162
CPL (ours)	9.0	820	25	84462	2975	845
EVOLVE-MEM	5.0	2367	1196	356361	206373	3564
KG-RAG	8.0	220	22	24231	90553	242
LLM-Only	1.0	20	23	4224	91331	42
RAG-BM25	8.0	226	18	24492	72855	245
Reflexion	11.0	318	38	35559	5326	356
Self-RAG	3.0	219	26	24422	153711	244
StructAligned	15.0	274	61	33558	8302	336

Table 8: Action-match accuracy on 100 ALFWorld instances.

Method	Action-Match (%)
A-Mem	99.80
RAG-BM25	99.40
KG-RAG	98.95
Mem0	98.73
Reflexion	98.20
CPL (preliminary)	95.78
CPL (ours)	95.60
Self-RAG	94.20
StructAligned	94.00
EVOLVE-MEM	91.32
COT	0.00
LLM-Only	0.00

ized ALFWorld interactive task instances. The action-match metric strictly measures the consistency between the model-generated executable policy actions and the standard reference actions, which can directly reflect whether the model's memory understanding and reasoning results can be accurately mapped to downstream agent execution behaviors, serving as a crucial indicator for evaluating memory-enhanced agent systems.

As illustrated in Table 8, traditional memory retrieval methods including A-Mem, RAG-BM25, and KG-RAG achieve the highest action-match accuracy, all exceeding 98%. These lightweight retrieval-based methods form stable and simple action generation logic by directly matching historical memory rules, thus performing well in fixed-pattern ALFWorld tasks. Our CPL (ours) achieves a promising action-match accuracy of 95.60%, which is only slightly lower than the preliminary version of CPL and the top-tier retrieval baselines, while significantly outperforming advanced methods such as Self-RAG (94.20%), StructAligned (94.00%), and EVOLVE-MEM (91.32%). This stable and high action matching performance indicates that the memory policy learned by CPL maintains strong consistency and executability in interactive agent tasks, and will not produce invalid or mismatched action decisions due to complex memory reasoning. Notably, vanilla LLM-Only and COT methods completely fail in this task with a 0.00% action-match rate. This phenomenon strongly proves that standalone large language models lack effective long-term memory management and policy induction capabilities. Without explicit memory modeling and decision guidance mechanisms, pure reasoning or original LLMs cannot understand task action specifications, resulting in completely invalid agent output. In contrast, the well-designed memory policy learning module of CPL effectively bridges long-term memory reasoning and downstream action execution, ensuring the practical effectiveness of memory enhancement in real agent deployment scenarios.

A.4 Additional Evaluation on LongMemEval

To fully dissect the adaptive performance and inherent limitations of the CPL framework in diverse long-term memory reasoning scenarios, we supplement comprehensive quantitative evalu-

ations on the LongMemEval dataset. Different from the task-oriented LoCoMo and ALFWorld benchmarks, LongMemEval subdivides long-term memory queries into six fine-grained categories covering knowledge update, multi-session interaction, single-session user/assistant preference, and temporal reasoning, which can comprehensively evaluate the model's capabilities in memory updating, cross-session association, user preference perception, and time-sensitive logic reasoning. This fine-grained categorization evaluation helps us accurately locate the applicable scenarios and optimization directions of our method. As summarized in Table 9, existing methods show distinct performance differentiation across different query types. The advanced EVOLVE-MEM achieves the best overall EM accuracy (9.80%), benefiting from its dynamic memory evolution mechanism that adapts to continuously updated knowledge and cross-session interactive content. Our CPL (ours) obtains a competitive overall EM score of 5.60%, surpassing the vanilla LLM-Only baseline (3.80%) and achieving comparable performance with the COT reasoning method (6.80%). In terms of fine-grained task performance, CPL exhibits significant advantages in temporal-reasoning queries, obtaining the highest accuracy among all compared baselines, which verifies that our decision-level memory optimization mechanism can effectively capture time-dependent memory logic and mine temporal association information in long-term dialogue sequences.

Meanwhile, the fine-grained results also clearly reveal the limitations of the CPL framework. CPL performs poorly on knowledge-update queries, with an accuracy of only 1.28%, which is significantly lower than EVOLVE-MEM and StructAligned. This indicates that our current policy learning mechanism lacks sufficient dynamic iteration and rapid overwrite adaptation capabilities when facing frequent memory content updates and knowledge revision scenarios. In addition, all methods achieve zero accuracy on single-session preference queries, which reflects the universal challenge for current long-term memory models in capturing implicit user preference information from limited single-round interactive content. In general, the LongMemEval ablation results validate the scenario advantages of CPL in temporal logic reasoning, and also point out clear future optimization directions for dynamic memory updating and implicit preference modeling.

Table 9: Exact-match accuracy (%) on LongMemEval by question type.

Method	Knowledge-Update	Multi-Session	Single-Session-Assistant	Single-Session-User	Temporal-Reasoning	Total
LLM-Only	5.13	3.01	3.57	5.71	3.76	3.80
COT	5.13	12.78	3.57	10.00	3.01	6.80
CPL (ours)	1.28	9.02	3.57	8.57	5.26	5.60
StructAligned	8.97	6.77	3.57	12.86	4.51	6.60
EVOLVE-MEM	11.54	14.29	5.36	14.29	6.02	9.80

B Detailed Experimental Setup

This appendix provides the implementation and evaluation details required to reproduce the experiments in the main paper. We follow the same evidence hierarchy throughout all experiments: L2 is used for candidate identification, L1 is used for intervention-based validation, and L3 is reserved for temporally later transfer evaluation. In particular, outcomes from L3 are never used to modify the admission decision.

B.1 Benchmarks

LoCoMo. We use LoCoMo (Maharana et al., 2024) to evaluate continual experience reuse and transfer across temporally ordered questions. The benchmark is treated as a non-interactive reasoning setting. Therefore, interventions performed on LoCoMo are interpreted as *decision-level intervention effects* rather than environmental causal effects. We report Semantic Similarity (Sim) for the main benchmark evaluation.

For the intervention stress test, we construct two matched intervention sets. The first set targets decisions that are explicitly supported by a retrieved policy (*policy-removal*). The second set randomly selects a decision from the corresponding trajectory (*random-removal*). Both sets contain 132 inter-

vention instances. The purpose of this comparison is not to maximize intervention sensitivity, but to examine whether the observed effect is localized to the decision identified by CPL.

ALFWorld. We use ALFWorld (Shridhar et al., 2020) as the primary executable environment for intervention validation. Unlike LoCoMo, ALFWorld permits the agent to execute alternative actions and observe the resulting environmental outcome. Therefore, intervention results on ALFWorld provide the strongest behavioral evidence in the current study.

The reported end-task evaluation contains 26 tasks. In addition, we construct 74 intervention points from failed trajectories. At each intervention point, the original decision is replaced by either the policy-guided alternative or a randomly selected valid alternative. The environment is then executed from the corresponding state.

B.2 Baselines

The main experiments compare CPL with the following representative approaches where corresponding results are available:

- **LLM-Only**: the base agent without external memory or retrieval;
- **CoT** (Wei et al., 2022): chain-of-thought prompting;
- **RAG-BM25** (Robertson & Zaragoza, 2009): lexical retrieval using BM25;
- **Self-RAG** (Asai et al., 2024): retrieval-augmented generation with self-reflection;
- **Reflexion** (Shinn et al., 2023): verbal reflection from previous trials;
- **Mem0** (Chhikara et al., 2025): persistent memory management;
- **EVOLVE-MEM** (Shah et al., 2025): continual memory evolution;
- **A-MEM** (Xu et al., 2026): structured agent memory;
- **Structurally Aligned** (Shen et al., 2026): structure-aware memory and retrieval;
- **AMA-Agent** (Zhao et al., 2026): agent memory and continual adaptation.

For intervention experiments, the primary control is not another memory system but a *random valid intervention*. This is important because the research question concerns decision localization rather than overall memory quality.

B.3 Evaluation Metrics

Semantic Similarity. For LoCoMo, we use the benchmark's Semantic Similarity (Sim) metric. Higher values indicate greater semantic agreement between the generated answer and the reference answer.

Success Rate. For ALFWorld, the success rate is defined as:

$$\text{Success} = \frac{N_{\text{successful episodes}}}{N_{\text{episodes}}}. \tag{1}$$

Intervention Effect. We define the intervention effect of a policy P as:

$$\text{IE}(P) = \mathbb{E}_{\tau \sim \mathcal{V}_P} \left[\text{Perf}(\tau) - \text{Perf}(\tau_{P\text{-violated}}) \right], \tag{2}$$

where $\mathcal{V}_P$ is the held-out validation set associated with policy P. A positive value means that violating the policy decreases performance under the tested intervention.

Recovery Rate. The recovery rate after intervention is defined as:

$$\text{Recovery} = \frac{N_{\text{recovered}}}{N_{\text{intervention points}}}. \tag{3}$$

Reuse Performance. The reuse performance metric is defined as:

$$\text{ReusePerf} = \frac{\#\{T_{\text{new}} : y(\pi(\cdot \mid \mathcal{M} \cup P)) \neq y(\pi(\cdot \mid \mathcal{M}))\}}{\#\{T_{\text{new}} : P \text{ is retrieved}\}}. \tag{4}$$

If no policy is admitted or no policy is retrieved, ReusePerf is not defined. Such cases are reported as "–" rather than being assigned a numerical value.

Retrieval Coverage. Retrieval coverage is calculated as:

$$\text{Coverage} = \frac{N_{\text{queries with retrieved policy}}}{N_{\text{evaluation queries}}}. \tag{5}$$

B.4 Policy Admission Configuration

The admission confidence score is defined as:

$$\text{Conf}_{\text{adm}}(P) = w_1\overline{\text{Rel}}(P) + w_2\widehat{\text{IE}}(P) + w_3\text{Cons}(P) + w_4\text{Evidence}(P). \tag{6}$$

The default weight configuration used in all experiments is:

$$(w_1, w_2, w_3, w_4) = (0.25, 0.40, 0.20, 0.15). \tag{7}$$

The intervention effect is additionally used as a hard admission gate:

$$\text{IE}(P) > 0. \tag{8}$$

Therefore, the complete admission rule is

$$P \in \mathcal{M} \iff [\text{IE}(P) > 0] \wedge [\text{Conf}_{\text{adm}}(P) \geq \theta_{\text{adm}}]. \tag{9}$$

This distinction is important. Intervention evidence is not merely another ranking feature: a candidate with non-positive intervention effect cannot be admitted even when its observational relevance is high.

B.5 Data Isolation and Temporal Ordering

CPL uses three evidence levels:

- **L2**: successful–failed contrast for candidate generation;
- **L1**: same-task intervention for policy validation;
- **L3**: temporally later unseen tasks for transfer evaluation.

L2 and L1 operate on experiences available before policy admission. L3 is strictly downstream of admission. In particular, the outcome of an unseen task is never used to increase, decrease, or otherwise modify $\text{Conf}_{\text{adm}}(P)$.

This separation prevents temporal leakage from making a policy appear validated simply because it performed well on the tasks on which it was later evaluated.

C Algorithm

This section gives the complete procedural description of CPL. The algorithm separates candidate discovery, policy induction, intervention validation, admission, and subsequent reuse.

C.1 Overall Procedure

Given an ordered stream of agent experiences, CPL first identifies decision-level differences between successful and failed experiences. These differences are converted into structured policies. Each policy is then tested by intervention on held-out validation trajectories. Only policies that satisfy both the intervention gate and the confidence threshold are stored.

Algorithm 1 Counterfactual Policy Learning

Require: Experience stream $\mathcal{E}$, admission threshold θ_{adm}, relevance threshold θ_{rel}
Ensure: Persistent policy memory $\mathcal{M}$

Initialize $\mathcal{M} \leftarrow \emptyset$
for each task type k **do**
  Collect successful experiences $\mathcal{E}_k^+$ and failed experiences $\mathcal{E}_k^-$
  Align states and decisions between $\mathcal{E}_k^+$ and $\mathcal{E}_k^-$
  Generate candidate decision differences $\mathcal{D}_k$
  for each candidate $d \in \mathcal{D}_k$ **do**
    Estimate observational relevance $\mathrm{Rel}(d)$
    if $\mathrm{Rel}(d) < \theta_{\mathrm{rel}}$ **then**
      Discard d
    else
      Convert d into intervention-testable policy $P = (C, G, S, A, R, E)$
      Construct held-out validation trajectories $\mathcal{V}_P$
      Construct violated trajectories $\tau_{P\text{-violated}}$ by replacing the policy-supported decision
      Execute intervention and estimate $\widehat{\mathrm{IE}}(P)$
      Compute consistency $\mathrm{Cons}(P)$
      Compute evidence support $\mathrm{Evidence}(P)$
      Compute

$$\mathrm{Conf}_{\mathrm{adm}}(P) = 0.25\overline{\mathrm{Rel}}(P) + 0.40\widehat{\mathrm{IE}}(P) + 0.20\mathrm{Cons}(P) + 0.15\mathrm{Evidence}(P)$$

      if $\widehat{\mathrm{IE}}(P) > 0 \wedge \mathrm{Conf}_{\mathrm{adm}}(P) \geq \theta_{\mathrm{adm}}$ **then**
        Add P to persistent memory $\mathcal{M}$
      else
        Reject P
      end if
    end if
  end for
end for
Retrieve policies from $\mathcal{M}$ only for later unseen tasks **return** $\mathcal{M}$

C.2 Candidate Generation

For each task type, successful and failed experiences are aligned according to semantic similarity between their states. A candidate is formed when the aligned states exhibit different decisions:

$$s_t^+ \approx s_j^-, \qquad a_t^+ \neq a_j^-. \tag{10}$$

The candidate is retained only when its relevance exceeds θ_{rel}. The relevance score is treated as an observational quantity. It does not enter the paper's causal interpretation.

C.3 Policy Induction

Each candidate decision is transformed into the structured representation

$$P = (C, G, S, A, R, E), \tag{11}$$

where:

- C specifies the contextual condition under which the policy applies;
- G specifies the goal associated with the decision;
- S specifies the strategy or decision principle;
- A specifies the concrete action to be taken;
- R specifies restrictions or conditions under which the action should not be applied;
- E records the source experiences supporting the policy.

The policy must contain an explicit replaceable action. A statement that cannot be operationalized as a changed decision is not intervention testable and therefore cannot pass the L1 validation stage.

C.4 Intervention Construction

For each policy-supported decision, CPL constructs a violated version by changing the target decision while preserving the task and preceding trajectory as much as the benchmark permits.

For an executable environment, the replacement action is executed from the corresponding environment state. For a non-interactive benchmark, the intervention is applied to the decision or reasoning operation and the resulting answer is evaluated.

Random intervention is used as a control. A random intervention is sampled from the set of valid alternatives available at the same decision point. The purpose is to determine whether the effect is concentrated on the candidate decision rather than being a generic consequence of modifying any trajectory component.

C.5 Policy Admission

A policy is admitted only when both of the following conditions hold:

$$\widehat{\mathrm{IE}}(P) > 0 \tag{12}$$

and

$$\mathrm{Conf}_{\mathrm{adm}}(P) \geq \theta_{\mathrm{adm}}. \tag{13}$$

Policies failing either condition are discarded.

This design intentionally allows the persistent memory to remain empty. An empty memory is preferable to retaining a policy whose intervention evidence is insufficient.

C.6 Policy Retrieval

For a temporally later task T_{new}, the agent first identifies the task type and retrieves policies whose contextual conditions are compatible with the current state. Retrieved policies are provided as policy context to the agent.

The retrieval stage is not used to revalidate policies. A policy that has already passed admission remains fixed during the corresponding evaluation episode. This avoids using test-time outcomes to retrospectively justify the policy.

D Prompt Templates

This section describes the prompt roles used by CPL. The prompts are structured so that the language model proposes candidate interpretations and interventions, while the final evidence is determined by the measured outcome of the intervention.

D.1 Decision Relevance Prompt

The relevance evaluator receives an aligned successful–failed pair and determines whether the decision difference is potentially relevant to the divergent outcomes.

Task: Decision Relevance Evaluation
You are given two experiences from the same task type.
Successful experience: [successful trajectory]
Failed experience: [failed trajectory]
Identify the decision-level difference that best explains the divergence between the two experiences.
Return:

1. the relevant state or decision point;
2. the successful decision;

3. the failed decision;
4. a relevance score in $[0, 1]$;
5. a brief explanation of why the decision difference is potentially relevant.

Do not claim that the decision caused the outcome; this evaluation is intended only for candidate generation.

This final instruction is important because this stage relies on observational evidence. The evaluator is not asked to infer a causal relationship.

D.2 Policy Induction Prompt

For a selected candidate, the model converts the decision difference into a structured policy:

Task: Policy Induction
Given the candidate decision difference below, formulate an intervention-testable policy.
Candidate decision: [candidate]
Supporting experiences: [evidence]
Return the following fields:

1. Context (C): when this policy is applicable;
2. Goal (G): what the agent is trying to accomplish;
3. Strategy (S): the decision principle;
4. Action (A): the action that can be executed;
5. Restriction (R): conditions under which the action should not be applied;
6. Evidence (E): identifiers of supporting experiences.

The action must be explicitly replaceable so that the policy can later be tested by intervention.

D.3 Intervention Construction Prompt

The intervention generator receives a validated policy and a trajectory containing a corresponding decision:

Task: Construct an Intervention
Given the current state and the candidate policy, identify the action specified by the policy and construct a valid alternative action that violates the policy.
Current state: [state]
Original action: [action]
Policy: [policy]
Return:

1. the target decision;
2. the policy-consistent action;
3. the policy-violating alternative;
4. the reason why the alternative violates the policy.

Do not modify preceding states or unrelated decisions.

For executable environments, the generated alternative is additionally checked against the environment's valid-action constraints before execution.

D.4 Policy Admission Prompt

The admission stage does not ask the model to decide whether a policy is "causal." Instead, structured evidence is used to compute the admission score.

Task: Policy Evidence Assessment
Given a candidate policy and its supporting evidence, assess:

1. consistency of the decision direction;
2. amount of supporting evidence;
3. agreement between the induced policy and its source experiences.

Return structured values for $\mathrm{Cons}(P)$ and $\mathrm{Evidence}(P)$ in $[0, 1]$.
Do not use outcomes from future unseen tasks. Do not infer intervention effects from observational evidence.

The final admission decision is then computed by the deterministic admission rule described in the Algorithm appendix.

D.5 Retrieval Prompt

For later tasks, retrieved policies are presented as contextual guidance:

Task: Solve the Current Task
Current task: [current task]
Current state: [current state]
Retrieved policy: [policy]
Use the retrieved policy only when its contextual conditions match the current situation. If the policy is not applicable, ignore it and solve the task independently.
Return the action or answer required by the environment.

This instruction prevents a retrieved policy from being treated as an unconditional rule.

E Intervention Protocol

E.1 General Intervention Principle

The purpose of intervention is to test a decision under a controlled change. For each candidate policy, CPL constructs a pair:

$$(\tau, \tau_{P\text{-violated}}), \tag{14}$$

where the two trajectories share the same task and preceding context as closely as the benchmark permits, while the target decision differs.

The intervention therefore follows:

$$\text{same task} \rightarrow \text{same preceding context} \rightarrow \text{different target decision} \rightarrow \text{different observed outcome}. \tag{15}$$

Only the target decision is intended to change. If the benchmark forces subsequent state transitions to differ, those downstream changes are treated as consequences of the intervention rather than additional manually modified variables.

E.2 ALFWorld Protocol

ALFWorld is used as the primary intervention environment because actions can be executed and their environmental consequences can be observed.

The intervention procedure is:

1. Run the agent until a failed trajectory is obtained.
2. Identify an incorrect or policy-relevant decision point.
3. Retrieve or construct the corresponding policy action.
4. Replace the original decision with the policy-guided action.
5. Continue execution from the corresponding environment state.

Table 10: ALFWorld intervention protocol and reported outcomes.

Condition	Intervention Points	Recoveries	Recovery Rate (%)
Policy-guided	74	8	10.8
Random valid	74	0	0.0

Table 11: LoCoMo intervention stress test.

Condition	N	Mean IE	Flip Rate (%)
Policy-removal	132	0.167	16.7
Random-removal	132	0.197	19.7
Difference	–	-0.030	-3.0

6. Record whether the episode is recovered.

The same procedure is applied to the random control condition, except that the replacement is sampled randomly from valid alternatives.

The reported intervention set contains 74 failed decision points. Policy-guided intervention recovers 8 of these points, corresponding to a 10.8% recovery rate, whereas random valid intervention recovers none.

Because the intervention is executed in the environment, this experiment provides behavioral evidence under the tested states. It does not establish a general causal law for the entire task distribution.

E.3 LoCoMo Protocol

LoCoMo does not provide the same executable environment interface as ALFWorld. We therefore use a decision-removal protocol rather than an environmental replay protocol.

Two intervention conditions are constructed:

1. **Policy-removal**: remove or alter a decision that is explicitly supported by a retrieved policy;
2. **Random-removal**: remove or alter a randomly selected decision from the corresponding reasoning trajectory.

Each condition contains 132 intervention instances. The resulting values are summarized below.

The random-removal condition produces a larger mean intervention effect than policy-removal. Consequently, this experiment is treated as a stress test rather than evidence of successful policy localization.

E.4 Candidate-Level Intervention Test

The L2 candidate analysis further illustrates the distinction between candidate relevance and intervention effectiveness.

The reported candidate pool contains 534 L2 candidates, of which 389 (72.8%) can be converted into intervention-testable policies. The L2 candidate set has a mean relevance of 0.472, compared with 0.373 for random candidates. The proportion with relevance at least 0.6 is 27.2% for L2 candidates and 8.4% for random candidates.

However, positive intervention effects occur for only 4/389 L2 candidates (1.0%), compared with 8/512 random candidates (1.6%). The reported Fisher exact test gives $p = 0.57$.

These results motivate the separation between L2 and L1. Contrastive selection improves candidate relevance, but relevance alone does not predict intervention success.

Table 12: Candidate relevance and intervention validation.

Candidate Source	Mean Rel.	Rel.≥ 0.6	Positive IE	Testable
Random	0.373	8.4%	1.6% (8/512)	512
L2 Contrastive	**0.472**	**27.2%**	1.0% (4/389)	389

E.5 Intervention Control Principle

The intervention control is designed to answer a narrow question:

> Does changing the decision identified by the candidate policy produce a different outcome more consistently than changing an arbitrary decision?

A positive policy intervention effect is therefore interpreted jointly with the random control. A large effect from both policy-targeted and random interventions indicates that the task is generally sensitive to decision perturbations and weakens the evidence for localization.

F Statistical Analysis

F.1 Descriptive Statistics

All main intervention results are first reported using descriptive statistics. For binary outcomes, we report the number of successful interventions together with the corresponding proportion. For continuous intervention effects, we report the mean IE.

For the ALFWorld intervention, the policy-guided condition recovers $8/74$ decision points, whereas the random valid control recovers $0/74$. The observed difference in recovery rate is

$$0.108 - 0 = 0.108.$$

We report this as an observed difference in the tested intervention set, rather than as a population-level causal effect.

F.2 Fisher Exact Test for Candidate Selection

For the L2 candidate analysis, we compare the proportion of positive intervention effects between contrastive candidates and random candidates. The observed counts are:

	Positive IE	Non-positive IE
L2 Contrastive	4	385
Random	8	504

We use Fisher's exact test because the number of positive intervention effects is small. The reported two-sided test gives

$$p = 0.57.$$

Thus, the current data do not provide evidence that L2 contrastive selection increases the probability of a positive intervention effect. The corresponding result is consistent with our interpretation of L2 as a candidate-generation mechanism rather than a validation mechanism.

F.3 Confidence Interval for Retrieval Performance

For the nested random-retrieval ablation, the reported ReusePerf is 43.81%, corresponding to 301 successful cases out of 687 evaluation questions. The reported 95% confidence interval is

$$[40.10\%, 47.52\%].$$

The confidence interval is used to quantify uncertainty around the observed proportion. It is not used as an admission criterion.

F.4 Intervention Effect Comparison

For LoCoMo, the observed intervention means are

$$0.167 \quad \text{and} \quad 0.197.$$

The observed difference is

$$0.167 - 0.197 = -0.030.$$

The negative difference indicates that the policy-targeted intervention was not more influential than random intervention in this setting. Because the current experiment does not report a paired bootstrap interval or a formal paired significance test for these continuous intervention effects, we do not attach a significance claim to the difference.

F.5 Small-Sample Interpretation

Several experiments in this study contain relatively small sample sizes. In particular, the ALFWorld end-task evaluation contains 26 episodes and the intervention experiment contains 74 intervention points. We therefore avoid interpreting small absolute differences as stable population-level effects.

For binary intervention outcomes, we emphasize the raw counts in addition to percentages. For example, "$8/74$" is reported alongside 10.8%, rather than reporting 10.8% alone.

F.6 Multiple-Comparison Considerations

The ablation and threshold analyses are primarily descriptive. They are used to characterize the behavior of the admission mechanism under controlled changes rather than to establish a family of confirmatory hypotheses.

Consequently, we do not treat the threshold sweep as a collection of independent hypothesis tests and do not select a threshold based on the highest observed ReusePerf alone. The threshold experiment is reported as a sensitivity analysis, while the main admission configuration is kept separate from the sweep.

F.7 Statistical Reporting Principle

The statistical analysis follows three principles:

1. **Report counts with proportions** when the sample is small;
2. **Use an exact test** for the sparse candidate-level intervention comparison;
3. **Separate statistical uncertainty from policy admission**, so that confidence intervals and future-task outcomes cannot leak into the admission mechanism.

The purpose of the statistical analysis is therefore to quantify uncertainty and test the specific comparisons supported by the current data, rather than to convert the observed intervention results into a broader causal claim.

G Additional Admission and Sensitivity Results

G.1 Admission Threshold

We evaluate a separate pool of ten candidate policies under several admission thresholds.

The results show a coverage–reliability trade-off. Increasing the threshold reduces the number of admitted policies. We do not interpret the threshold sweep as evidence that any particular threshold is universally optimal.

G.2 Evidence-Gate Ablation

The complete system and ablated variants are evaluated on 687 unseen questions.

Table 13: Admission-threshold sensitivity on a separate pool of 10 candidate policies.

θ_{adm}	Admission	Coverage (%)	ReusePerf (%)	Survival (%)
0.1	10/10	100.0	96.2	30.0
0.2	10/10	100.0	96.2	30.0
0.3	8/10	80.0	96.2	37.5
0.5	7/10	70.0	100.0	41.2
0.6	1/10	10.0	100.0	11.5
0.7	0/10	0.0	–	–

Table 14: Ablation of L2 candidate generation, L1 intervention validation, and L3 retrieval.

Variant	Admission	Coverage (%)	ReusePerf (%)	Δ
CPL (Full)	0/17	0.00	–	–
w/o L1	6/17	27.07	53.86	-3.35
w/o L2	0/17	0.00	–	–
w/o L3	0/17	0.00	–	–
w/o L1+L3	6/17	100.00	43.81	-13.40

The full CPL configuration admits no policy in the reported 17-policy run. Therefore, ReusePerf is undefined for the full configuration in this particular admission experiment. This should be distinguished from the separate benchmark-level LoCoMo result.

Removing L1 admits six policies, with a reported ReusePerf of 53.86%. Three of the six admitted policies have negative intervention effects. This provides evidence that intervention validation acts as a conservative admission gate.

The nested w/o L1+L3 condition retains the same non-empty policy memory but replaces task-type-aware retrieval with random retrieval. ReusePerf decreases to 43.81%, indicating that retrieval constraints can reduce negative transfer after policies have been admitted.

G.3 Interpretation of the Sensitivity Results

The sensitivity experiments should be interpreted jointly with the main intervention results.

First, increasing the admission threshold makes policy retention more conservative. Second, removing L1 increases policy coverage but permits policies with negative intervention effects. Third, removing task-type matching from retrieval further decreases transfer performance.

These results do not establish that every retained policy is universally correct. They instead characterize how the proposed evidence gate changes the precision–coverage behavior of persistent policy memory.

H Reproducibility Checklist

For reproducibility, the implementation should preserve the following experimental constraints:

1. L2 successful–failed contrasts are used only for candidate generation.
2. L1 intervention validation is performed on held-out validation trajectories rather than the trajectories used to induce the policy.
3. The admission threshold is fixed before evaluating temporally later tasks.
4. L3 outcomes are never used to modify admission.
5. Random intervention controls use valid alternative decisions.
6. ReusePerf is computed only after policy admission.
7. When no policy is admitted, ReusePerf is reported as undefined rather than assigning an artificial score.

Table 15: Evidence boundaries in CPL.

Level	Role	Supported interpretation
L2	Candidate generation	Observational relevance; identifies decisions worth testing.
L1	Intervention validation	Behavioral intervention evidence under the tested conditions.
L3	Transfer evaluation	Whether an already-admitted policy transfers to later unseen tasks.

8. Small-sample intervention results are reported using both raw counts and percentages.

The resulting experimental chain is:

$$\boxed{\text{Experience} \rightarrow \text{L2 Contrast} \rightarrow \text{Candidate} \rightarrow \text{L1 Intervention} \rightarrow \text{Admission} \rightarrow \text{L3 Reuse Evaluation}} \tag{16}$$

This ordering is the central reproducibility constraint of CPL. In particular, reversing the L1 and L3 stages would introduce temporal leakage and would change the interpretation of the method.

I Summary of Evidence Boundaries

For clarity, the empirical claims supported by each evidence level are summarized below.

The distinction is deliberate. L2 does not establish causality, L3 cannot validate a policy retrospectively, and only executable interventions in the environment support the strongest intervention-based behavioral interpretation used by the paper.