Neural Procedural Memory: Empowering LLM Agents with Implicit Activation Steering

Chengfeng Zhao[1,2] Yuqiao Tan[1,2] Shizhu He[1,2,3] Yequan Wang[3]* Jun Zhao[1,2] Kang Liu[1,2]†

[1]The Key Laboratory of Cognition and Decision Intelligence for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]Beijing Academy of Artificial Intelligence, Beijing, China
{zhaochengfeng2024, tanyuqiao2025}@ia.ac.cn,
{shizhu.he, kliu, jzhao}@nlpr.ia.ac.cn, tshwangyequan@gmail.com

Abstract

While Large Language Models (LLMs) excel as static solvers, transforming them into autonomous agents remains challenging. This transition requires continuous environmental interaction, yet current agents lack the necessary persistent procedural memory. Existing approaches predominantly employ Retrieval-Augmented Generation (RAG) to inject explicit textual guidelines into model contexts. However, relying solely on symbolic instructions can introduce a text-action disconnect, frequently failing to activate the internal representations necessary for correct task execution. To address this, the paper introduces **Neural Procedural Memory (NPM)**, a training-free framework that represents agent memory through implicit activation steering rather than explicit instructions. By distilling procedural skills from historical contrastive experiences into steering vectors in the activation space, NPM directly activates the task-relevant neural mechanisms to guide task execution. Evaluations across four agent benchmarks show that NPM performs comparably to baselines using explicit textual instructions. Furthermore, the results show that combining implicit steering with explicit workflows provides complementary advantages, leading to more robust task execution. Representational analyses indicate that these steering vectors encode consistent task logic, forming organized structures within the activation space. These findings suggest that implicit activation steering provides a promising approach for managing agent memory.

1 Introduction

As Large Language Models (LLMs) evolve from stateless reasoners to the central controllers of autonomous agents, their operational scope has expanded to open-ended, dynamic environments—ranging from web navigation to embodied interaction (Wang et al., 2023, 2024a; Liu et al., 2025; Tan et al., 2026a). Succeeding in these complex settings requires agents to leverage persistent memory to transform past experiences into reusable skills, making memory a critical component of long-term autonomy (Park et al., 2023; Zhong et al., 2024; Omidi et al., 2025; Hu et al., 2025).

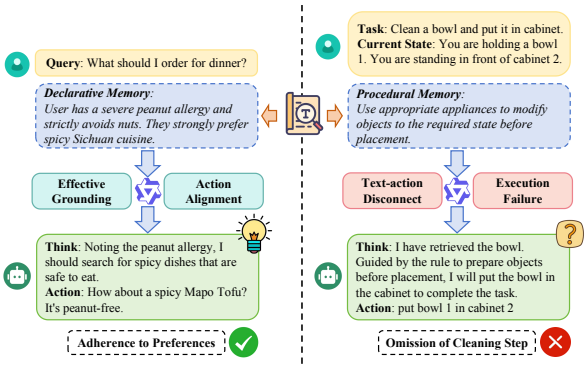


Figure 1: **Comparison between Declarative and Procedural Memory in LLM Agents. Left**: Declarative memory successfully grounds reasoning in static factual knowledge, enabling the agent to strictly adhere to explicit user constraints. **Right**: Textual procedural memory can introduce a text-action disconnect where the agent struggles to align the retrieved workflow onto the execution trajectory and omits intermediate steps.

Agent memory functionally consists of declarative memory for descriptive facts (Wang et al., 2024b; Gutierrez et al., 2024; Chhikara et al., 2025) and procedural memory for executing action sequences (Zheng et al., 2023; Zhao et al., 2024; Yu et al., 2025a). While declarative recall is straightforward, effectively representing and utilizing procedural memory remains a bottleneck (Han et al., 2025; Zhang et al., 2025c). Traditional methods typically embed procedural knowledge offline into model parameters via supervised fine-tuning (Shao et al., 2023; Zhang et al., 2025b) or reinforcement learning (Yao et al., 2024; Tan et al., 2026c,b). However, these parameter-updating approaches in-

*Corresponding Author
†Corresponding Author

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 18107–18128
October 24-29, 2026 ©2026 Association for Computational Linguistics

cur high computational overhead and struggle to adapt in real time to dynamic environments (Zhai et al., 2023; Xia et al., 2024; Luo et al., 2025).

By contrast, recent studies have focused on the second paradigm: online external knowledge retrieval, most notably Retrieval-Augmented Generation (RAG). Although successful with declarative memory, such as fact retrieval (Zhong et al., 2024; Packer et al., 2024; Chhikara et al., 2025), this solution encounters limitations when applied directly to procedural memory. Unlike static facts, procedural memory is inherently implicit and abstract. Such knowledge is often ineffable: compressing complex reasoning patterns into discrete natural language tokens inevitably leads to information loss (Mahowald et al., 2024). Relying exclusively on explicit textual descriptions to transfer procedural skills can lead to a text-action disconnect. Agents might comprehend retrieved instructions yet fail to strictly act upon them (Wen et al., 2024; Sun et al., 2025; Geng et al., 2025). As shown in Figure 1, while declarative memory successfully grounds static constraints, conveying procedural workflows through text fails to translate into sequential actions, leading agents to omit critical intermediate steps during execution. When relying solely on texts as a reasoning medium, symbolic descriptions of actions can sometimes fail to consistently elicit the targeted behavioral trajectories required for task completion.

Therefore, it is intuitive that procedural memory could benefit from mechanisms more closely aligned with intrinsic behavioral modulation rather than relying solely on external text. Such a perspective is corroborated by cognitive neuroscience (Squire, 1992, 2004), which posits that procedural memory is non-verbalizable and manifests through the modulation of neural activity rather than explicit declarative recall.

Following this insight, we propose **Neural Procedural Memory (NPM)**, a training-free framework that introduces an alternative paradigm of agent memory based on implicit activation steering. Instead of retrieving text for the agent to read, NPM identifies relevant historical tasks and extracts procedural signals from contrastive experiences into steering vectors within the activation space. Specifically, we employ a dual-granularity strategy to address the challenge of agents failing to obtain successful trajectories in complex tasks: inter-trajectory contrast aligns successful and failed trajectories, while intra-trajectory contrast exploits step-level differences within individual failed attempts. During inference, the steering vectors are dynamically synthesized and injected into the residual stream, directly modulating the agent's reasoning process and action selection without expanding the context window or updating parameters.

We evaluate NPM across four agent benchmarks including ALFWorld (Shridhar et al., 2021), WebShop (Yao et al., 2022), ScienceWorld (Wang et al., 2022), and BabyAI (Chevalier-Boisvert et al., 2019). Results demonstrate that implicit activation steering achieves performance comparable to explicit textual memory baselines and exhibits complementary synergy when combined with explicit textual workflows. While textual memory provides high-level symbolic guidance, implicit steering reinforces procedural adherence directly within the neural representation. We further dissect the extracted vectors to reveal semantically meaningful behavioral primitives, providing an interpretable view of how procedural knowledge is represented in activation space. These findings suggest implicit activation steering as an effective path for representing procedural memory in LLM agents. The code is available at `https://github.com/JJJAYYYZhao/NPM-vllm-version`.

Our contributions are summarized as follows:

- This paper proposes Neural Procedural Memory (NPM), a training-free framework that integrates implicit activation steering into agent memory, enabling effective intrinsic behavioral modulation.
- This paper introduces a dual-granularity extraction method using both inter-trajectory and intra-trajectory contrasts, offering a flexible intervention mechanism and mitigating the reliance on successful demonstrations.
- Extensive experiments demonstrate that activation steering serves as an effective carrier for agentic skills, yielding interpretable vectors that correlate with specific task behaviors.

2 Related Work

Functions of Agent Memory Agent memory is typically categorized into factual (declarative) and experiential (procedural) components (Hu et al., 2025). While Retrieval-Augmented Generation (RAG) effectively maintains declarative facts (Hu et al., 2025; Park et al., 2023), it faces limitations

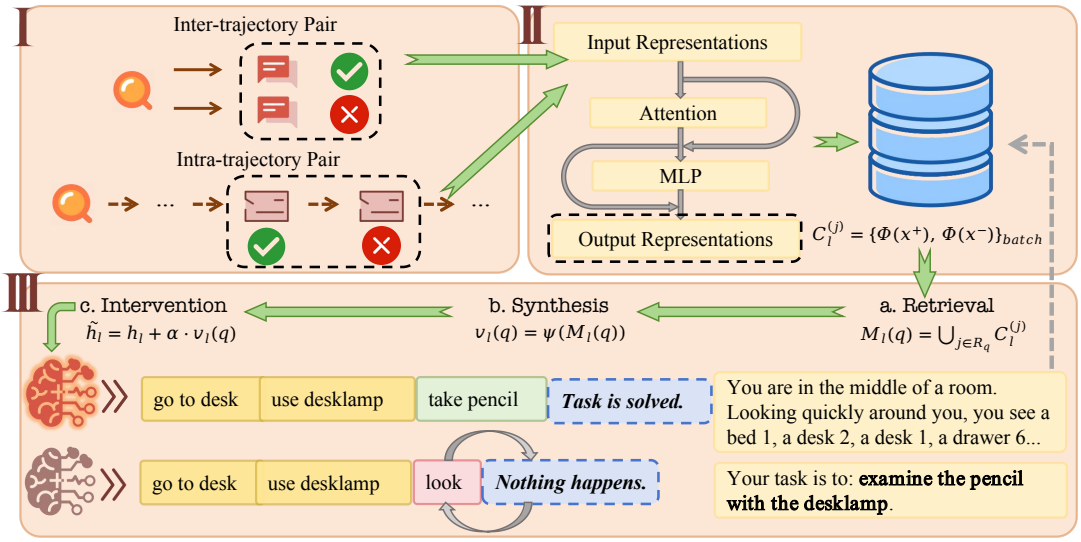


Figure 2: **Overview of the Neural Procedural Memory (NPM) framework.** (1) **Contrastive Experience Construction:** Formulating dual-granularity (inter- and intra-trajectory) contrastive pairs from historical interactions. (2) **Procedural Memory Extraction:** Extracting continuous representations from these pairs to construct a historical memory repository. (3) **Inference-Time Intervention:** Retrieving relevant experiences to dynamically synthesize a steering vector, which is injected into the residual stream to guide agent reasoning.

when applied to procedural memory. Existing approaches typically approximate procedural memory by injecting textual guidelines or code snippets into the context window (Shinn et al., 2023; Wang et al., 2023). However, this explicit approach suffers from a text-action disconnect: natural language serves as an inefficient proxy for encoding internal neural representations (Mahowald et al., 2024). Moreover, the reliance on lengthy text instructions increases computational cost and degrades performance in long-horizon tasks (Geng et al., 2025; Sinha et al., 2025). NPM shifts from explicit verbalization to implicit activation modulation, eliminating the need for in-context instructions.

Forms of Agent Memory Current approaches explore various representational modalities. Token-level methods (e.g., RAG) offer flexibility but are bounded by context window limits and inference latency. Conversely, parametric approaches encode knowledge within model weights permanently (Zhang et al., 2025b; Wang et al., 2025a), offering efficiency but lacking the flexibility required for rapid adaptation or unlearning without retraining. Latent memory offers a middle ground by storing information as continuous vectors. However, existing works primarily focus on context compression (Chevalier et al., 2023; Wang et al., 2024c) or transient working memory (Zhang et al., 2025a; Yu et al., 2025b). For instance, MemGen (Zhang et al., 2025a) synthesizes latent tokens during reasoning but requires auxiliary encoders and extensive training. While NPM shares the underlying concept of latent memory, it views memory storage as activation steering, enabling representation-level modulation of the generation path without additional token overhead or parameter updates.

Representation Engineering and Activation Steering. Our framework builds upon Representation Engineering (RepE) that monitors and manipulates high-level cognitive states in the activation space (Zou et al., 2025). Prior studies (Rimsky et al., 2024; Lee et al., 2025) have demonstrated the efficacy of steering models toward coarse-grained, static attributes such as honesty or harmlessness. However, existing RepE work is predominantly limited to single-turn global concepts. NPM extends these techniques to dynamic, task-specific agentic trajectories. By introducing a retrieval-augmented framework where steering vectors are synthesized from contrastive historical experiences, we enable agents to adaptively recall and apply procedural skills specific to the task context.

3 Neural Procedural Memory

We introduce **Neural Procedural Memory (NPM)**, a training-free framework designed to internalize agentic skills directly into the model's activation space. The framework shown in Figure 2 consists of three phases: (1) **Contrastive Experience Construction** (§3.1), which isolates

effective reasoning patterns from historical failures; (2) **Steering Vector Extraction** (§3.2), which distills discrete textual contrasts into continuous steering vectors; and (3) **Inference-Time Intervention** (§3.3), where these vectors are dynamically synthesized and injected to modulate the agent's behavior.

3.1 Contrastive Experience Construction

Formally, given a task query q and a task environment $\mathcal{E}$, an agent generates a trajectory $\tau = \{s_1, a_1, \ldots, s_K, a_K\}$, where s_k denotes the state and a_k denotes the action at step k. The objective is to construct a contrastive dataset $\mathcal{D}$ composed of paired reasoning segments (x^+, x^-), where x^+ represents a desirable reasoning mode and x^- represents a degenerate one. A dual-granularity strategy is employed to capture procedural signals at both the inter-trajectory and intra-trajectory levels.

3.1.1 Inter-trajectory Contrast

For tasks where the agent has historically generated both successful and failed attempts, we aim to capture the global behavioral shift. Let $\mathcal{T}^+$ and $\mathcal{T}^-$ denote the sets of successful and failed trajectories for a specific task, respectively. A contrastive pair is formed by aligning a failed attempt with a successful one:

$$\mathcal{P}_{\text{inter}} = \left\{(\tau^+, \tau^-) \mid \tau^+ \in \mathcal{T}^+, \tau^- \in \mathcal{T}^-\right\}. \quad (1)$$

This contrast directionally distinguishes successful trajectories from failed attempts.

3.1.2 Intra-trajectory Contrast

In many complex tasks, the agent frequently fails to obtain successful trajectories, resulting in a sparse distribution of positive samples. To enable learning from failure, we exploit step-level differences within a single failed trajectory. A **Degenerate Step** set $S_{\text{deg}} \subset \tau$ is defined to contain actions that exhibit irrational behaviors or violate environmental constraints, identified via the following criteria:

- **Redundancy:** Consecutive repetition of an identical command, indicating a heuristic loop or failure to progress without state change.
- **Invalidity:** Actions that trigger environment-specific error feedback, such as format violations or inadmissible moves.

Conversely, an **Effective** step set $S_{\text{eff}} = \tau \setminus S_{\text{deg}}$ is defined to contain steps validly advancing the state.

Instead of pairing individual steps, which serves as a noisy signal, we aim to contrast the collective effective behavior against the degenerate one within the trajectory. The contrastive pair is constructed by grouping the steps into two sets:

$$\mathcal{P}_{\text{intra}} = \left\{(S_{\text{eff}}, S_{\text{deg}}) \mid S_{\text{deg}} \neq \emptyset\right\}. \quad (2)$$

This intra-trajectory contrast isolates specific degenerate modes from the otherwise valid reasoning process. (Quantitative analysis of heuristic isolation rules is provided in Appendix C.2).

3.2 Procedural Memory Extraction

The core of NPM is to map the textual contrast defined in $\mathcal{D}$ into a geometric transformation in the continuous representation space. The goal is to derive a vector $\mathbf{v} \in \mathbb{R}^d$ representing the shift from a degenerate state $\mathbf{h}^-$ to a desirable one $\mathbf{h}^+$.

Hidden State Representation. We define the representation function $\phi_l(\cdot)$ to extract a fixed-dimensional representation vector $\mathbf{h} \in \mathbb{R}^d$ depending on the contrast granularity.

- *Inter-trajectory:* For this comparison, we utilize the sequence of hidden states of the full trajectory τ and extract the last token to capture the accumulated context for each layer.
- *Intra-trajectory:* Since reasoning steps span multiple tokens, we apply mean-pooling over the tokens to obtain a step representation:

$$\phi_l(S) = \frac{1}{|S|} \sum_{s \in S} \left(\frac{1}{|s|} \sum_{i=1}^{|s|} \mathbf{h}_{l,i}^{(s)} \right). \quad (3)$$

Here, $|S|$ denotes the total number of steps in the set, $|s|$ denotes the length of an individual step s, and $\mathbf{h}_{l,i}^{(s)}$ corresponds to the activation of the i-th token within step s at layer l.

Memory Storage. For each historical task Q_j, we organize its extracted representations into a task-specific contrastive pair set $\mathcal{C}_l^{(j)} = \{(\phi_l(x_i^+), \phi_l(x_i^-))\}_{i=1}^{N_j}$, where N_j denotes the total number of contrastive pairs collected for the task and (x_i^+, x_i^-) represents the i-th pair corresponding to the granularities defined in §3.1. These pre-computed sets serve as the raw procedural memory stored in our repository.

3.3 Inference-Time Intervention

NPM applies procedural memory dynamically at inference time without modifying model weights. To ensure low latency, we pre-compute and store the hidden state representations defined in §3.2 for all historical experiences. The inference process follows a Retrieval-Synthesis-Intervention pipeline:

1. **Contextual Retrieval:** Given a new test task q, we utilize a dense retriever to identify the top-K most similar historical tasks $\mathcal{R}_q$.

2. **Dynamic Vector Synthesis:** We fetch the stored representation sets for the retrieved tasks to form a collective memory pool: $\mathcal{M}_l(q) = \bigcup_{j \in \mathcal{R}_q} \mathcal{C}_l^{(j)}$. The synthesis strategy selects between inter- and intra-trajectory contrasts. A task-specific consensus direction $\mathbf{v}_l(q)$ is then derived from the collective memory pool via extraction function $\psi(\cdot)$:

$$\mathbf{v}_l(q) = \psi\big(\mathcal{M}_l(q)\big) \tag{4}$$

3. **Activation Steering:** During autoregressive generation, we inject the synthesized vector into the residual stream at each time step t. For a target layer l, the intervened activation $\tilde{\mathbf{h}}_{l,t}$ is computed as:

$$\tilde{\mathbf{h}}_{l,t} = \mathbf{h}_{l,t} + \alpha \cdot \mathbf{v}_l(q) \tag{5}$$

   where $\mathbf{h}_{l,t}$ denotes the original hidden state of the current token, and α is a scalar parameter controlling the intervention strength.

By shifting the activation distribution, NPM implicitly nudges the agent's intuition away from known failure modes and towards effective reasoning paths. Detailed storage and latency analyses are provided in Appendix D.

4 Experiments

4.1 Experimental Setup

We evaluate the proposed framework using MiniCPM3-4B (Hu et al., 2024) and Qwen3 models (4B and 8B) (Yang et al., 2025) across four agent benchmarks: ALFWorld (Shridhar et al., 2021), WebShop (Yao et al., 2022), ScienceWorld (Wang et al., 2022), and BabyAI (Chevalier-Boisvert et al., 2019). The evaluation relies on the success rate for ALFWorld and the average reward for the remaining environments. Detailed descriptions of each simulated environment and task configurations are deferred to Appendix A.

We compare **NPM** against baselines representing no memory, explicit textual memory, and implicit activation steering. The explicit baselines augment the model context, utilizing **Insights** (Zhao et al., 2024) for global procedural guidelines or **Workflows** (Wang et al., 2025b) for concrete execution patterns. The implicit steering baselines include **CAA** (Panickssery et al., 2024) and **Mass-Mean** (Marks and Tegmark, 2024), applying fixed activation shifts computed across the entire dataset. Our approach differs from these static methods by dynamically synthesizing task-specific intervention vectors from contrastive experiences. We additionally evaluate a **Hybrid** configuration combining implicit steering with explicit workflows to examine whether the two representation modalities offer complementary advantages.

4.2 Main Results

The experimental results are presented in Table 1.

Effectiveness of Implicit Steering. NPM improves performance across most evaluated model configurations and environments compared to the base model without memory. For example, the average score for MiniCPM3-4B increases from 22.60 to 28.87 with implicit steering, and Qwen3-8B improves from 30.63 to 36.32. Furthermore, NPM outperforms static baselines such as CAA and Mass-Mean. These alternative methods struggle to capture multi-step procedural skills since they rely on fixed mean differences computed across the entire dataset. The performance difference indicates that retrieving contextually relevant historical pairs and adapting the intervention to the specific task provides a more accurate behavioral correction.

Competitiveness with Explicit Memory. Implicit activation steering achieves results comparable to explicit textual paradigms. On average, NPM outperforms the Insights approach, demonstrating that direct intrinsic modulation can be more effective than broad textual principles. While the more structured Workflows baseline shows stronger performance by offering explicit step-by-step guidance, NPM remains competitive and outperforms in certain settings. On the WebShop using Qwen3-4B, NPM scores 48.00, exceeding the 45.73 achieved by explicit workflows. While textual workflows provide specific steps, translating these textual instructions into actions can introduce a text-action

Table 1: **The overall performance comparison.** We report the success rate (%) for ALFWorld and the average reward for WebShop, ScienceWorld, and BabyAI. "Avg" denotes the average score across the four benchmarks.

Model	Paradigm	Method	ALFWorld	WebShop	ScienceWorld	BabyAI	Avg
MiniCPM3-4B	None	No Memory	23.88	32.21	7.29	27.00	22.60
	Explicit	Insights	28.36	47.96	10.0	27.78	28.53
		Workflows	32.84	**54.97**	10.5	32.39	32.68
	Implicit	CAA	23.88	13.85	6.29	30.29	18.58
		Mass-Mean	26.87	34.21	8.02	29.73	24.71
		NPM (Ours)	31.34	43.53	8.70	31.91	28.87
	Hybrid	NPM + Workflows	**38.81**	51.21	**10.97**	**36.90**	**34.47**
Qwen3-4B	None	No Memory	30.60	44.81	18.40	18.75	28.14
	Explicit	Insights	44.03	40.72	22.59	12.09	29.86
		Workflows	52.99	45.73	**23.59**	14.60	34.23
	Implicit	CAA	26.87	25.52	11.48	3.48	16.84
		Mass-Mean	29.10	39.07	18.16	17.82	26.04
		NPM (Ours)	40.30	**48.00**	18.20	19.04	31.39
	Hybrid	NPM + Workflows	**61.19**	47.84	21.97	**19.38**	**37.60**
Qwen3-8B	None	No Memory	39.55	46.25	24.98	11.74	30.63
	Explicit	Insights	46.27	45.22	28.09	7.77	31.84
		Workflows	62.69	48.93	29.57	10.40	37.90
	Implicit	CAA	47.01	20.66	9.17	14.31	22.79
		Mass-Mean	49.25	44.26	25.32	8.71	31.89
		NPM (Ours)	56.72	48.24	26.07	14.26	36.32
	Hybrid	NPM + Workflows	**66.42**	**53.94**	**31.89**	**15.31**	**41.89**

disconnect during long sequences (Detailed qualitative case study is provided in Appendix F.2). Furthermore, maintaining extensive textual memory consumes context window space and increases processing overhead. NPM offers an efficient alternative by guiding the model directly in the activation space, delivering competitive performance without occupying context window tokens.

Complementary Synergy of Hybrid Memory. Integrating implicit steering with explicit workflows yields the highest overall performance, setting the top average scores across all three backbone models. Combining both memory forms on Qwen3-8B elevates the success rate on ALFWorld to 66.42 percent and reaches a peak score of 31.89 on ScienceWorld. This synergy indicates that explicit text and implicit representations operate as a complementary system. Textual workflows supply high-level symbolic planning steps, whereas NPM provides direct neural intervention to ensure adherence to these plans during long execution horizons. This persistent modulation helps prevent the agent from deviating from prompt-based instructions during extended interactions.

4.3 Dual-Granularity Contrasts

Tasks across different environments vary in complexity, therefore a single level of intervention may not always provide the appropriate correction. We evaluate the individual effects of intra-trajectory and inter-trajectory steering to understand this behavior. As shown in Figure 3, relying exclusively on a single granularity results in performance variance across different benchmarks and model architectures. Intra-trajectory steering performs well in step-intensive environments such as ALFWorld by correcting local action errors. Inter-trajectory steering proves more effective when agents require macro-level planning guidance like WebShop. To balance these effects, we dynamically select the appropriate intervention granularity based on the task context. This combination reduces performance variance and provides stable generalization across configurations. Although the implemented selection strategy occasionally yields lower success rates than an oracle baseline in certain boundary

cases, the empirical results confirm that the dual-granularity contrast provides a reliable mechanism for managing procedural interventions. The representational differences underlying these two steering granularities are further examined in Section 5.

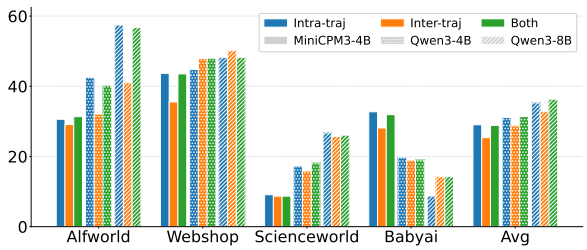


Figure 3: Performance comparison of different steering granularities across multiple environments and models.

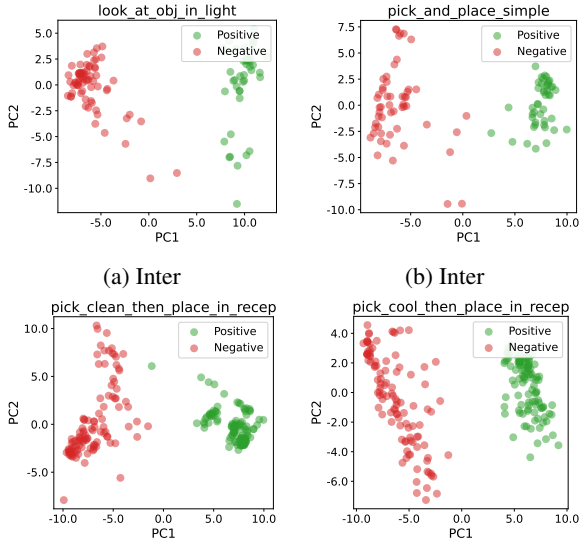
5 Analysis

5.1 Representational Separability of Activations

Effective activation steering relies on the assumption that successful patterns and failure modes occupy distinct regions in the representation space. Projecting the hidden states from the intervention layer onto their primary axes reveals a clear geometric separation. As illustrated in Figure 4, the projections for both inter-trajectory and intra-trajectory contrasts show that successful paths and degenerate modes form distinct clusters instead of overlapping. We also confirm this linear separability in the original high-dimensional space by training a linear classifier, which achieves high accuracy in distinguishing between the two classes (detailed in Appendix C.1). This geometric separation suggests that the behavioral shift from failure to success can be projected onto a prominent direction. Extracting the first principal component of the contrastive differences helps capture this main axis of variation, mitigating some instance-specific noise.

5.2 Consistency of Steering Vectors

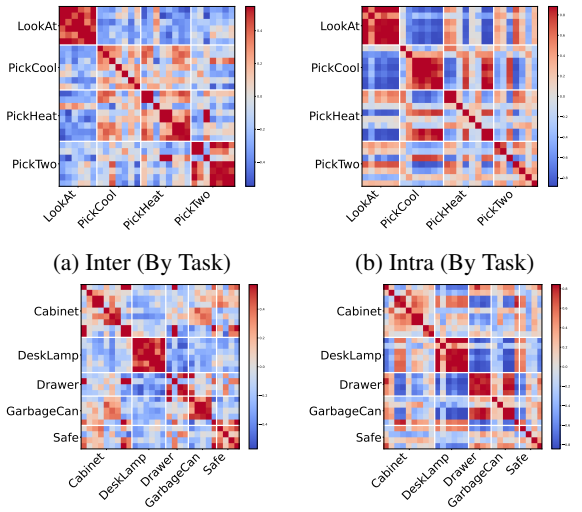
We evaluate the structural properties of the steering vectors by computing their pairwise cosine similarities across different task instances. The heatmaps in Figures 5a and 5b demonstrate that these vectors maintain high internal consistency within specific task categories. Tasks such as *LookAt* involve distinct interaction logic and show low similarity to the *PickCool*, *PickHeat*, and *PickTwo* groups. These three categories display inter-task similarity alongside their high intra-task consistency because



(a) Inter (b) Inter

(c) Intra (d) Intra

Figure 4: PCA projections of hidden states at Layer 18 for Qwen3-4B on AlfWorld. Positive (green) and negative (red) representations exhibit distinct clustering along the primary axis of variation.



(a) Inter (By Task) (b) Intra (By Task)

(c) Inter (By Interaction) (d) Intra (By Interaction)

Figure 5: Geometric consistency of procedural steering vectors. Heatmaps display the pairwise cosine similarity of centered vectors. (a-b) Vectors clustered by task type; (c-d) Vectors clustered by interaction target. The diagonal block structure suggests that the extracted vectors align with task-specific categories and share related behavioral features.

they share the fundamental *Pick* sub-action. This overlapping pattern indicates the framework captures common procedural knowledge across related behaviors. The resulting diagonal block structure suggests that the memory extraction process yields representations that align with task-specific cate-

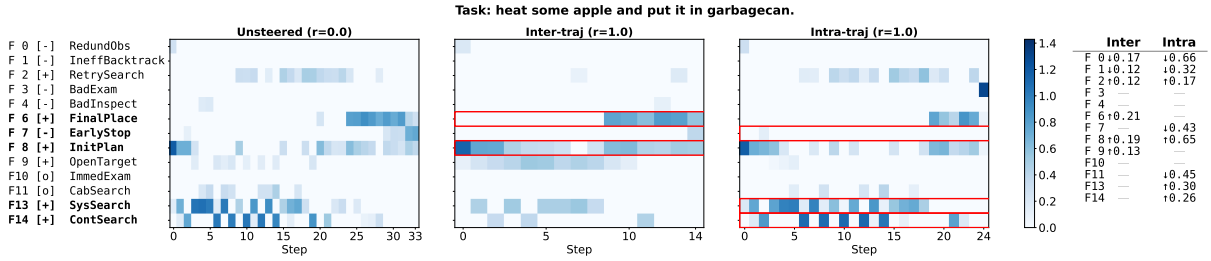


Figure 6: **Temporal activation of behavioral primitives across execution steps in a PickHeat task. (Left)** The unsteered baseline exhibits an extended 33-step trajectory. **(Middle)** *Inter-trajectory* steering is associated with a shorter trajectory by promoting early planning and object placement. **(Right)** *Intra-trajectory* steering is observed to amplify search features and suppress premature task termination signals during execution.

gories rather than random noise.

Grouping the vectors by specific interaction targets in Figures 5c and 5d highlights functional differences between the two extraction strategies. The intra-trajectory contrasts form diagonal block structures with high similarity. Local error-correction signals associated with particular items like a *DeskLamp* or *Drawer* remain uniform across different task instances. Correcting localized degenerate steps relies primarily on the intrinsic properties of the target object rather than the broader task context. The inter-trajectory contrasts show a less structured pattern with lower consistency. Global reasoning paths depend on the initial state of the agent, environmental layouts, and prior execution sequences. This contextual variance prevents inter-trajectory vectors from forming the distinct clusters observed in the local signals.

5.3 Feature Decomposition and Interpretability of Steering Vectors

To understand the behavioral semantics encoded within the steering vectors, we decompose the continuous representations into interpretable basis directions using sparse dictionary learning (detailed in Appendix B). Each direction corresponds to specific behavioral primitives, which we validate by computing the mutual information between basis activations and the generated action types across the dataset. Projecting the steering vectors onto these bases reveals that the two contrastive strategies operate through distinct mechanistic pathways.

The *PickHeat* task in Figure 6 illustrates how the two extraction strategies modulate representations through distinct mechanisms. Without intervention, the baseline model generates an extended 33-step trajectory. Applying inter-trajectory steering leads to global behavioral changes. This vector increases early planning features (F8) and sustains

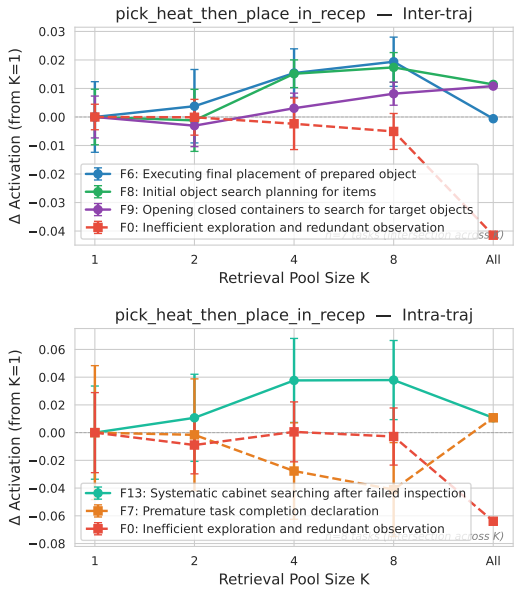


Figure 7: The impact of retrieval pool size on specific behavioral primitive activations (Solid lines for positive features; dashed lines for negative features).

activations related to concrete object placement (F6), shortening the execution to 14 steps. Intra-trajectory steering targets local execution errors instead of broad execution plans. It corrects action loops by amplifying features associated with targeted searching (F13) and container interaction (F14) while suppressing signals that trigger premature task termination (F7). This difference in activation patterns suggests that the steering vectors target distinct aspects of model behavior. They systematically adjust representations to address different types of procedural errors (see more quantitative analysis in Appendix C.3 and Appendix C.4).

5.4 Retrieval Size and Dynamic Synthesis

To investigate how the retrieval size affects the extracted procedural signals, we track the activation

of behavioral primitives as the number of aggregated contrastive pairs increases. As shown in Figure 7, features display two distinct scaling patterns depending on their universality. For general behaviors, such as basic exploration (F9) and redundant observation (F0), activation magnitudes grow continuously as the size expands. Since these patterns are shared across most tasks, they remain prominent when aggregated across the entire dataset. Conversely, task-specific primitives exhibit a different pattern. Features tied to explicit procedural logic, such as placing objects (F6), systematic searching (F13), or preventing premature completion (F7), reach peak activation at a moderate retrieval scale before declining. While minimal pools fail to separate the core procedural signal from instance-level noise, overly large pools introduce cross-task interference that suppresses specific task rules. This observation confirms the necessity of dynamic synthesis: constraining the retrieval to a contextually relevant subset effectively balances noise reduction with signal preservation.

6 Conclusion

We propose Neural Procedural Memory (NPM), a training-free framework that represents and applies procedural behaviors through implicit activation steering. By distilling procedural knowledge from dual-granularity contrastive experiences, NPM provides an alternative paradigm for encoding memory that operates directly in the activation space. Extensive experiments show that NPM achieves performance comparable to textual baselines and provides complementary benefits when combined with explicit workflows. By utilizing the continuous activation space to encode procedural skills, NPM provides a practical foundation for managing procedural memory in LLM agents.

Limitations

While our proposed approach demonstrates promising results, there are several aspects that could be improved. Firstly, the framework requires direct intervention in the residual stream, restricting its application to open architectures where internal activation spaces are completely accessible. Furthermore, building the contrastive repository relies on the agent occasionally generating successful trajectories, which introduces a cold-start problem in complex environments. Additionally, extracting degenerate steps relies on heuristics such as action redundancy or format invalidity. These conditions struggle to identify implicit logical fallacies without explicit environment errors. Finally, synthesized vectors compress procedural knowledge into a single representation applied constantly during generation. This static modulation lacks the flexibility to shift between behavioral primitives across different execution stages. Exploring dynamic interventions that adapt throughout task execution is a direction for future work.

Acknowledgments

This work was supported by the Beijing Major Science and Technology Project (No. Z251100008125025), the National Natural Science Foundation of China (No.62376270) and the independent research project of the Key Laboratory of Cognition and Decision Intelligence for Complex Systems. This work was supported by Beijing Academy of Artificial Intelligence (BAAI).

References

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. 2023. Language models can explain neurons in language models. https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3829–3846, Singapore. Association for Computational Linguistics.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2019. BabyAI: First steps towards grounded language learning with a human in the loop. In *International Conference on Learning Representations*.

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

Yilin Geng, Haonan Li, Honglin Mu, Xudong Han, Timothy Baldwin, Omri Abend, Eduard Hovy, and Lea Frermann. 2025. Control illusion: The failure of instruction hierarchies in large language models. *Preprint*, arXiv:2502.15851.

Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. HippoRAG: Neurobiologically inspired long-term memory for large

language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Dongge Han, Camille Couturier, Daniel Madrigal Diaz, Xuchao Zhang, Victor Rühle, and Saravan Rajmohan. 2025. Legomem: Modular procedural memory for multi-agent llm systems for workflow automation. *Preprint*, arXiv:2510.04851.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, and 1 others. 2024. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, Senjie Jin, Jiejun Tan, Yanbin Yin, Jiongnan Liu, Zeyu Zhang, Zhongxiang Sun, Yutao Zhu, Hao Sun, Boci Peng, and 28 others. 2025. Memory in the age of ai agents. *Preprint*, arXiv:2512.13564.

Bruce W. Lee, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Erik Miehling, Pierre Dognin, Manish Nagireddy, and Amit Dhurandhar. 2025. Programming refusal with conditional activation steering. In *The Thirteenth International Conference on Learning Representations*.

Bang Liu, Xinfeng Li, Jiayi Zhang, Jinlin Wang, Tanjin He, Sirui Hong, Hongzhang Liu, Shaokun Zhang, Kaitao Song, Kunlun Zhu, Yuheng Cheng, Suyuchen Wang, Xiaoqiang Wang, Yuyu Luo, Haibo Jin, Peiyan Zhang, Ollie Liu, Jiaqi Chen, Huan Zhang, and 29 others. 2025. Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems. *Preprint*, arXiv:2504.01990.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2025. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 33:3776–3786.

Kyle Mahowald, Anna A Ivanova, Idan A Blank, Nancy Kanwisher, Joshua B Tenenbaum, and Evelina Fedorenko. 2024. Dissociating language and thought in large language models. *Trends in cognitive sciences*, 28(6):517–540.

Samuel Marks and Max Tegmark. 2024. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. In *First Conference on Language Modeling*.

Parsa Omidi, Xingshuai Huang, Axel Laborieux, Bahareh Nikpour, Tianyu Shi, and Armaghan Eshaghi. 2025. Memory-augmented transformers: A systematic review from neuroscience principles to enhanced model architectures. *Preprint*, arXiv:2508.10824.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. Memgpt: Towards llms as operating systems. *Preprint*, arXiv:2310.08560.

Nina Panickssery, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. 2024. Steering llama 2 via contrastive activation addition. *Preprint*, arXiv:2312.06681.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, New York, NY, USA. Association for Computing Machinery.

Gonçalo Santos Paulo, Alex Troy Mallen, Caden Juang, and Nora Belrose. 2025. Automatically interpreting millions of features in large language models. In *Forty-second International Conference on Machine Learning*.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. 2024. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522, Bangkok, Thailand. Association for Computational Linguistics.

Daniel Scalena, Gabriele Sarti, and Malvina Nissim. 2024. Multi-property steering of large language models with dynamic activation composition. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 577–603, Miami, Florida, US. Association for Computational Linguistics.

Yunfan Shao, Linyang Li, Junqi Dai, and Xipeng Qiu. 2023. Character-LLM: A trainable agent for role-playing. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13153–13187, Singapore. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Akshit Sinha, Arvindh Arun, Shashwat Goel, Steffen Staab, and Jonas Geiping. 2025. The illusion of diminishing returns: Measuring long horizon execution in llms. *Preprint*, arXiv:2509.09677.

Larry R. Squire. 1992. Declarative and nondeclarative memory: Multiple brain systems supporting learning and memory. *Journal of Cognitive Neuroscience*, 4(3):232–243.

Larry R. Squire. 2004. Memory systems of the brain: A brief history and current perspective. *Neurobiology of Learning and Memory*, 82(3):171–177. Multiple Memory Systems.

Wangtao Sun, Chenxiang Zhang, Xueyou Zhang, Xuanqing Yu, Ziyang Huang, Haotian Xu, Shizhu He, Jun Zhao, and Kang Liu. 2025. Beyond instruction following: Evaluating inferential rule following of large language models. In *Chinese Computational Linguistics: 24th China National Conference, CCL 2025, Jinan, China, August 11–14, 2025, Proceedings*, page 408–434, Berlin, Heidelberg. Springer-Verlag.

Yuqiao Tan, Jinxiang Meng, Fangyu Lei, Minzheng Wang, Shizhu He, Jun Zhao, and Kang Liu. 2026a. Swe-touch: Benchmarking coding agents when users touch the code. *Preprint*, arXiv:2608.02499.

Yuqiao Tan, Minzheng Wang, Shizhu He, Huanxuan Liao, Chengfeng Zhao, Qiunan Lu, Tian Liang, Jun Zhao, and Kang Liu. 2026b. Bottom-up policy optimization: Your language model policy secretly contains internal policies. *Preprint*, arXiv:2512.19673.

Yuqiao Tan, Minzheng Wang, Bo Liu, Zichen Liu, Tian Liang, Shizhu He, Jun Zhao, and Kang Liu. 2026c. From $p(y|x)$ to $p(y)$: Investigating reinforcement learning in pre-train space. *Preprint*, arXiv:2604.14142.

BAAI RoboBrain Team, Mingyu Cao, Huajie Tan, Yuheng Ji, Xiansheng Chen, Minglan Lin, Zhiyu Li, Zhou Cao, Pengwei Wang, Enshen Zhou, Yi Han, Yingbo Tang, Xiangqi Xu, Wei Guo, Yaoxu Lyu, Yijie Xu, Jiayu Shi, Mengfei Du, Cheng Chi, and 34 others. 2025. Robobrain 2.0 technical report. *Preprint*, arXiv:2507.02029.

Constantin Venhoff, Iván Arcuschin, Philip Torr, Arthur Conmy, and Neel Nanda. 2025. Base models know how to reason, thinking models learn when. *Preprint*, arXiv:2510.07364.

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

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024a. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. Scienceworld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11279–11298.

Tiannan Wang, Meiling Tao, Ruoyu Fang, Huilin Wang, Shuai Wang, Yuchen Eleanor Jiang, and Wangchunshu Zhou. 2024b. Ai persona: Towards life-long personalization of llms. *Preprint*, arXiv:2412.13103.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, Jingbo Shang, and Julian McAuley. 2024c. Memoryllm: towards self-updatable large language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Yu Wang, Xinshuang Liu, Xiusi Chen, Sean O'Brien, Junda Wu, and Julian McAuley. 2025a. Self-updatable large language models by integrating context into model parameters. In *The Thirteenth International Conference on Learning Representations*.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2025b. Agent workflow memory. In *Forty-second International Conference on Machine Learning*.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxin Xu, Yiming Liu, Jie Tang, Hongning Wang, and Minlie Huang. 2024. Benchmarking complex instruction-following with multiple constraints composition. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

Yuchen Xia, Jiho Kim, Yuhan Chen, Haojie Ye, Souvik Kundu, Cong Callie Hao, and Nishil Talati. 2024. Understanding the performance and estimating the cost of llm fine-tuning. In *2024 IEEE International Symposium on Workload Characterization (IISWC)*, pages 210–223.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, Ran Xu, Phil Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. 2024. Retroformer: Retrospective large language agents with policy gradient optimization. *Preprint*, arXiv:2308.02151.

Yiqun Yao, Xiang Li, Xin Jiang, Xuezhi Fang, Naitong Yu, Wenjia Ma, Aixin Sun, and Yequan Wang. 2026. Flm-audio: Natural monologues improves native full-duplex chatbots via dual training. *Preprint*, arXiv:2509.02521.

Tao Yu, Zhengbo Zhang, Zhiheng Lyu, Junhao Gong, Hongzhu Yi, Xinming Wang, Yuxuan Zhou, Jiabing Yang, Ping Nie, Yan Huang, and Wenhu Chen. 2025a. Browseragent: Building web agents with human-inspired web browsing actions. *Preprint*, arXiv:2510.10666.

Xinlei Yu, Chengming Xu, Guibin Zhang, Zhangquan Chen, Yudong Zhang, Yongbo He, Peng-Tao Jiang, Jiangning Zhang, Xiaobin Hu, and Shuicheng Yan. 2025b. Vismem: Latent vision memory unlocks potential of vision-language models. *Preprint*, arXiv:2511.11007.

Yuexiang Zhai, Shengbang Tong, Xiao Li, Mu Cai, Qing Qu, Yong Jae Lee, and Y. Ma. 2023. Investigating the catastrophic forgetting in multimodal large language models. *ArXiv*, abs/2309.10313.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. 2025a. Memgen: Weaving generative latent memory for self-evolving agents. *Preprint*, arXiv:2509.24704.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, Jian Xie, Yuxuan Sun, Boyu Gou, Qi Qi, Zihang Meng, Jianwei Yang, Ning Zhang, Xian Li, Ashish Shah, and 11 others. 2025b. Agent learning via early experience. *Preprint*, arXiv:2510.08558.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2025c. A survey on the memory mechanism of large language model-based agents. *ACM Trans. Inf. Syst.*, 43(6).

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. 2023. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *The Twelfth International Conference on Learning Representations*.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19724–19731.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, and 2 others. 2025. Representation engineering: A top-down approach to ai transparency. *Preprint*, arXiv:2310.01405.

A Implementation Details

A.1 Dataset Details

Our evaluation utilizes four simulated environments that test diverse procedural skills and long-horizon planning capabilities.

ALFWorld (Shridhar et al., 2021) is a text-based embodied environment where agents navigate simulated rooms and interact with household objects to complete daily tasks. WebShop (Yao et al., 2022) provides a simulated e-commerce website where agents must execute search queries, navigate product pages, and select specific item attributes to purchase products matching detailed user instructions. ScienceWorld (Wang et al., 2022) simulates an elementary school science laboratory, requiring agents to follow strict procedural logic to conduct experiments. BabyAI (Chevalier-Boisvert et al., 2019) is a partially observable gridworld environment where agents execute sequential movement and manipulation commands to complete specified spatial tasks.

A.2 Data Construction and Model Configuration

We construct the contrastive dataset by sampling tasks directly from the training splits of ALFWorld, WebShop, ScienceWorld, and BabyAI. For each training task, we generate multiple trajectories using the target model configured with specific sampling parameters.

To isolate degenerate behaviors for intra-trajectory contrasts, we apply deterministic rules based on action redundancy and invalidity. Redundancy is identified through pattern matching that detects repetitive loops or consecutive identical action sequences. Categorizing segments where the agent enters a state unable to progress normally as degenerate steps provides a reasonable heuristic for capturing procedural failures. Invalidity is determined by matching environment-specific error messages and format violations, which filters irrational actions from the effective trajectory pool.

All experiments employ `sentence-transformers/all-mpnet-base-v2` for dense retrieval to retrieve a fixed number of the most relevant historical tasks. The agent then performs greedy decoding to ensure the reproducibility of the evaluation results. The detailed sampling parameters and retrieval configurations for all evaluated environments are summarized in Table 2.

Table 2: Data construction parameters. All parameters except the maximum interaction steps are kept consistent across the four evaluated benchmarks.

Parameter	Value
Sampled trajectories (N)	16
Temperature	0.7
Top-p	0.8
Top-k	20
Max turns (ALFWorld)	50
Max turns (WebShop)	15
Max turns (ScienceWorld)	30
Max turns (BabyAI)	20
Retrieval count (K)	8

A.3 Dynamic Synthesis and Vector Extraction

To determine the intervention granularity g, we estimate task complexity using the average execution length of the retrieved successful trajectories, denoted as L_q. We compare this length against the global average trajectory length of the respective benchmark, denoted as $\bar{L}$. Tasks with extended horizons typically encounter localized procedural failures such as repetitive loops, making the concentrated error-correction of intra-trajectory steering more appropriate. Conversely, shorter tasks typically fail due to misaligned global planning, which benefits from the broader behavioral alignment provided by inter-trajectory steering. The system selects the appropriate granularity based on a simple thresholding strategy, with the scaling factor γ set to 1.5 in our experiments:

$$g = \begin{cases} \text{intra-trajectory}, & \text{if } L_q > \gamma \bar{L} \\ \text{inter-trajectory}, & \text{otherwise} \end{cases} \tag{6}$$

Nevertheless, identifying the optimal granularity for every instance remains a non-trivial challenge. Our heuristic selection occasionally scores lower than the oracle single-granularity baseline in specific settings. Future research could explore more effective methods for allocating steering vectors.

After selecting the appropriate contrastive pool, we extract the steering vectors using a standard principal component analysis pipeline. The paired positive and negative activations are first mean-centered to eliminate shared task-specific biases. We then compute the principal components of these normalized representations and extract the dominant component as the final continuous steering vector.

A.4 Steering Configuration

We apply activation steering to the middle-to-late layers of the language models. Specifically, we inject the steering vectors into layers 17 to 19 for Qwen3-4B and Qwen3-8B, and layers 46 to 48 for MiniCPM3-4B.

To achieve dynamic steering across these targeted layers, we adapt a simplified version of the calibration mechanism based on Kullback-Leibler divergence proposed by Scalena et al. (2024). This method tracks the probability shift caused by the steering vector and reduces the intervention strength if the modified next-token distribution $P_\alpha(\cdot \mid x_{<t})$ deviates from the original unsteered distribution $P(\cdot \mid x_{<t})$ by a large margin. We define a discrete set of candidate scales $\mathcal{A}$ and select the optimal strength α^* by identifying the maximum value that keeps the divergence below a target threshold ϵ:

$$\alpha^* = \max \Big\{ \alpha \in \mathcal{A} \;\Big|\; D_{\text{KL}}\big(P(\cdot \mid x_{<t}) \parallel P_\alpha(\cdot \mid x_{<t})\big) \leq \epsilon \Big\} \tag{7}$$

B Steering Vector Interpretability and Feature Analysis Details

B.1 Feature Extraction and Semantic Annotation

To investigate the underlying mechanisms of implicit activation steering, we decompose the continuous representation space into interpretable basis directions following the methodology of Venhoff et al. (2025). We collect step-level hidden states from historical interaction trajectories at layer 18 for the Qwen3-4B model and apply a sparse dictionary learning algorithm with a top-k activation constraint. The dictionary size is constrained to 16 dimensions with a sparsity threshold of 3. This dimensional configuration aligns with prior research on isolating specific high-level procedural behaviors. The empirical observation of inactive dimensions within this learned dictionary indicates that a 16-dimensional space provides adequate capacity for capturing the procedural variations specific to agentic task-solving steps.

We assign interpretable semantics to these learned basis directions by analyzing their activation patterns across the dataset. We quantify behavioral selectivity by calculating the mutual information between the activation state of each basis

direction f and the discrete action type a generated by the agent. This statistical measurement is defined as:

$$MI(F;A) = \sum_a \Bigg[P(f,a) \log \frac{P(f,a)}{P(f)P(a)} + P(\neg f, a) \log \frac{P(\neg f, a)}{P(\neg f)P(a)} \Bigg] \tag{8}$$

Building on established automated interpretability techniques (Bills et al., 2023; Paulo et al., 2025), we utilize DeepSeek-V3.2 to analyze these statistical distributions alongside top-activating examples. The model processes this information according to the prompt template detailed in Figure 8 to generate descriptive textual labels and determine the behavioral polarity for each active basis direction. The resulting semantic annotations for the ALFWorld benchmark are summarized in Table 3.

B.2 Geometric Projection of Steering Vectors

We analyze the functional composition of the extracted principal component steering vectors by projecting them onto the annotated basis directions. The geometric projection is computed as the dot product between the normalized steering vector $\mathbf{v}$ and each unit-norm basis vector $\mathbf{w}_i$ derived from the dictionary learning phase:

$$p_i = \mathbf{w}_i^\top \frac{\mathbf{v}}{\|\mathbf{v}\|} \tag{9}$$

A positive projection value $p_i > 0$ indicates that the steering intervention amplifies the corresponding behavioral primitive. A negative value $p_i < 0$ signifies suppression of that specific behavior. This mathematical decomposition confirms that the steering vectors function as composite modifiers that simultaneously inhibit pathological patterns and stimulate necessary subsequent steps to correct agent reasoning.

C Quantitative Validation of Methodology

C.1 Linear Separability in High-Dimensional Manifold

We train a linear support vector machine on the hidden states extracted from the target intervention layers of the retrieved trajectories to verify the linear separability of successful and failed procedural logic. This evaluation determines whether effective and degenerate reasoning modes occupy distinct regions within the representation space (Marks and Tegmark, 2024; Zou et al., 2025). We perform 5-fold stratified cross-validation on the Qwen3-4B model across all four evaluated benchmarks. The classification accuracies at both the inter-trajectory and intra-trajectory levels are reported in Table 4. The consistently high classification accuracy across different environments confirms that a linear hyperplane effectively divides the two reasoning classes. This mathematical separability provides the theoretical justification for applying principal component analysis to extract a dominant directional vector for behavioral modulation.

C.2 Accuracy of Heuristic Rules for Degenerate Steps

The efficacy of intra-trajectory steering depends on isolating degenerate steps without contaminating the negative representation pool. We evaluate the reliability of our deterministic isolation rules based on consecutive loop detection and invalid format matching by analyzing sampled steps from 1,000 historical trajectories per benchmark. An LLM-based evaluator assesses each step against the criteria of action redundancy and environment constraints to compute the false positive and false negative rates. The evaluation metrics presented in Table 5 show a low overall false positive rate of 0.61% and a low overall false negative rate of 6.34%, indicating that the heuristics reliably isolate genuine procedural failures and maintain high purity in the negative representations. The minor variance in error rates across benchmarks, such as the slightly higher false negative rate of 9.30% in WebShop, occurs because rigid deterministic rules can occasionally miss subtle, domain-specific logical errors. These undetected failures subsequently mix into the effective step pool. Although this introduces noise into the positive representations, the empirical results confirm that the overall contrastive signal remains robust for behavioral modulation. Future work could address this limitation by replacing heuristic rules with more capable evaluation mechanisms, such as utilizing a language model as a judge to identify nuanced reasoning failures during data construction.

C.3 Distribution Analysis of Inter- and Intra-Trajectory Feature Activations

To evaluate how steering vectors modulate behaviors, we analyze their geometric distribution across the basis directions defined in Equation 9 using nor-

Table 3: Automated annotations and activation statistics of the learned basis directions in the ALFWorld environment.

Index	Semantic Label	Overall Active	Active on Success	Active on Failure	Polarity
0	**RedundObs**: Inefficient exploration and redundant observation	42.0%	7.3%	46.2%	Negative
1	**IneffBacktrack**: Ineffective backtracking after failed object placement	10.0%	0.5%	11.1%	Negative
2	**RetrySearch**: Searching for objects after failed container checks	17.9%	24.3%	17.1%	Positive
3	**BadExam**: Misguided final-step execution with examination tasks	1.5%	0.8%	1.6%	Negative
4	**BadInspect**: Misguided visual inspection of desk objects	23.5%	7.7%	25.4%	Negative
5	**Dead**: Dead feature	0.0%	0.0%	0.0%	N/A
6	**FinalPlace**: Executing final placement of prepared object	23.1%	33.0%	21.9%	Positive
7	**EarlyStop**: Premature task completion declaration	36.5%	8.0%	40.0%	Negative
8	**InitPlan**: Initial object search planning for food items	32.6%	67.1%	28.4%	Positive
9	**OpenTarget**: Opening closed containers to search for target objects	20.9%	22.7%	20.7%	Positive
10	**ImmedExam**: Immediate visual examination of target objects	6.4%	6.3%	6.5%	Neutral
11	**CabSearch**: Opening closed cabinets to search for objects	22.1%	24.7%	21.8%	Neutral
12	**Dead**: Dead feature	0.0%	0.0%	0.0%	N/A
13	**SysSearch**: Systematic cabinet searching after empty inspection	41.2%	60.0%	38.9%	Positive
14	**ContSearch**: Opening closed containers to search for objects	22.3%	37.6%	20.4%	Positive
15	**Dead**: Dead feature	0.0%	0.0%	0.0%	N/A

Table 4: Linear SVM classification accuracy (%) on the native high-dimensional hidden states of Qwen3-4B across 5-fold stratified cross-validation.

Benchmark	Inter-traj.	Intra-traj.
ALFWorld	99.55 ($\pm$0.51)	99.99 ($\pm$0.01)
WebShop	88.46 ($\pm$0.55)	99.91 ($\pm$0.10)
ScienceWorld	93.41 ($\pm$0.69)	99.81 ($\pm$0.12)
BabyAI	99.53 ($\pm$0.23)	99.61 ($\pm$0.10)

Table 5: Evaluation of heuristic rule accuracy based on step-level annotation across four benchmarks. False positive rate (FPR) and false negative rate (FNR) are evaluated by an LLM-as-a-judge on sampled steps from 1,000 trajectories per environment.

Benchmark	**FPR**	**FNR**
ALFWorld	0.00% (0/2,862)	3.57% (106/2,967)
WebShop	0.64% (16/2,490)	9.30% (268/2,883)
ScienceWorld	0.04% (1/2,833)	6.97% (184/2,640)
BabyAI	1.75% (51/2,908)	5.73% (171/2,983)
Overall	0.61% (68/11,093)	6.34% (729/11,473)

malized entropy and top-three concentration. For any task, the projection proportion along each basis direction is:

$$\tilde{p}_i = \frac{|p_i|}{\sum_{j=1}^{N} |p_j|} \tag{10}$$

The global uniformity of this distribution is captured by the normalized entropy:

$$H_{\text{norm}} = -\frac{\sum_{i=1}^{N} \tilde{p}_i \log \tilde{p}_i}{\log N} \tag{11}$$

where N denotes the total number of active directions. To measure local focus, we also compute the top-three concentration, representing the cumulative proportion of the three largest absolute projection values:

$$C_{\text{top3}} = \frac{\sum_{i \in \Omega} |p_i|}{\sum_{j=1}^{N} |p_j|} \tag{12}$$

with Ω denoting the indices of these three dominant directions.

Empirical analysis of these metrics reveals distinct structural differences between the two steering granularities. As shown in Figure 9, inter-trajectory steering vectors exhibit a mean normalized entropy of 0.919, whereas intra-trajectory vectors yield a lower mean of 0.885. This difference indicates that macro-level planning steering operates via a more distributed modulation across multiple basis directions. Conversely, Figure 10 shows that the mean top-three concentration is 38.0% for inter-trajectory steering and 43.6% for intra-trajectory steering. The higher concentration of intra-trajectory vectors indicates a more targeted activation pattern. Instead of adjusting the overall reasoning flow, intra-trajectory intervention selectively modulates specific behavioral dimensions to resolve localized failures. These patterns suggest that global planning guidance tends to rely on distributed representations, whereas local error recovery is associated with targeted corrections.

C.4 Causal Intervention on Behavioral Features

To examine whether the interpretable features identified in Section 5.3 causally modulate model behavior, we conduct bidirectional interventions on two failure-related directions: F7 (`EarlyStop`), representing premature task completion, and F0 (`RedundObs`), representing redundant exploration. We directly amplify or suppress each direction in the residual stream and evaluate the resulting task success rate.

As shown in Table 6, suppressing either failure-related feature improves performance, whereas amplifying it consistently reduces performance. Jointly suppressing F0 and F7 achieves the highest success rate of 46.27%, an improvement of 11.20 percentage points over the baseline. The predictable bidirectional effects and the gain from joint intervention provide direct causal evidence that the extracted directions encode specific and compositionally controllable procedural behaviors.

Table 6: Causal interventions on individual behavioral features. Δ denotes the absolute percentage-point change from the baseline.

Intervention	Success Rate	Δ
Baseline (no steering)	35.07%	—
Suppress F7 (`EarlyStop`)	38.06%	+2.99
Amplify F7 (`EarlyStop`)	27.61%	-7.46
Suppress F0 (`RedundObs`)	42.54%	+7.47
Amplify F0 (`RedundObs`)	32.84%	-2.23
Suppress F0 + F7	**46.27%**	**+11.20**

D System Efficiency and Overhead

D.1 Storage and Computation Overhead

The framework introduces minimal storage overhead compared to text-based retrieval. For each historical trajectory, the system extracts three representations at each targeted layer: one last-token state for inter-trajectory contrast and two mean-pooled states representing effective and degenerate steps for intra-trajectory contrast. Given a hidden dimension d, floating-point byte size b, and L target layers, the total storage requirement per trajectory is $3 \times L \times d \times b$. For Qwen3-4B with $d = 2560$ in half-precision format ($b = 2$) targeting three layers, the footprint is 45 kilobytes. This fixed constraint ensures that storage scales independently of the original interaction length. Unlike traditional retrieval systems that consume substantial video memory by expanding the key-value cache with extensive textual contexts, activation steering modifies the residual stream directly to bypass this limitation. The offline computation for memory construction is limited to a single forward pass to extract representations. These continuous vectors can be efficiently managed in a standard database without requiring model parameter updates.

D.2 Latency Analysis

We evaluate the inference latency of the proposed framework across retrieval, scale selection probing, vector synthesis, prompt prefill, and autoregressive decoding. The timing results in Table 7 indicate that textual memory incurs a substantial prefill overhead, increasing latency from 63.46 ms to 279.89 ms, as the self-attention mechanism must process extended textual contexts. In contrast, NPM maintains a prefill latency of 71.09 ms, nearly identical to the No Memory baseline. Although NPM introduces additional phases to compute steering vectors and steering strength, this process imposes no extra latency penalty relative to textual memory.

During autoregressive decoding, vector injection involves only an element-wise addition within the residual stream. This basic operation adds no measurable delay to token generation. These efficiency characteristics establish implicit activation steering as a practical and scalable paradigm for persistent agentic memory.

Table 7: Inference latency breakdown (ms).

Method	Retrieval	Synthesis	Probe	Prefill
No Memory	—	—	—	63.46
Textual Memory	13.83	—	—	279.89
NPM (Ours)	13.83	77.42	130.46	71.09

E Generalization on Suppressing Degenerate Actions

To evaluate the applicability of NPM across different models, we conduct additional experiments on RoboBrain (Yao et al., 2026; Team et al., 2025). We first identify 30 tasks in which RoboBrain exhibits degenerate actions without intervention. Among these tasks, 11 are used to select the intervention configuration, while the remaining 19 constitute a held-out test set. For each task, we construct intra-trajectory contrasts using its top-8 retrieved trajectories generated by Qwen3-4B. The corresponding activations are then computed with RoboBrain and used to extract the steering vector.

For each held-out task, we branch from exactly the same state at which the baseline trajectory first triggers a repetitive loop. With NPM intervention, RoboBrain produces a valid action and escapes the original degenerate actions in 57.9% of the held-out tasks. At the full-trajectory level, the percentage of tasks containing degenerate actions decreases from 100.0% to 43.1%, while the proportion of degenerate actions among all generated actions decreases from 70.0% to 39.7%, as shown in Table 8. These results demonstrate that the procedural directions extracted by NPM generalize to a new model and selectively suppress the encoded repetitive failure patterns.

F Examples and Case Studies

F.1 Examples of Textual Procedural Memory

Examples are provided in Figure 11, 12.

Table 8: Suppression of degenerate actions.

Metric	No Intervention	NPM
Tasks with degenerate actions (%)	100.0	43.1
Degenerate-action proportion (%)	70.0	39.7

F.2 Case Study: Alleviating Text-Action Disconnect in Action Execution

We examine agent behavior in a multi-object retrieval task requiring the placement of two CDs into a safe (Table 9). The baseline model receives an explicit textual workflow outlining the necessary sequential operations and repetition conditions. When the environment rejects an attempt to pick up a second object due to inventory constraints, the text-augmented agent fails to update its internal state representation. The model generates subsequent commands based on an incorrect assumption of successful object acquisition and enters an invalid action loop of attempting to place an unpossessed item into the receptacle. This behavior demonstrates the text-action disconnect where semantic comprehension of discrete textual rules does not ensure sustained state maintenance during execution. The agent equipped with neural procedural memory completes the entire fetch-and-place cycle without textual prompting. Intervening directly in the residual stream allows the synthesized steering vectors to suppress neural activations associated with redundant loops while amplifying features related to sequential planning as identified in prior mechanistic analyses. Modulating the continuous activation space directly enforces the procedural logic required for multi-step execution and prevents the trajectory from degrading into the repetitive errors observed when relying solely on external textual memory.

Table 9: Qualitative comparison of procedural memory in the “find two cd and put them in safe" task. Despite the **Textual Workflow** providing explicit instructions, the agent fails to manage the multi-step trip after an initial action failure. **NPM** steers the agent implicitly adhering to the sequential requirements.

Task	**Your task is to: find two cd and put them in safe.**	
Memory	**Textual Workflow:** *“[Workflow: search_and_place_multiple_objects] ... **take {object 1}** from {location} ... **go to {target_receptacle}** ...”*	**NPM (Ours):** *Implicit steering vector distilled from historical multi-object retrieval tasks.*
Discovery	Both agents successfully locate two target CDs in the environment and attempt to pick them up.	
Constraint Hit	**Action:** `take cd 1` → `go to desk 2` → `take cd 2` **Obs:** (Action rejected as hands are full). **Thought:** “I have taken cd 2 from desk 2. Now I need to go to the safe to place both cds.” *(Erroneous state tracking)*	**Action:** `take cd 3` → `go to desk 1` → `take cd 1` **Obs:** (Action rejected as hands are full). **Thought:** “I have found the second cd (cd 1) on desk 1. Now I need to go to the safe and put both cds into it.” *(Correct multi-step planning)*
Procedural Divergence	**Thought:** “I need to move cd 2 into the safe... I will move cd 2 into it.” **Action:** `move cd 2 to safe 1` *(Hallucination)*	**Thought:** “I have placed the first cd (cd 3) into the safe... Now I need to find the second cd (cd 1)... **I will go to desk 1 to retrieve the second cd**.” **Action:** `go to desk 1`
Execution	**Action:** `move cd 2 to safe 1` → `move cd 2 to safe 1` ... **Obs:** Repetition loop detected.	**Action:** `take cd 1 from desk 1` → `go to safe 1` → `move cd 1 to safe 1`
Result	**Failed: Text-Action Disconnect** (Understands the instruction but fails the execution)	**Success: Task Solved** (Implicitly adheres to the sequential retrieval pattern)

Prompt Template for Automated Feature Annotation

We are analyzing features learned by a sparse dictionary trained on the hidden states of an LLM agent solving interactive tasks.
Environment: ALFWorld is a text-based household environment where an agent must complete tasks like finding objects, cleaning them, heating/cooling them, and placing them in target locations. The agent navigates rooms, opens containers, picks up objects, and uses appliances through text commands.
Each "feature" is a direction in the model's representation space that activates on specific behavioral patterns. Your job is to identify what behavioral pattern causes Feature [ID] to activate, and whether it is associated with effective or ineffective agent behavior.
Top Activating Examples (sorted by activation strength)
Example 1 (activation=[val], reward=[val])
Observation: [text]
Action: [text]
[... additional examples ...]
Non-Activating Examples (Feature [ID] does NOT activate on these)
Example 1 (reward=[val])
Observation: [text]
Action: [text]
[... additional examples ...]
Outcome Statistics
- This feature activates in [X]% of all steps
- Activation rate on successful trajectories: [X]%
- Activation rate on failed trajectories: [X]%
- Mean activation strength on successful trajectories: [val]
- Mean activation strength on failed trajectories: [val]
Action Type Distribution (when this feature is active)
The following shows ALL action types in this environment, sorted by how much more likely they are to co-occur with this feature compared to the overall action distribution (ratio > 1 means over-represented, < 1 means under-represented):
- [action type]: [ratio]x (P(active|action)=[X]%, mean_activation=[val])
[... additional action types ...]
Instructions Based on ALL the evidence above (examples, outcome statistics, and action distribution), provide a comprehensive interpretation of this feature:
1. Identify the common behavioral pattern across the activating examples that distinguishes them from the non-activating examples.
2. Provide a short label (3-8 words in English) for this feature. The label should reflect both WHAT the behavior is and WHETHER it contributes to task success or failure.
3. Provide a one-paragraph explanation of what this feature detects, integrating evidence from the examples, outcome statistics, and action type distribution.
4. Classify the feature's polarity: "positive" if it is associated with effective/successful behavior, "negative" if associated with ineffective/failing behavior, or "neutral" if no clear association.
5. Rate your confidence (high/medium/low) in this interpretation.
Respond in JSON format:
{"label": "...", "explanation": "...", "polarity": "positive | negative | neutral", "confidence": "high | medium | low", "key_evidence": ["...", "..."]}

Figure 8: Prompt template for automated interpretation of geometric basis directions.

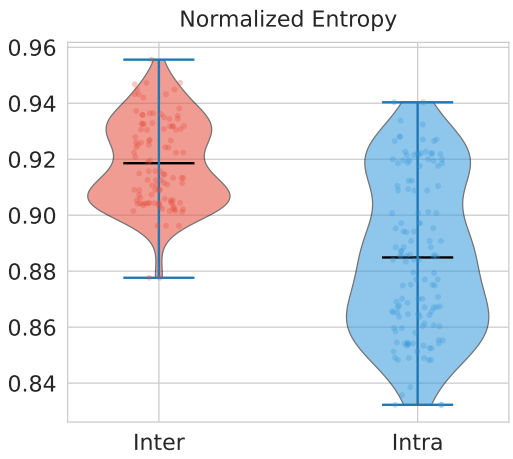


Figure 9: Normalized entropy for inter-trajectory and intra-trajectory steering vectors across the ALFWorld benchmark.

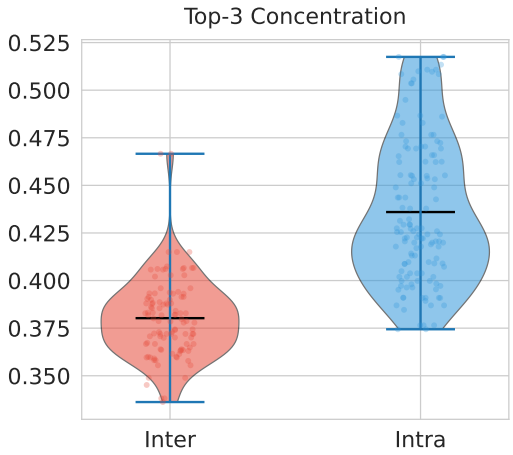


Figure 10: Top-3 concentration for inter-trajectory and intra-trajectory steering vectors across the ALFWorld benchmark.

Example of Insights Baseline

Task Instruction:
"i am looking for wild caught, ready to eat sardines in a tomato sauce, and price lower than 50.00 dollars"
Insights:

- Ensure the product's title and description explicitly mention the requested item type, size, color, and price range to avoid misselection.
- Always verify that the product's price is clearly stated and falls within the user's specified budget range to avoid overpayment.
- Prioritize products with clear and detailed descriptions that explicitly confirm the product meets all requested specifications, including intended use, size, and color.
- Ensure the product's size is explicitly mentioned in both the title and description and matches the exact requested dimensions to avoid purchasing an incorrect product.
- Always confirm that the product's title and description explicitly mention the requested use case, color, and price to ensure alignment with the user's requirements.
- Verify that the product's key features explicitly include the requested functionality (e.g., wireless, battery inclusion) in both the title and description.
- Prioritize products where the requested attributes (such as type, size, and price) are explicitly mentioned in both the title and description to minimize ambiguity.
- Always check the product's price in both the title and description before proceeding to purchase to avoid unexpected costs.

... *(total 20 insights provided in context.)*

Figure 11: Example of Insights Baseline for webshop

Example of Workflows Baseline

Task Goal:
put a clean soapbar in countertop.
Workflows:

- find_and_take_object
  When you need to find and take an object, search in likely locations.
  - go to {likely_location}
  - [if container is closed] open {container}
  - take {object} from {location}
- open_and_search_container
  When you need to open a container and search inside.
  - go to {container}
  - open {container}
  - [check contents]
- search_multiple_locations
  When you need to search multiple locations for an object.
  - go to {location 1}
  - go to {location 2}
  - go to {location 3}
  - [repeat as needed]
- clean_and_place_object
  When you need to clean an object and place it in a specific location.
  - go to {cleaning_location}
  - clean {object} with {cleaning_location}
  - go to {target_location}
  - move {object} to {target_receptacle}

...

Figure 12: Example of Workflows Baseline for AlfWorld