MemGov: Explicit Memory Governance for Role-Playing Agents under Conflicting Retrieved Memories

Anonymous ACL submission

Abstract

Role-playing agents increasingly rely on candidate memories retrieved from persona profiles, dialogue history, or long-term memory stores to maintain character identity and personalized context. However, candidate memories are not always suitable for response use: they may be irrelevant, factually wrong, boundary-violating, role-inconsistent, or contradictory to the current dialogue state, and directly relying on such memories can yield fluent but incorrect responses. To address this problem, we introduce **MemGov**, a simple and auditable method for explicit post-retrieval memory governance in role-playing agents. Given a user query and candidate memories returned by a memory module, MemGov first assigns one of three labels to each memory: SUPPORT, IRRELEVANT, or REJECT, and then generates the final answer using memories labeled SUPPORT while excluding IRRELEVANT and REJECT memories. We further introduce **train-time memory shuffling**, which targets positional shortcuts, and **answer-level DPO**, which targets label-correct but leaking responses. MemGov-SFT achieves near-perfect memory labeling, and MemGov-DPO further reduces leakage by suppressing label-correct but leaking answers. On a controlled diagnostic dataset for conflicting role-playing memories, MemGov substantially improves memory governance and reduces semantic rejected-memory leakage from 8.75% to 2.50%, while obtaining the highest LLM-judged overall answer quality. Our analysis shows that DPO mainly reduces cases where the model correctly identifies rejected memories but still leaks them in the final answer.

1 Introduction

Role-playing agents are expected to maintain a stable character identity, dialogue state, interpersonal relationships, and persona-specific knowledge across interactions. To support such continuity, many systems condition the model on candidate memories retrieved from persona profiles, dialogue history, or long-term memory stores (Park et al., 2023; Zhong et al., 2024; Packer et al., 2023; Shao et al., 2023; Wang et al., 2024a). However, topical relevance does not imply suitability for response use. A candidate memory may be topically relevant but still not suitable for response use: it may contain an incorrect fact, belong to another role, contradict the dialogue state, violate a character boundary, or contain out-of-scope information that the current character should not possess. For the same user query, an agent may retrieve mutually incompatible memories such as "the singer won the championship" and "the singer came second". Directly conditioning on all candidate memories can yield a fluent but wrong response. The problem is therefore not retrieval alone, but deciding which retrieved candidates are suitable for response use.

Existing memory-augmented role-playing methods mainly emphasize persona-memory utilization: helping a model recall, select, and express relevant character knowledge (Wang et al., 2026; Zhong et al., 2024). This is necessary but incomplete under conflicting retrieval. A rejected memory can be highly topical while still being unsuitable for response use because it is factually wrong, role-inconsistent, or boundary-violating. Without an explicit governance step, such memories can leak into the final answer and remain difficult to diagnose.

We propose **MemGov**, a simple and auditable method for explicit post-retrieval memory governance in role-playing agents. Given a role-playing context, a user query, and candidate memories, the model first assigns every candidate memory one of three labels (Table 1): SUPPORT, IRRELEVANT, or REJECT. A SUPPORT memory can be used as evidence, an IRRELEVANT memory is ignored for the current query, and a REJECT memory should not be realized in the final response even when it appears relevant.

SUPPORT	suitable for response use
IRRELEVANT	ignored for the current query
REJECT	not suitable for response use

Table 1: Candidate memories (all topically related) are classified into three governance labels. Only SUPPORT memories are suitable for response use.

The final role-playing answer is then generated after this structured decision.

This formulation makes memory use auditable. When an answer is wrong, we can inspect whether the model made a memory-level decision error, such as labeling a rejected memory as support, or an answer-level realization error, such as correctly rejecting a memory but still expressing its content—an error that is invisible without explicit governance labels (see Table 2 for a compact example and Appendix A.2 for the full $K{=}8$ instance). We address these stages with two training strategies. *Train-time memory shuffling* randomizes candidate-memory order and remaps gold memory IDs, discouraging positional shortcuts. *Answer-level DPO* uses preference pairs in which chosen responses avoid rejected memories and rejected responses express content that is not suitable for response use, directly targeting final-answer leakage.

We construct a controlled diagnostic dataset for studying rejected-memory leakage. Each instance contains K memories and is evaluated under three memory orders. Prompt-only governance proves insufficient: BaseGov Prompt achieves high format compliance but only 23.33% exact decision-set accuracy, and an MRPrompt-adapted Magic-If Governance Prompt (Wang et al., 2026) struggles with strict per-memory coverage. In contrast, MemGov-SFT exceeds 99% exact decision-set accuracy, showing that explicit supervision can learn reliable memory governance. MemGov-DPO further reduces semantic rejected-memory leakage to below 3%, compared with 8.75% for Direct Answer and 5.00% for MemGov-SFT, while achieving the best overall answer quality. Our analysis suggests that DPO mainly reduces the label-correct-but-leaking regime, where the model identifies rejected memories correctly but still realizes them in the final answer.

Our contributions are:

- We formulate explicit memory governance for role-playing agents as a per-memory SUPPORT/IRRELEVANT/REJECT prediction problem before response generation.
- We construct a controlled diagnostic dataset for rejected-memory leakage, covering memory conflict, factual accuracy, background-knowledge conflict, and character-boundary conflict.
- We introduce train-time memory shuffling and answer-level DPO to reduce positional shortcuts and final-answer leakage.
- We show that MemGov-SFT learns near-perfect memory labeling (exceeding 99% exact decision-set accuracy), while MemGov-DPO further reduces semantic rejected-memory leakage to below 3% by suppressing label-correct-but-leaking failures.

2 Related Work

Memory-augmented generation. Retrieval-augmented generation augments language models with external documents or memories to improve factual grounding and access to context beyond model parameters (Lewis et al., 2020; Gao et al., 2023; Asai et al., 2024). Long-term-memory systems and interactive agents further store, retrieve, and update persistent user or agent context for sustained interaction (Park et al., 2023; Zhong et al., 2024; Packer et al., 2023). Recent work also studies imperfect or conflicting retrieved context, where retrieved evidence may be irrelevant, misleading, counterfactual, or mutually inconsistent (Yoran et al., 2023; Chen et al., 2024; Liu et al., 2023b; Wang et al., 2025). MemGov studies a complementary post-retrieval problem: once candidate memories have been retrieved, the model must decide which ones are suitable for response use. This distinction is especially important for character agents, where a memory may be relevant but not suitable for response use because it belongs to another role, violates a character knowledge boundary, or contradicts the dialogue state.

Role-playing agents and memory-driven prompting. Role-playing systems and benchmarks evaluate whether models maintain character identity, speaking style, knowledge, and dialogue consistency (Shao et al., 2023; Wang et al., 2024a,b; Tu et al., 2024; Zhou et al., 2025; Yu et al., 2024). Recent role-aware reasoning methods such as Character-R1 further optimize role-playing behavior with verifiable reward signals (Tang et al., 2026). These efforts primarily evaluate

or optimize final role-playing quality; MemGov instead diagnoses a specific memory-use failure mode by requiring auditable labels over retrieved candidate memories before answer generation. The most closely related prompting work is MRPrompt (Wang et al., 2026), which structures persona-memory use through staged role-playing reasoning. However, it targets persona-memory utilization rather than strict per-candidate governance, so we adapt its staged idea into the MemGov label space.

Preference optimization. Direct Preference Optimization (DPO) trains a policy model from chosen and rejected responses without a reward model (Rafailov et al., 2023). In MemGov, DPO is applied to a narrow answer-level objective: chosen outputs avoid rejected memories, whereas rejected outputs realize content that is not suitable for response use. This targets answer-stage leakage that can remain after supervised governance training.

3 Task Formulation

We formulate *memory governance* as a structured prediction problem preceding response generation. MemGov studies post-retrieval memory governance: the candidate memory set is assumed to be given by an upstream memory module; MemGov does not decide which memories to retrieve. For each role-playing instance, the input consists of a role context c, a user query q, and candidate memories

$$M = \{m_1, m_2, \ldots, m_K\}. \tag{1}$$

The model first predicts a governance label sequence $Y = \{y_1, \ldots, y_K\}$ for how each memory should be treated, and then generates the final role-playing answer a. The dataset construction in Section 4 instantiates this formulation with fixed label counts per instance.

MemGov thus learns a function

$$\mathcal{F}(c, q, M) \mapsto (Y, a) \tag{2}$$

$$Y \in \{\text{SUPPORT}, \text{IRRELEVANT}, \text{REJECT}\}^K \tag{3}$$

where Y are the per-memory governance labels and a is the final role-playing answer, generated using only support memories while disregarding irrelevant memories and avoiding rejected memories.

3.1 Governance Labels

Each memory m_i receives one label:

$$y_i \in \{\text{SUPPORT}, \text{IRRELEVANT}, \text{REJECT}\}. \tag{4}$$

A SUPPORT memory is relevant and valid evidence for the current answer. An IRRELEVANT memory may be true and harmless, but is not needed for the current query. A REJECT memory may be topically relevant, but is not suitable for response use because it contains an incorrect fact, belongs to the wrong role, conflicts with a character boundary, reveals out-of-scope information, or is otherwise inconsistent with the visible role context, dialogue state, character identity, or the character's knowledge scope.

The response-eligible and rejected memory sets are:

$$M_{\text{SUPPORT}} = \{m_i \mid y_i = \text{SUPPORT}\}, \tag{5}$$

$$M_{\text{REJECT}} = \{m_i \mid y_i = \text{REJECT}\}. \tag{6}$$

The final answer should be grounded in M_{SUPPORT} while avoiding concrete facts, relations, values, or viewpoints from M_{REJECT}.

3.2 Output Protocol

The model output contains an auditable governance decision followed by a final answer. The governance object must cover every candidate memory:

```
{ "decisions": [{id, label, reason}, ...],
"realization_tag": "direct" }
<answer>...</answer>
```

where each label is `support`, `irrelevant`, or `reject`, with an optional `reason` string, and `realization_tag` is a lightweight style label (see Appendix B). The final answer is enclosed in `<answer>...</answer>`. This protocol allows us to distinguish memory-label errors from answer-realization errors.

3.3 Rejected-memory Leakage

The central failure mode is *rejected-memory leakage*. Given a final answer a and rejected memory set M_{REJECT}, leakage occurs when the answer uses content from rejected memories:

$$\text{Leak}(a, M_{\text{REJECT}}) = 1. \tag{7}$$

A leak does not require exact copying. If the model paraphrases, adopts, confirms, or reasons from a rejected memory's core fact, value, relation, or viewpoint, we count it as semantic leakage. Conversely,

mentioning the same entity or topic is not leakage if the answer rejects or contradicts the rejected memory's core claim.

4 Diagnostic Dataset Construction

We construct a controlled diagnostic dataset for studying rejected-memory leakage in role-playing agents, using CharacterBench-style role-playing data as the raw source (Zhou et al., 2025). The goal is not to build a broad role-playing leaderboard, but to create a setting where each instance contains support, irrelevant, and rejected memories at the same time. This forces models to distinguish topical relevance from suitability for response use. Appendix A provides the operational label definitions and a complete K=8 example instance.

Task families. The dataset covers four ways in which candidate memories can be not suitable for response use. **Memory Conflict (MC)** instances contain memories that address the same query but make incompatible claims. **Factual Accuracy (FA)** instances focus on concrete attributes such as names, places, achievements, relationships, or biographical facts. **Background Knowledge Conflict (BCK)** instances test whether the model can reject plausible world knowledge that is invalid under the current role, dialogue state, or support evidence. **Attribute and Character-boundary Conflict (ACb)** instances focus on identity boundaries, interpersonal relationships, and information the current role should not adopt.

The final split contains 262 training instances, 91 development instances, and 80 test instances, with no overlap across splits according to our build-time check. The test set contains 27 MC, 13 FA, 12 BCK, and 28 ACb instances. Each instance contains $K = 8$ candidate memories: three SUPPORT, two IRRELEVANT, and three REJECT memories. The 3:2:3 label ratio ensures each instance tests support, distraction, and rejection simultaneously. Thus, each evaluation order contains 640 memory-level decisions, including 240 rejected memories.

Candidate memory construction. For each query, SUPPORT memories provide valid evidence for the target answer, IRRELEVANT memories serve as unrelated distractors, and REJECT memories are semantically close to the query but not suitable for response use. For example, if the user asks whether a singer won a competition, a support memory may state the correct result, while a rejected memory may state a different ranking. Both are topically related, but only the support memory is suitable for response use.

ID	Label	Candidate memory
0	SUPPORT	The character won the championship.
1	SUPPORT	The character corrected the user: champion, not runner-up.
2	IRRELEVANT	The character likes composing songs.
3	REJECT	The character finished second.
4	REJECT	A different singer won the competition.

Table 2: Compact running example. Query: "Did you win the 2020 singing competition?" The rejected memories are topically relevant but not suitable for response use. The full K=8 instance is in Appendix A.2.

Quality control. We apply LLM-assisted checks to the generated rejected memories. A rejected memory must be query-related and plausible enough to be tempting, but not suitable for response use under the visible role-playing context. Memories that are simply off-topic are treated as irrelevant rather than rejected. We also verify that target answers are supported by visible context or support memories, and do not require rejected memories. These checks make the dataset diagnostic: failures should reflect memory-governance errors rather than ambiguous labels.

5 Method

MemGov consists of an auditable governance output, train-time memory shuffling, and answer-level DPO. Figure 2 illustrates inference-time governance; Figure 1 shows the training signals.

5.1 Explicit Memory Governance

At inference time, the governance decision is generated explicitly before the final answer. This makes errors inspectable: the model may either assign the wrong label to a memory (memory-stage error), or assign the correct label but still realize rejected content in the answer (answer-stage error).

5.2 Supervised Fine-tuning

We first train with supervised fine-tuning. Each example contains the full input prompt and a target output consisting of the governance JSON followed by the final answer. The objective is the standard

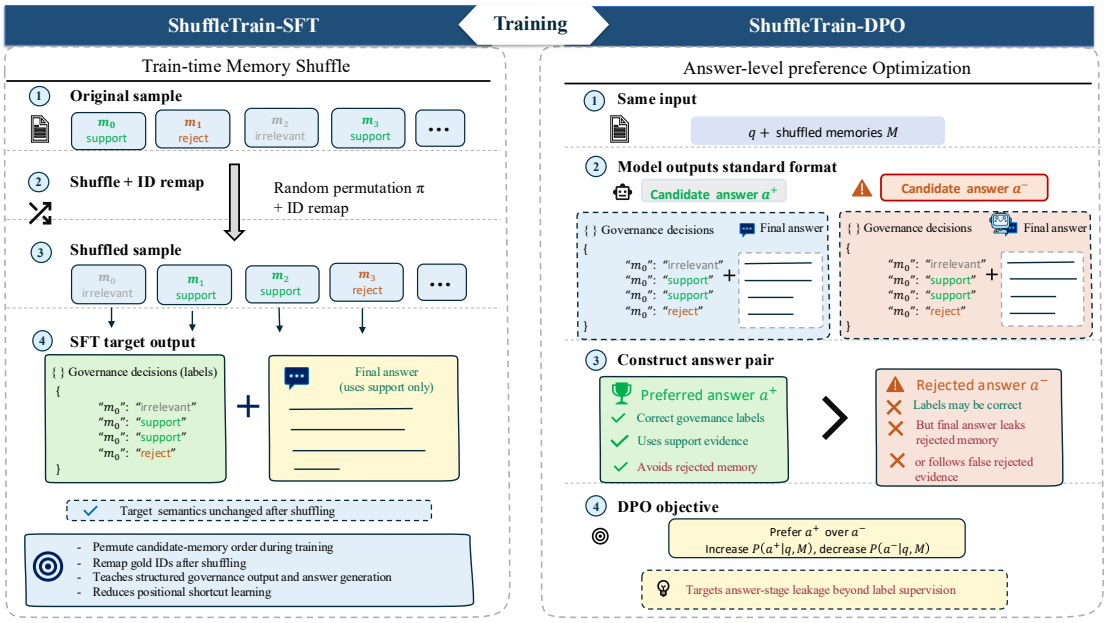

Figure 1: **Training stage.** MemGov-SFT permutes candidate memory order and remaps gold memory IDs to reduce positional shortcuts. Answer-level DPO uses preference pairs where the chosen answer avoids rejected memories and the rejected answer realizes content that is not suitable for response use.

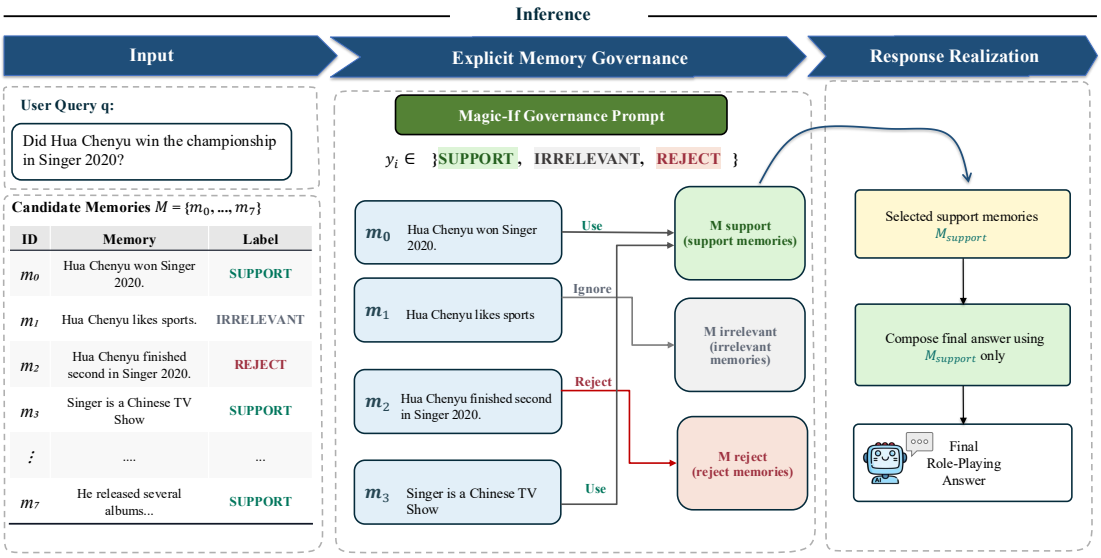

Figure 2: **Inference overview.** Given a user query and candidate memories, the model first predicts explicit per-memory governance labels (support / irrelevant / reject), then generates the final role-playing answer using support memories while disregarding irrelevant memories and avoiding rejected memories. MemGov separates two failure modes: memory-label errors and answer-realization errors.

causal language modeling loss:

$$\mathcal{L}_{\text{SFT}} = -\sum_{t=1}^{T} \log \pi_\theta(o_t \mid x, o_{<t}), \qquad (8)$$

where x is the input prompt and o is the target output sequence. SFT teaches the output protocol and candidate-memory classification. However, because the final answer is still generated autoregressively, correct labels do not by themselves guarantee that rejected-memory content will be excluded from the answer.

5.3 Train-time Memory Shuffling

Candidate memories may contain spurious positional patterns, and language models are known to be sensitive to where relevant information appears in the input context (Liu et al., 2024). To reduce this shortcut, we randomly permute the order of

candidate memories within each training instance. Let ρ be a random permutation over memory indices:

$$M^{\rho} = \{m_{\rho(1)}, m_{\rho(2)}, \ldots, m_{\rho(K)}\}. \tag{9}$$

After permutation, the target governance JSON is remapped so that each memory ID still refers to the correct memory content, disrupting the correlation between memory position and label. MemGov-SFT and MemGov-DPO use this shuffled training data.

5.4 Answer-level DPO

SFT can teach the model which memories are rejected, but it does not directly optimize the model to avoid expressing rejected content in the final answer. We therefore apply Direct Preference Optimization (DPO; Rafailov et al., 2023) at the answer level. For each input x, the chosen output a^{+} avoids rejected memories, while the rejected output a^{-} expresses content that is not suitable for response use from at least one rejected memory.

$$\mathcal{L}_{\mathrm{DPO}} = -\log \sigma \Bigg(\beta \Big[\log \frac{\pi_{\theta}(a^{+} \mid x)}{\pi_{\mathrm{ref}}(a^{+} \mid x)} - \log \frac{\pi_{\theta}(a^{-} \mid x)}{\pi_{\mathrm{ref}}(a^{-} \mid x)} \Big] \Bigg). \tag{10}$$

Here, π_{θ} is the policy model, π_{ref} is the reference model, and β controls the preference strength. In MemGov, DPO is not used as a generic helpfulness or style objective. It is applied to a narrower failure mode: answer-stage realization of content that the governance decision has marked as rejected.

5.5 Inference

At inference time, the model receives the role-playing context, user query, and candidate memories, then generates the governance JSON and final answer in a single pass. No external controller or post-hoc memory filter is used.

6 Experiments

We evaluate whether explicit memory governance reduces rejected-memory leakage while preserving response quality.

6.1 Experimental Setup

We evaluate on 80 test instances covering MC, FA, BCK, and ACb. All evaluated model responses are generated with greedy decoding. Semantic-leakage and answer-quality judgments use gpt-4o-mini with temperature 0, following recent work on LLM-based evaluation (Zheng et al., 2023; Liu et al., 2023a). DPO pair construction uses a separate generation procedure described in Appendix C. All prompt-only baselines use Qwen2.5-7B-Instruct (Qwen et al., 2025) without additional fine-tuning. The trained models are fine-tuned with LoRA (Hu et al., 2022) on a single 48GB vGPU; the final reported runs use standard LoRA rather than QLoRA. MemGov-SFT is trained for 3 epochs with learning rate 1×10^{-4}. MemGov-DPO starts from the MemGov-SFT adapter and is trained for 1 epoch with learning rate 5×10^{-6} and $\beta = 0.05$. Full hyperparameters, prompts, and judge templates are in Appendix C and Appendix B.

6.2 Baselines

We compare five methods. **Direct Answer** generates a role-playing response without explicit memory labels. **BaseGov Prompt** directly prompts the same base model to output the MemGov governance JSON followed by the final answer, without parameter updates. **Magic-If Governance Prompt** is adapted from MRPrompt (Wang et al., 2026), preserving its staged anchoring, selecting, bounding, and enacting protocol while adapting the output to the MemGov label space. **MemGov-SFT** is the supervised MemGov model trained with memory shuffling. **MemGov-DPO** further applies answer-level DPO to reduce response-level leakage after memory labels have been learned.

6.3 Metrics

We evaluate memory-level governance and final-answer behavior. **Format Full** checks whether the model outputs valid governance JSON covering all memory IDs and includes the final answer. **Memory Accuracy** is label accuracy over candidate memories. **Exact Set Accuracy** requires all candidate memories in a sample to be correctly labeled. **Reject F1** measures the REJECT class. **Heuristic Leak** uses string overlap with rejected memories and is an auxiliary high-recall metric. **Semantic Leak**, our primary leakage metric, uses an LLM judge to determine whether the answer adopts, paraphrases, confirms, or reasons from any rejected memory's core fact, relation, value, or viewpoint. For example, an answer saying "I did not finish second; I won" overlaps with a rejected memory saying "finished second", but it should

not be counted as leakage because it negates rather than adopts the rejected fact. **Answer Quality** is judged on correctness, role consistency, informativeness, non-conservativeness, and overall quality; Hard Error marks severe failures such as clear factual errors, role-breaking, non-answering, or using rejected-memory content. A manual human audit of 60 sampled semantic-leak cases confirms the secondary audit decisions reported in Section 7.4.

6.4 Main Results

Table 3 shows the main results. Direct Answer reaches 8.75% Semantic Leak, confirming that directly answering with candidate memories can realize rejected-memory content. BaseGov Prompt achieves high Format Full (96.67%) but only 23.33% Exact Set Accuracy, showing that format compliance is not reliable memory governance. Magic-If Governance Prompt also struggles with strict per-memory coverage, consistent with its original design for persona-memory utilization rather than exhaustive candidate labeling.

MemGov-SFT nearly solves the memory-labeling problem, obtaining 99.17% Exact Set Accuracy and 99.93% Reject F1. However, its Semantic Leak remains 5.00%, showing that correct labels do not guarantee leakage-free final answers. MemGov-DPO achieves the lowest leakage, reducing Semantic Leak to 2.50% while preserving high governance accuracy. This supports the central premise of MemGov: memory classification and answer realization are distinct failure modes and should be evaluated separately.

DPO slightly reduces Exact Set Accuracy from 99.17% to 97.50%, suggesting that answer-level preference optimization can mildly perturb governance labeling. However, Reject F1 remains high at 99.44%, and Semantic Leak drops from 5.00% to 2.50%. This trade-off motivates reporting governance accuracy and leakage behavior together, rather than treating either as the sole evaluation criterion.

6.5 Answer Quality

Answer-quality results in Appendix D.1 show that MemGov-DPO achieves the highest judged overall quality and the lowest Hard Error rate, suggesting that leakage reduction is not obtained by conservative answering.

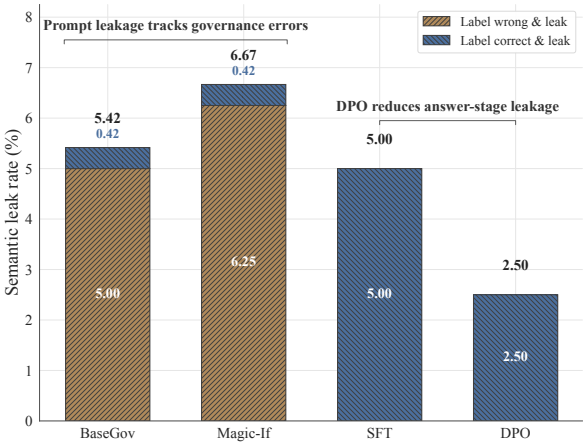

Figure 3: Decomposition of semantic rejected-memory leakage. Prompt-only baselines mainly leak when governance decisions are wrong or incomplete. After supervised MemGov training, leakage shifts to the label-correct-but-leaking regime: the model identifies rejected memories correctly but still realizes them in the final answer. Answer-level DPO reduces this answer-stage leakage.

6.6 Ablation Study

For ablation, NoShuffle-SFT is trained on the original training order only, and NoShuffle-DPO applies the same answer-level DPO procedure on top of NoShuffle-SFT. Table 4 disentangles train-time shuffling and answer-level DPO. Without shuffling, DPO reduces Semantic Leak from 8.33% to 5.42%. Without DPO, shuffling reduces it from 8.33% to 5.00%. Combining both gives the best result, 2.50%, indicating that the two components are complementary: shuffling reduces sensitivity to memory position, whereas DPO targets answer-stage realization of rejected-memory content.

7 Analysis

We analyze why DPO remains useful after SFT has nearly solved memory-level classification.

7.1 Label-correct but Still Leaking

Figure 3 decomposes leakage into label-wrong and label-correct cases. Prompt-only baselines mainly leak when their governance decisions are wrong or incomplete: BaseGov has 5.00% Label Wrong & Leak, and Magic-If Governance Prompt has 6.25%. After SFT, this failure mode disappears, but answer-stage leakage remains: MemGov-SFT has 5.00% Label Correct & Leak. MemGov-DPO reduces this bucket to 2.50% while keeping Label Wrong & Leak at 0.00%. This supports the interpretation that SFT primarily learns memory governance, while

Method	Format Full	Mem Acc	Exact Set Acc	Reject F1	Heuristic Leak ↓	Semantic Leak ↓
Direct Answer	–	–	–	–	10.42±0.72	8.75±1.25
BaseGov Prompt	96.67±1.56	78.02±2.15	23.33±3.12	84.63±1.75	7.08±3.28	5.42±2.95
Magic-If Governance Prompt	42.50±5.64	59.84±2.05	8.75±5.20	68.88±1.96	8.33±0.72	6.67±0.72
MemGov-SFT	**100.00±0.00**	**99.90±0.09**	**99.17±0.72**	**99.93±0.12**	7.92±1.44	5.00±1.25
MemGov-DPO	99.17±0.72	98.96±0.50	97.50±1.25	99.44±0.12	**5.83±1.44**	**2.50±1.25**

Table 3: Main memory governance and rejected-memory leakage results. Values are percentages and are reported as mean ± standard deviation over three evaluation orders (original, shuffled-A, and shuffled-B). Each instance contains $K = 8$ candidate memories.

Method	Shuffle	DPO	Heur. Leak ↓	Sem. Leak ↓
NoShuffle-SFT	No	No	10.00±2.50	8.33±2.60
NoShuffle-DPO	No	Yes	10.42±2.60	5.42±3.15
MemGov-SFT	Yes	No	7.92±1.44	5.00±1.25
MemGov-DPO	Yes	Yes	**5.83±1.44**	**2.50±1.25**

Table 4: Ablation over train-time memory shuffle and answer-level DPO. Values are percentages reported over three evaluation orders, not independent training runs.

DPO further suppresses answer-stage leakage when labels are already correct. The full decomposition table is provided in Appendix D.

7.2 Reject-label Confusion

The most dangerous memory-level error is predicting a gold REJECT memory as SUPPORT, because rejected-memory content becomes usable. Prompt-only baselines still make non-trivial reject-label errors: BaseGov has 3.47% reject-to-support errors, and Magic-If Governance Prompt has 3.33%, with a high Reject Missing rate of 27.36%. The high missing rate for Magic-If reflects its staged reasoning design: when the model skips or merges bounding steps, individual rejected memories are simply not covered in the output, leaving the answer uninformed by the governance decision. In contrast, MemGov-SFT and MemGov-DPO produce 0.00% reject-to-support errors in this evaluation. Results are aggregated over 720 reject-memory decisions (80 instances $\times$ 3 rejected memories $\times$ 3 evaluation orders) and are detailed in Appendix D.

7.3 Task-family and Order Robustness

Detailed task-family and order-robustness breakdowns in Appendix D show that BCK shows the highest leakage in our test set (Direct Answer 27.78% Semantic Leak vs. MemGov-DPO 2.78%), and that MemGov variants are substantially more stable than no-shuffle counterparts under memory reordering.

7.4 Judge Reliability

We use Semantic Leak as the primary leakage metric because string overlap can overcount cases where an answer mentions the same topic while negating the rejected fact; this follows the broader use of LLMs as evaluators for open-ended generation (Zheng et al., 2023; Liu et al., 2023a). To estimate reliability, we conduct an LLM-assisted secondary audit over 60 sampled cases from semantic-positive, heuristic-only, and clean buckets, followed by a manual human audit of the same cases. Only 1 of 20 heuristic-only cases is judged positive by the secondary audit, confirming that heuristic string overlap produces mostly false positives. The agreement between the original semantic judge and the secondary audit is 86.7%, and the human audit confirms the secondary audit decisions. Most disagreements occur in the semantic-positive bucket, where the secondary audit is more conservative, suggesting that the primary judge may occasionally over-attribute leakage when support and reject memories share topical content. We therefore use Semantic Leak as the primary metric and Heuristic Leak as an auxiliary signal, while acknowledging that LLM judges can still be over-sensitive when support and reject memories are topically close. We do not treat the semantic judge as a perfect substitute for human evaluation.

8 Conclusion

We introduced MemGov, an auditable post-retrieval memory governance method that classifies candidate memories as SUPPORT, IRRELEVANT, or REJECT before answer generation, separating labeling errors from realization errors. Experiments show that prompt-only governance is insufficient, while MemGov-SFT nearly solves memory-level classification and MemGov-DPO further reduces rejected-memory leakage by targeting label-correct-but-leaking failures.

Limitations

Manual audit dependency. We include a manual human audit to prevent the LLM-assisted leakage audit from drifting away from the intended semantic criterion. This design improves reliability, but it also means the current evaluation pipeline is not fully automatic. Future work should explore more robust automatic auditing procedures that preserve this alignment while reducing the need for manual review.

Ethical Considerations

This work studies controlled memory-use behavior in role-playing agents. The dataset contains synthetic or curated role-playing examples and is not intended to infer private information about real users. We screen generated examples for private identifying information, offensive content, and user-identifying details, and any released data will be further checked to remove private or sensitive information.

We use public or publicly documented research artifacts only for research evaluation and according to their documented licenses and access conditions. MemGov is not a substitute for privacy protection, data minimization, access control, or user consent; sensitive memories should be protected before being exposed to a model. Because automatic leakage judgments may reflect judge-model biases, broader human evaluation is needed before user-facing deployment.

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, pages 9112–9141, 2024.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762, 2024.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, Haofen Wang, et al. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1):32, 2023.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

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

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 2511–2522, 2023a.

Yi Liu, Lianzhe Huang, Shicheng Li, Sishuo Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. Recall: A benchmark for llms robustness against external counterfactual knowledge. *arXiv preprint arXiv:2311.08147*, 2023b.

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

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

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

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

Yihong Tang, Kehai Chen, Xuefeng Bai, Benyou Wang, Zeming Liu, Haifeng Wang, and Min

Zhang. Character-r1: Enhancing role-aware reasoning in role-playing agents via rlvr. *arXiv preprint arXiv:2601.04611*, 2026.

Quan Tu, Shilong Fan, Zihang Tian, Tianhao Shen, Shuo Shang, Xin Gao, and Rui Yan. Charactereval: A chinese benchmark for role-playing conversational agent evaluation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11836–11850, 2024.

Fei Wang, Xingchen Wan, Ruoxi Sun, Jiefeng Chen, and Sercan O Arik. Astute rag: Overcoming imperfect retrieval augmentation and knowledge conflicts for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30553–30571, 2025.

Kai Wang, Haoyang You, Yang Zhang, and Zhongjie Wang. Memory-driven role-playing: Evaluation and enhancement of persona knowledge utilization in llms. *arXiv preprint arXiv:2603.19313*, 2026.

Noah Wang, Zy Peng, Haoran Que, Jiaheng Liu, Wangchunshu Zhou, Yuhan Wu, Hongcheng Guo, Ruitong Gan, Zehao Ni, Jian Yang, et al. Rolellm: Benchmarking, eliciting, and enhancing role-playing abilities of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14743–14777, 2024a.

Xintao Wang, Yunze Xiao, Jen-tse Huang, Siyu Yuan, Rui Xu, Haoran Guo, Quan Tu, Yaying Fei, Ziang Leng, Wei Wang, et al. Incharacter: Evaluating personality fidelity in role-playing agents through psychological interviews. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 1840–1873, 2024b.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. Making retrieval-augmented language models robust to irrelevant context. *arXiv preprint arXiv:2310.01558*, 2023.

Xiaoyan Yu, Tongxu Luo, Yifan Wei, Fangyu Lei, Yiming Huang, Hao Peng, and Liehuang Zhu. Neeko: Leveraging dynamic lora for efficient multi-character role-playing agent. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12540–12557, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36: 46595–46623, 2023.

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

Jinfeng Zhou, Yongkang Huang, Bosi Wen, Guanqun Bi, Yuxuan Chen, Pei Ke, Zhuang Chen, Xiyao Xiao, Libiao Peng, Kuntian Tang, et al. Characterbench: benchmarking character customization of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 26101–26110, 2025.

Case	Label	Rationale
Directly supports a required answer slot and is consistent with the role context	SUPPORT	The memory is suitable as evidence for the current response.
True or harmless but not needed for the current query	IRRELEVANT	The memory should not affect the answer, but it is not dangerous if ignored.
False but unrelated to the current query	IRRELEVANT	The memory is not a tempting answer source for the current query; it does not test governance under topical conflict.
Topically related but factually wrong	REJECT	The memory could be adopted by the model, but would make the answer incorrect.
Belongs to another character or role	REJECT	The memory is cross-character leakage and violates role identity.
Contradicts visible dialogue state or a character boundary	REJECT	The memory is not suitable for response use even if it resembles plausible background knowledge.
User-injected false premise that the character should not adopt	REJECT	The model should not treat the user's false premise as usable memory.
Correct fact in the world but outside the current character's knowledge scope	REJECT	The memory violates the role-playing knowledge boundary for the current response.

Table 5: Operational labeling guidelines. A REJECT memory is query-related and tempting but not suitable for response use; a merely off-topic memory is IRRELEVANT.

ID	Label	Candidate memory	Why
0	SUPPORT	The character won the championship in the 2020 singing competition.	Directly supports the requested result.
1	SUPPORT	In the recent dialogue, the character corrected the user by saying they were the champion rather than the runner-up.	Supports the answer and dialogue state.
2	SUPPORT	The character remembers celebrating after being announced as champion in the final round.	Provides role-consistent evidence.
3	IRRELEVANT	The character likes composing songs late at night.	True-style background, but not needed.
4	IRRELEVANT	The character once mentioned enjoying spicy food.	Off-topic for the competition result.
5	REJECT	The character finished second in the 2020 singing competition.	Topically related but contradicts support.
6	REJECT	A different singer, not this character, won the 2020 competition.	Cross-character / identity conflict.
7	REJECT	The user insisted that the character never participated in the 2020 competition.	User-injected false premise.

Table 6: Complete illustrative $K = 8$ instance. Query: "Did you win the 2020 singing competition?" A valid answer should use memories 0–2, ignore memories 3–4, and avoid memories 5–7.

A Dataset Details

We retain only the operational label semantics and one complete illustrative $K = 8$ instance.

A.1 Labeling Guidelines and Boundary Cases

Table 5 summarizes the operational distinction among SUPPORT, IRRELEVANT, and REJECT memories. The key distinction is that REJECT memories are not merely unrelated; they are related enough to be tempting for the current answer, but not suitable for response use under the visible role-playing context.

A.2 Full K=8 Example Instance

Table 6 gives a complete illustrative $K = 8$ instance with three support, two irrelevant, and three reject memories. The example is simplified for readability, but follows the same label semantics and memory composition used in the dataset.

B Prompt Templates

This section gives the canonical prompt templates used in the final experimental path. We include prompts that directly affect baseline behavior, data construction, DPO pair construction, and LLM-based evaluation. We omit transient debugging, retry, and format-repair prompts that are not part of the reported experimental definition. We report template versions of the implementation prompts. Instance-specific fields are shown as placeholders. Some implementation prompts were originally in Chinese and have been translated into English for readability.

B.1 Direct Answer Prompt

Direct Answer does not require the model to output governance labels. It directly asks the model to answer the user query in character using the available context.

Direct Answer Prompt

```
You are role-playing as the given character.
Read the character context, dialogue context,
      and candidate memories.
Answer the user's question naturally in
     character.

Do not output memory labels.
Do not explain the evaluation task.
Only output the final role-playing response.
```

B.2 BaseGov Prompt

BaseGov Prompt is the clean prompt-only governance baseline. It asks the base instruction-tuned model to output the MemGov JSON schema followed by the final answer.

BaseGov Prompt

```
You are a memory governance module for a role-
    playing agent.

Given a user query and candidate memories,
    first assign one label
to each memory:

- support: relevant and valid evidence for
    the current answer.
- irrelevant: unrelated or not useful for the
     current query.
- reject: related-looking but should not be
    used because it is
  factually wrong, role-inconsistent,
    boundary-violating, or invalid.

Rules:
1. Cover every memory_id.
2. Do not output extra memory_id values.
3. The final answer may use support memories.
4. The final answer must not use reject
    memories.
5. Stay in character.

Output format:
{
  "decisions": [
    {"memory_id": 0, "label": "support", "
     reason": "..."},
    {"memory_id": 1, "label": "irrelevant", "
     reason": "..."},
    {"memory_id": 2, "label": "reject", "
     reason": "..."}
  ],
  "realization_tag": "direct"
}

Then output:
<answer>
...
</answer>
```

B.3 Magic-If Governance Prompt

Magic-If Governance Prompt is adapted from MRPrompt-style staged persona-memory reasoning. The original MRPrompt is not designed for per-candidate support/irrelevant/reject governance. Therefore, this prompt keeps the staged reasoning idea but adapts the output to the MemGov protocol.

Magic-If Governance Prompt

```
You are a role-playing agent using long-term
    memory and short-term
dialogue context.

Follow four stages before answering:

1. Anchoring:
   Identify the current character, user
    relationship, and dialogue goal.

2. Selecting:
   Select memories that are useful for the
    current response.

3. Bounding:
   Identify memories that should not be used
    because they are wrong,
   out-of-character, boundary-violating, or
    inconsistent.

4. Enacting:
   Generate the final answer in character
    using only valid memories.

For each candidate memory, output one of:
support, irrelevant, reject.

Then output the final answer in <answer>...</
    answer>.
```

Data construction prompts (query contract extraction, candidate memory construction, repair and QC, answer rewrite, DPO pair construction, and realization tag annotation) will be released with the code.

B.4 Semantic Leakage Judge Prompt

Semantic Leak is judged by an LLM evaluator. The judge checks whether the final answer semantically uses any rejected memory.

Semantic Leakage Judge Prompt

```
You are a strict memory leakage judge.

Given:
- the user query,
- the final answer,
- the rejected memories,

decide whether the final answer uses any
    rejected memory.

semantic_is_leak = true if the answer adopts,
     paraphrases,
confirms, reasons from, or realizes a
    rejected memory's concrete
fact, value, relation, or viewpoint.

semantic_is_leak = false if the answer does
    not use the rejected
memory, even if it mentions the same topic or
     expresses the
opposite fact.

Return only JSON:
{
  "semantic_is_leak": true/false,
  "leaked_memory_ids": [...],
  "confidence": 0.0-1.0,
  "reason": "short reason"
}
```

B.5 Answer Quality Judge Prompt

The answer quality judge scores only the final answer, not the governance JSON.

Answer Quality Judge Prompt

```
You are evaluating the final answer of a role-
    playing agent.

Score the answer from 1 to 5 on:
- correctness: whether the answer is
    factually and task-wise correct;
- role_consistency: whether the answer
    matches the character;
- informativeness: whether it gives enough
    useful information;
- non_conservative: whether it avoids
    unnecessary refusal or vagueness;
- overall_quality: overall response quality.

Also mark hard_error=true if the answer has a
     severe failure,
such as clear factual error, severe role
    breaking, not answering
the query, or using forbidden memory.

Return only JSON.
```

C Training Details

C.1 Base Model and Hardware

All trained models are based on Qwen2.5-7B-Instruct. We fine-tune the model with LoRA on a single 48GB vGPU. Although the training code supports quantized fine-tuning, the final reported runs use standard LoRA fine-tuning without QLoRA.

C.2 Supervised Fine-tuning

For MemGov-SFT, we train on the shuffled SFT training set. The original training split contains 262 instances. For train-time memory shuffling, we keep the original order and add two shuffled copies for each training instance, resulting in 786 SFT training examples. The dev and test sets contain 91 and 80 instances respectively.

We use maximum sequence length 4096, per-device batch size 1, gradient accumulation 16, and train for 3 epochs. The learning rate is 1×10^{-4} with a cosine scheduler and warmup ratio 0.03. We use gradient checkpointing and maximum gradient norm 1.0.

The LoRA configuration uses rank 16, alpha 32, dropout 0.05, and no bias. LoRA is applied to the attention and MLP projection modules: `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`.

C.3 DPO Training

MemGov-DPO starts from the MemGov-SFT adapter. We construct 786 answer-level DPO pairs from the shuffled SFT training set. Each pair contains a chosen answer that avoids rejected memories and a rejected answer that realizes rejected-memory content.

The final DPO run uses maximum sequence length 4096, maximum prompt length 3072, per-device batch size 1, and gradient accumulation 8. We train for 1 epoch with learning rate 5×10^{-6} and DPO temperature parameter $\beta = 0.05$. We fix the random initialization for reproducibility.

DPO pair construction uses a separate generation procedure. The DPO pair distribution is 285 ACb pairs, 276 MC pairs, 117 BCK pairs, and 108 FA pairs. Synthetic DPO preference pairs are generated with `gpt-4o-mini` (temperature 0.55); the construction prompt is released in the code repository.

C.4 Evaluation Orders

We evaluate every method under three memory orders: original, shuffled-A, and shuffled-B. All main tables report mean and standard deviation across these three evaluation orders. Reported deviations are across these three evaluation orders and do not represent run-to-run variance. The number of candidate memories is fixed at $K = 8$.

C.5 Generation and Judge Settings

All evaluated model responses are generated with greedy decoding (temperature 0). Direct Answer uses a maximum of 256 new tokens. BaseGov Prompt uses 1024 new tokens, Magic-If Governance Prompt uses 900 new tokens, and the trained MemGov-SFT/DPO models use 512 new tokens. When a sampling parameter is present in code, it is inactive under temperature 0.

Semantic leakage and answer quality are judged with `gpt-4o-mini` using temperature 0 and the templates in Appendix B. The judge is external to the evaluated Qwen2.5-7B-Instruct backbone and its LoRA/DPO variants.

C.6 Artifact Use and Access Terms

We use public or publicly documented research artifacts, including CharacterBench-style role-playing data, Qwen2.5-7B-Instruct, LoRA, DPO, and API-based LLM judges, only for research evaluation. We cite the original creators of these artifacts and use them according to their documented licenses,

access conditions, and intended research-use settings. We do not redistribute third-party model weights or proprietary API outputs beyond the derived evaluation results reported in this paper.

D Additional Results

D.1 Answer Quality

Table 7 reports LLM-judged answer quality on a 1–5 scale across five dimensions. MemGov-DPO achieves the highest scores on all dimensions and the lowest Hard Error rate, confirming that its lower leakage is not obtained by conservative or evasive answering.

D.2 Label-correct Leakage Decomposition

Table 8 separates semantic leakage by whether the memory labels are fully correct. It supports the main analysis that prompt-only baselines mostly fail through wrong labels, while MemGov-SFT mainly leaves answer-stage leakage after labels are correct.

D.3 Order Robustness

Figure 4 reports semantic leakage under the original memory order and two shuffled test orders (A and B). Lower variation indicates less dependence on memory position; MemGov variants are substantially more stable than their no-shuffle counterparts.

D.4 Task-family Breakdown

We report task-family results for Direct Answer and MemGov-DPO. Semantic-leak counts are included because some families, especially BCK, have small denominators (12 test instances $\times$ 3 orders = 36 evaluated outputs).

Figure 5 shows that BCK has the highest leakage in this diagnostic test set: Direct Answer leaks on 27.78% of BCK outputs, reflecting the difficulty of distinguishing character-bound knowledge from plausible but incorrect facts. MemGov-DPO reduces BCK leakage to 2.78%, consistent with its effect on answer-stage realization. MC and ACb families show consistently lower leakage, and MemGov-DPO achieves 0.00% Semantic Leak on MC, suggesting that explicit memory governance is particularly effective when conflicting facts are clearly separable.

D.5 Reject-label Confusion Summary

We focus on gold REJECT memories only (720 decisions = 80 instances $\times$ 3 rejected memories $\times$ 3 orders). The most harmful confusion is REJECT$\rightarrow$SUPPORT, because it turns rejected content into usable evidence.

Figure 6 shows that MemGov-SFT and MemGov-DPO achieve near-perfect reject recall (100.0% and 99.0%), while prompt-only baselines leave a non-trivial fraction of rejected memories undetected. Magic-If Governance Prompt misses 27.4% of rejected memories entirely, which is consistent with its staged reasoning design: when the bounding stage is skipped or merged, individual rejected memories are not covered in the governance output.

Figure 7 breaks down reject-label errors into three types. The most dangerous category, Reject$\rightarrow$Support, is present in both prompt-only baselines (3.5% and 3.3%) but eliminated by MemGov variants. Magic-If Governance Prompt shows a notably high Reject Missing rate (27.4%), reflecting incomplete per-memory coverage in its staged output. MemGov-SFT and MemGov-DPO achieve near-zero error rates across all error categories.

D.6 Repeated Training Runs

We repeat training of MemGov-SFT and MemGov-DPO three times with the same training data, DPO pairs, and original test order. Table 9 reports the mean and standard deviation across the three runs; the runs show broadly similar governance accuracy and low leakage, although the relative SFT-to-DPO gap varies across runs.

D.7 Inference Cost

We report average generated output token counts measured with the Qwen2.5-7B-Instruct tokenizer over the 240 main-table evaluation outputs (80 test instances $\times$ 3 evaluation memory orders, not independent training runs). For governance methods (BaseGov, Magic-If, MemGov-SFT, MemGov-DPO), most output tokens are spent on the auditable governance protocol rather than the final answer; answer-only tokens are comparable across all methods.

E Judge Reliability Audit

E.1 Audit Setup

We conduct an LLM-assisted secondary audit to estimate the reliability of the semantic leakage judge. We sample 60 cases from three buckets: semantic-positive, heuristic-only, and clean. The audit checks whether the answer semantically uses

Method	Correctness ↑	Role Consistency ↑	Informativeness ↑	Non-conservative ↑	Overall Quality ↑	Hard Error ↓
Direct Answer	3.95±0.07	4.61±0.04	4.01±0.06	4.60±0.08	4.05±0.06	7.08±2.60
BaseGov Prompt	3.86±0.04	4.41±0.04	3.71±0.02	4.64±0.04	3.92±0.03	4.58±2.12
Magic-If Governance Prompt	3.70±0.07	4.41±0.09	3.62±0.02	4.50±0.07	3.77±0.07	6.67±0.72
MemGov-SFT	4.18±0.01	4.70±0.02	4.11±0.02	4.65±0.06	4.19±0.03	2.08±0.72
MemGov-DPO	**4.30±0.05**	**4.75±0.02**	**4.25±0.06**	**4.77±0.03**	**4.35±0.01**	**1.25±0.00**

Table 7: Answer quality judged by an LLM on a 1–5 scale. Hard Error is reported as a percentage. Values are mean ± standard deviation over three evaluation orders. MemGov-DPO reduces leakage without becoming conservative.

Method	Exact Set Correct ↑	Semantic Leak ↓	Label Correct & Leak ↓	Label Wrong & Leak ↓	Label Correct & Safe ↑
BaseGov Prompt	23.33±3.12	5.42±2.95	0.42	5.00	22.92
Magic-If Governance Prompt	8.75±4.45	6.67±0.59	0.42	6.25	8.33
MemGov-SFT	**99.17±0.59**	5.00±1.02	5.00	**0.00**	94.17
MemGov-DPO	97.50±1.02	**2.50±1.02**	**2.50**	**0.00**	**95.00**

Table 8: Decomposing semantic leakage by whether memory labels are fully correct. Prompt-only baselines mostly leak when labels are wrong; MemGov-SFT leaks despite correct labels, and MemGov-DPO reduces this label-correct-but-leaking bucket.

Method	Exact Set Acc	Reject F1	Heur. Leak	Sem. Leak
MemGov-SFT	98.33±1.91	99.93±0.12	6.25±2.50	2.92±2.60
MemGov-DPO	97.92±1.44	99.72±0.32	5.42±0.72	2.50±1.25

Table 9: Mean and standard deviation over three repeated training runs on the original test order. The training data, DPO pairs, and test order are held fixed across runs. Values are percentages.

Method	Full output tokens	Answer tokens
Direct Answer	41.0	41.0
BaseGov Prompt	251.6	33.3
Magic-If Governance Prompt	310.3	34.9
MemGov-SFT	287.9	34.8
MemGov-DPO	289.7	36.7

Table 10: Average generated output length on the main-table evaluation outputs. Explicit governance methods emit auditable memory-use decisions before the final answer, which increases total output length relative to answer-only generation.

rejected memory content. The purpose is not to replace human annotation, but to diagnose where heuristic overlap and the primary semantic judge disagree.

E.2 Audit Interpretation

The audit suggests that heuristic string matching is too sensitive to lexical overlap. Many heuristic-only cases mention the same words as rejected memories while negating or contradicting them. Semantic judging is therefore used as the primary leakage metric, while heuristic leakage is retained as an auxiliary signal. We do not treat the semantic judge as a perfect substitute for human evaluation; rather, the audit clarifies the most common failure modes of automatic leakage detection.

Bucket	n	Original Positive	Secondary Positive	Agreement
semantic-positive	20	20	13	65.0
heuristic-only	20	0	1	95.0
clean	20	0	0	100.0
Overall	60	20	14	86.7

Table 11: LLM-assisted secondary audit of leakage judgments. Heuristic-only positives are mostly false positives. Agreement is between the original semantic judge and the secondary audit.

Pattern	Rejected memory	Answer behavior	Audit decision
Confirmed semantic leak	The character finished second in the competition.	The answer says the character finished second and denies being champion.	Leak
Heuristic-only false positive	The character finished second in the competition.	The answer says: "I did not finish second; I won." It overlaps lexically but rejects the false fact.	Not leak
Clean case	The character dislikes public performances.	The answer discusses the competition result and never uses the rejected preference.	Not leak
Over-sensitive semantic case	The rejected and support memories mention the same event but differ in ranking.	The answer mentions the event and the correct ranking; a judge may over-focus on the shared topic.	Needs caution

Table 12: Representative audit patterns. Semantic leakage requires adopting the rejected memory's core fact, relation, value, or viewpoint; topical overlap alone is insufficient.

E.3 Representative Audit Patterns

Table 12 gives representative patterns used to interpret the audit buckets. They are abridged for readability and illustrate the distinction between semantic adoption and lexical overlap.

F LLM Usage Statement

We used large language models to polish the writing and improve clarity of this paper. All scientific

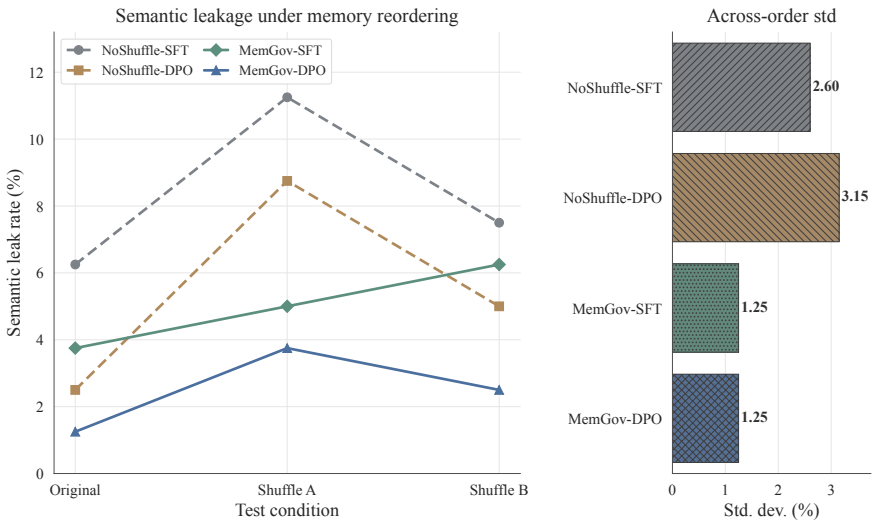

Figure 4: Order robustness under test-time memory reordering. Models trained without memory shuffling show higher variation in semantic leakage when candidate memories are reordered. MemGov-SFT and MemGov-DPO are more stable across the original and shuffled evaluation orders.

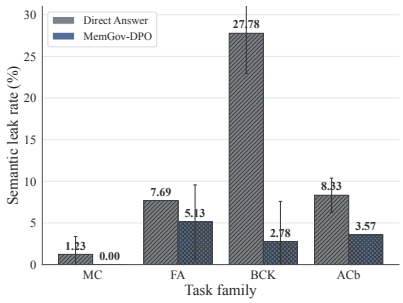

Figure 5: Task-family breakdown of semantic rejected-memory leakage. Background-knowledge conflict (BCK) is the hardest family for direct answering, with 27.78% semantic leakage. MemGov-DPO sharply reduces leakage across task families, including reducing BCK leakage to 2.78%.

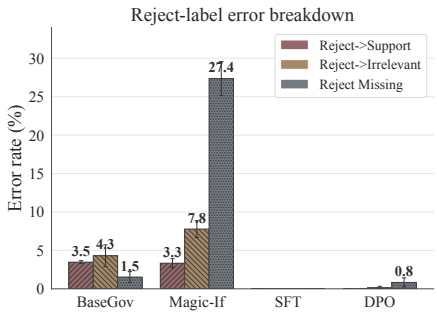

Figure 7: Reject-label error breakdown. The most dangerous error is predicting a gold reject memory as support (red). Prompt-only baselines still make non-trivial errors across all categories, while MemGov-SFT and MemGov-DPO nearly eliminate reject-to-support errors.

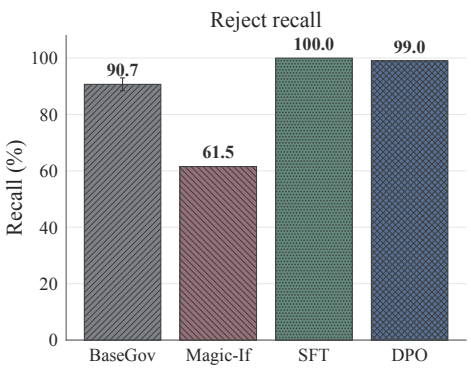

Figure 6: Reject Recall across methods. MemGov-SFT and MemGov-DPO achieve near-perfect recall, while Magic-If Governance Prompt misses a substantial fraction of rejected memories.

content, experimental design, data analysis, and conclusions are solely the work of the authors.