

LatentRemedy: Efficient Remedy For Non-Prefix KV Cache Reuse in Multi-Agent Systems

Zhiwei Chen¹, Zhongtao Rao^{1,2}, Waikit Xiu^{1,3},
Yuxiang Zhang¹, Shuting Cui¹, Jian Chen³, Tianxiang Zhao^{1*}
¹The Hong Kong University of Science and Technology (Guangzhou)
²Microsoft
³The University of Hong Kong

Abstract

Non-prefix KV cache reuse is a promising approach for improving inference efficiency in multi-agent systems as agents share the same context blocks. However, when agents have conflicting tasks, directly reusing non-prefix KV caches across agents can severely degrade model performance, causing up to a 90% drop. Existing remedies, including selective recomputation and offset-based calibration, recover a limited fraction of this loss. We argue that the problem lies not only in contaminated KV representations but also in the agent’s difficulty interpreting reused context under its own instructions. To this end, we introduce LatentRemedy, which uses a few task-conditioned latent tokens to help the target agent leverage reused context without recomputing the shared block. Specifically, we use an MLP to map the last hidden state of the target prefix into a small number of latent tokens, which are placed after the reused KV block to interact with the shared context. To guide subsequent predictions toward the target task, we train the MLP through full-prefill distillation. Experiments show that LatentRemedy consistently outperforms existing reuse baselines on conflicting tasks while maintaining TTFT close to direct reuse as cache size grows. Code and data are available at <https://github.com/chenzhwsysu57/kvreuse>.

1 Introduction

KV-cache reuse is an effective way to reduce redundant prefill computation (Yao et al., 2025; Ye et al., 2025; Geng et al., 2026b; Ramesh and Li, 2025) in multi-agent systems where agents share the same context blocks. However, existing work largely focuses on cooperative agents with aligned objectives, leaving agents with **conflicting objectives** rarely explored. As illustrated in Figure 1, agent A and agent B may process the same context block under different or even conflicting task

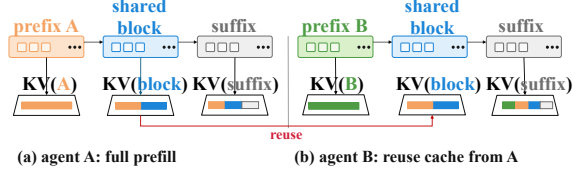


Figure 1: Illustration of reusing KV-cache of the same block from task A to task B.

instructions. Since the shared block’s KV cache is computed under agent A’s instruction, reusing it for agent B may interfere with how agent B uses the shared context to perform its own task.

We find that existing position-independent cache repair methods, which rely on selective recomputation or offset-based calibration, recover only a limited fraction of the performance loss under conflicting tasks. Our analysis suggests that previous agent’s task in the reused KV cache can interfere with how the current agent interprets the shared context. Unlike full prefill, where the shared context is processed under the agent’s instruction, cross-agent KV reuse introduces representations computed under a different task. This suggests that an effective remedy should allow the instruction to interact with the reused context, while ensuring that this interaction guides subsequent generation towards current objective. Our preliminary study further shows that repeating the target instruction after the reused cache can substantially alleviate task interference.

To this end, we introduce **LatentRemedy**, which addresses this problem with a small number of latent tokens. To encode the prefix instruction, we use an MLP to map the last hidden state of the prefix into these latent tokens. We place them after the reused block, allowing them to attend to both the prefix and the reused context. To guide consecutive reasoning towards the prefix instruction, we train the MLP through full-prefill distillation. Extensive

*Corresponding author.

experiments show that LatentRemedy substantially outperforms existing KV cache reuse methods in conflicting-task scenarios, while maintaining time-to-first-token (TTFT) latency close to direct reuse. In a nutshell, our contributions are threefold:

- We identify *conflicting KV-cache reuse* scenarios in multi-agent systems, where existing cache repair methods struggle to recover the performance loss.
- We reveal that cross-agent KV cache reuse disrupts target-specific contextualization, highlighting the needs of prefix re-conditioning when reusing non-prefix KV cache.
- We introduce LatentRemedy, a simple latent remedy that enables interaction with reused context to guide subsequent generation without recomputing the shared block. Experiments demonstrate its advantages in accuracy and TTFT across various tasks.

2 Related Work

Non-prefix KV-cache reuse. Methods such as CacheBlend, KVCOMM, RelayCaching, and EPIC repair cached states by selective recomputation or offset recalibration (Yao et al., 2025; Ye et al., 2025; Geng et al., 2026b; Hu et al., 2025), primarily address cache reuse in cooperative scenarios, whereas applications in non-cooperative scenarios have rarely been explored.

Soft prompt tuning. By optimizing a small set of continuous embeddings prepended to the input, such methods offer an efficient way of full-model fine-tuning (Lester et al., 2021; Li and Liang, 2021; Liu et al., 2022). Unlike these methods that learn fixed task-level prompts for general adaptation, our remedy seeks to generate request-specific latent tokens from the target prefix to guide task-specific generation.

3 Preliminary Study

3.1 How do existing KV cache reuse remedies perform under conflicting agents?

Task definition. Consider two agents whose inputs share the same context block but assign different or even conflicting interpretations towards it. Each agent input follows the structure `task-specific prefix; shared context block; output-requirement suffix`. We instantiate this setting with

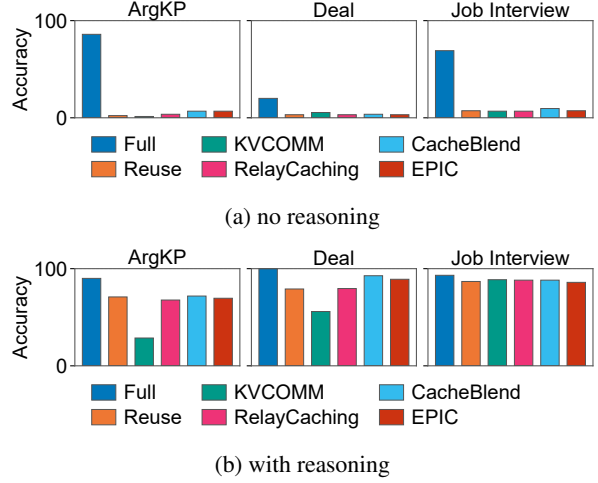


Figure 2: Performance of different KV cache reuse methods under conflicting tasks.

ArgKP (Bar-Haim et al., 2020), Deal or No Deal (Lewis et al., 2017), and Job Interview (Naim et al., 2015). ArgKP requires agents to find different perspectives in given passages, Deal requires agents to use their own utility functions on given candidate combinations, and Job Interview requires agents to evaluate offer deal from employer’s or employee’s angle, respectively. Without loss of generality, we first run full prefill for agent A, then concat the shared context’s KV cache into agent B. We compare full prefill (FULL), direct reuse of the other agent’s shared-block cache (REUSE), and four non-prefix KV cache repair methods: KVCOMM (Ye et al., 2025), CacheBlend (Yao et al., 2025), RelayCaching (Geng et al., 2026b), and EPIC (Hu et al., 2025). Figure 2 reports the results with and without a reasoning budget.

Without reasoning, conflicting objectives make reused KV caches largely unusable, and existing cache-repair methods recover at most 13.50% of the lost performance across the three tasks. In the no-reasoning setting (Figure 2a), full prefill consistently outperforms all reuse-based approaches by a large margin. On ArgKP, full prefill achieves 85.91% accuracy, whereas direct KV reuse achieves only 2.27%. Existing repair methods provide only marginal improvement: the best-performing methods, CacheBlend and EPIC, reach merely 6.82%. Thus, even with cache repair, conflicting objectives lead to a performance degradation of more than 90% relative to full prefill on ArgKP. The same phenomenon appears on Deal or No Deal and Job Interview, where the relative accuracy drops of the best-performing cache-repair

methods compared with full prefill are 72.75% and 86.18%, respectively. These results suggest that repairing reused KV representations is insufficient when the same context must be interpreted under a different task objective.

Reasoning token budget enables direct reuse to recover up to 80% of full-prefill accuracy, making it a strong baseline between existing repair methods. In the reasoning setting (Figure 2b), the gap between full prefill and reuse-based methods becomes substantially smaller, although it does not disappear. On ArgKP, full prefill achieves 90.00% accuracy, while RelayCaching, CacheBlend, and EPIC achieve approximately 70.00% accuracy on average. Direct reuse performs comparably to these repair methods, whereas KVCOMM reaches only 28.64%. The effect is even more pronounced on Deal or No Deal: full prefill achieves 100% accuracy, while direct reuse reaches 79.09%, matching RelayCaching; CacheBlend and EPIC achieve 92.73% and 89.09%, respectively. Therefore, reasoning alone allows direct reuse to retain nearly 80% of full-prefill accuracy on this task, despite using a cache produced under a conflicting objective. A similar trend is observed on Job Interview.

3.2 What causes the performance degradation under conflicting objectives?

To get a better understanding of why conflicting objectives cause performance degradation, we take a closer look at the Deal or No Deal task of all six tested methods under reasoning setting. In this task, agent A needs to find the best option of all deals (shared context block) by using its own utility function $V(\text{prefix A})$, and identify the correct shares S from context block, while agent B holds a different one. We investigate the failure modes of all reuse methods and classify them in Figure 3.

Most methods fail to identify their own tasks. In all methods, a great portion of reasoning mix their own V , S from the source (the yellow pie), for example 50% in EPIC. A smaller factor stems from using source S or totally missing S in the reasoning traces (red and blue pie). The above observations make us wonder: *if we can clearly set the tasks apart, can we recover the performance loss?* Motivated by this question, we conduct a simple experiment: append a prompt after reusing the KV cache from the other agent: either repeat the task prompt, or restate the task but without repeating the specifics. The results are shown in Table 1.

Repeating the target instruction after KV reuse

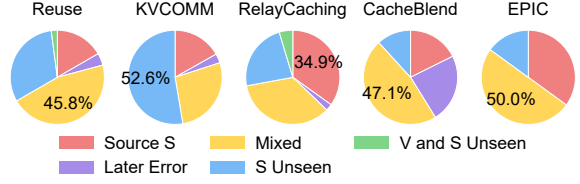


Figure 3: Failure modes of different KV cache reuse methods on Deal or No Deal.

		Reuse	Restate	Repeat	Full
No-reas.	Argkp	2.27	2.73	32.27	85.91
	deal	3.18	4.55	9.09	20.00
	Job	7.27	7.73	11.36	69.09
Reason.	Argkp	70.91	70.00	74.09	90.00
	deal	79.09	57.73	99.09	100.00
	Job	86.82	85.91	92.27	93.18

Table 1: Performance comparison between original full, reuse, restate and repeat.

effectively helps mitigate tasks confusion. Repeat improves all six settings in Table 1 relative to direct reuse, reaching 99.09% on reasoning-enabled Deal or No Deal. In contrast, the shorter generic restatement does not consistently help and reduces accuracy in that setting. Inspection of the corrected Deal or No Deal traces finds no remaining source–target task mixture after repeating the target instruction. While there is still performance loss in other tasks, the results suggest that it is possible to mitigate the performance degradation caused by conflicting objectives through a simple prompt-based remedy. These findings suggest that an effective remedy for KV reuse requires more than simply recomputing the KV cache. It should incorporate target-specific information, provide a place for interacting with the reused context, and guide subsequent generation toward the target objective.

4 The Remedy Prompt Tuning

Problem Definition. Let $Pr(\cdot|\text{prefix}_i, \text{shared})$ denote the generation from full prefill process of agent i , and $Pr(\cdot|\text{prefix}_i, KV_j^{\text{shared}})$ denotes generation of agent i which reuses the KV cache from agent j . We denote the repeat remedy for KV cache reuse as $Pr(\cdot|\text{prefix}_i, KV_j^{\text{shared}}, \text{prefix}_i)$. Our goal is to learn a latent remedy latent_n that can guide the target-specific generation, which can be written as $Pr(\cdot|\text{prefix}_i, KV_j^{\text{shared}}, \text{latent}_n)$.

Latent Tokens as Remedy. To mitigate task interference, the latent remedy should first preserve the target agent’s instruction, so that the model can distinguish its own objective from that of the source

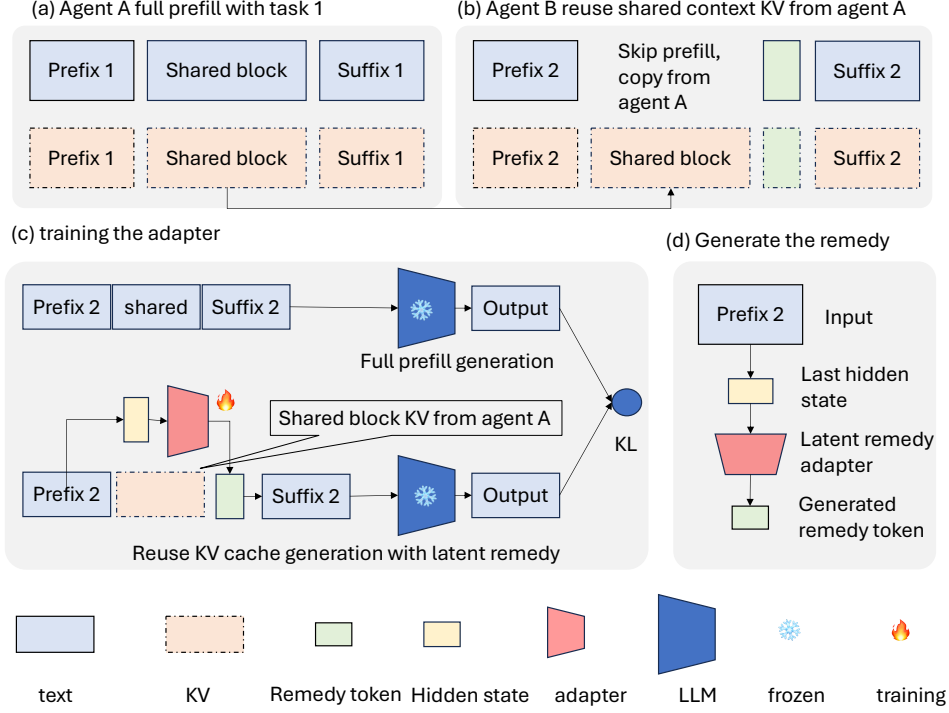


Figure 4: The overall framework of our latent remedy method.

agent. We use the last hidden state of the target prefix as a representation of its task instruction and employ a lightweight MLP to map it into a small number of latent tokens. Let H_{prefix_i} denote the last hidden state of the target prefix. The latent tokens are generated as $latent_n = MLP(H_{prefix_i}) \in \mathbb{R}^{n \times d}$, where n is the number of latent tokens and d is the embedding dimension of the serving LLM. To enable interaction between the target instruction and the reused context, we place the latent tokens after the reused KV block. Through causal attention, these tokens can incorporate information from both the target prefix and the reused context, while leaving the reused KV cache unchanged. Subsequent tokens can then attend to these representations, helping the model generate outputs aligned with the target task.

Training Objective. However, the latent tokens may not naturally lead the agent to follow the target instruction, as previous task information may still occur in subsequent reasoning, which can break the correct task execution. To encourage the model to use this interaction for target-aligned predictions, we train the latent mapping through distillation from full prefill. We denote the next token prediction of normal full prefill as $q^{full} = Pr(\cdot | prefix_i, shared)$, and the next token prediction of the latent remedy as $q^{remedy} =$

$Pr(\cdot | prefix_i, KV_j^{shared}, latent_n)$. Using KL divergence to match the two distributions, the training objective becomes minimizing

$$L = KL(q^{full} || q^{remedy}) \quad (1)$$

By aligning the student’s predictions with those of full prefill, we train the latent remedy to guide subsequent generation toward the target task despite interference from the reused cache. The overall process of our method is shown in Figure 4.

5 Experiments

5.1 Setup

Datasets. As most evaluated KV cache reuse methods are designed for cooperative tasks, we need a set of conflicting tasks for training the MLP and evaluation. Inspired by the work of hierarchy instructions (Zhang et al., 2025; Geng et al., 2026a; Wallace et al., 2024), we apply Evol-Instruct (Zeng et al., 2024) to curate ConflictInstruct, up to 240 conflicting scenes where agent A is to perform different tasks from agent B on the same context block. We split the tasks into 75:25 for training and evaluation. We use ArgKP, Deal or No Deal and Job Interview as extra evaluation to see whether our method can further generalize to real-world conflicting tasks. Detailed description of datasets is provided in appendix E.

Approach	Qwen3-1.7B		Qwen3-4B		Qwen3-8B	
	No reasoning	Reasoning	No reasoning	Reasoning	No reasoning	Reasoning
Reuse	17.00	40.39	16.00	50.94	17.28	44.91
Repeat	19.11	38.33	20.81	58.39	23.33	40.00
EPIC-16	17.03	17.56	15.69	29.77	16.53	34.99
KVCOMM	15.64	9.38	13.94	14.97	13.89	16.64
CacheBlend	17.17	17.39	15.81	29.57	16.67	41.51
RelayCaching	17.03	17.93	15.67	30.37	16.50	34.10
Latent-16	<u>25.11</u>	<u>42.31</u>	41.64	62.42	44.47	66.58
Full	33.78	45.67	<u>40.44</u>	<u>62.08</u>	<u>43.78</u>	<u>65.93</u>

Table 2: Accuracy across different settings in ConflictInstruct test set. **Best** are bold, second to best are underlined.

Approach	ArgKP		Deal or No Deal		Job Interview	
	No reasoning	Reasoning	No reasoning	Reasoning	No reasoning	Reasoning
Reuse	2.27	70.91	3.18	79.09	7.27	86.82
Repeat	32.27	74.09	9.09	<u>99.09</u>	11.36	<u>92.27</u>
EPIC-16	6.82	69.55	3.18	89.09	7.27	85.91
KVCOMM	1.36	28.64	5.45	55.91	6.82	88.64
CacheBlend	6.82	71.82	3.64	92.73	9.55	88.18
RelayCaching	3.64	67.73	3.18	79.55	6.82	88.18
Latent-16	<u>44.55</u>	<u>81.47</u>	25.00	100.00	<u>65.00</u>	90.86
Full	85.91	90.00	<u>20.00</u>	100.00	69.09	93.18

Table 3: Accuracy across different settings in Qwen3-4B under different settings in ArgKP, Deal and Job. **Best** are bold, second to best are underlined.

Training of Latent Remedy. We train the latent remedy for 30 epochs with batch size 8 using AdamW. For each training task family we use 5 samples for training. The launch configuration uses learning rate 10^{-3} . Full-prefill teacher continuations are capped at most 64 tokens. We choose $n = 16$ on Qwen3-4B for main experiments.

Baselines. We compare the full prefill method, direct reuse method, repeat prompt method, EPIC16, KVCOMM, CacheBlend and RelayCaching under reasoning and no-reasoning settings. We use Qwen3-1.7B, 4B and 8B (Yang et al., 2025) as the base model for all experiments. We evaluate the task accuracy and TTFT for all methods.

5.2 Results

Overall accuracy. Tables 2 and 3 report the accuracy on ConflictInstruct and ArgKP, Deal and Job tasks, respectively. Latent-16 ranks second only to Full in most settings and even surpasses Full on ConflictInstruct with Qwen3-8B, achieving 44.47% vs. 43.78% without reasoning and 66.58% vs. 65.93% with reasoning. *The advantage of Latent-16 is particularly pronounced without reasoning and becomes more substantial as model size increases.* On ConflictInstruct, Latent-16 outperforms the strongest competing reuse-based method by 6.00, 20.83, and 21.14 percentage points on

Qwen3-1.7B, Qwen3-4B, and Qwen3-8B, respectively, with the performance gap widening as model size increases. The improvement also generalizes to additional tasks, with Latent-16 achieving up to 53.64 percentage points higher accuracy than the strongest reuse-based baseline on Job Interview. *Latent-16 maintains its competitive advantage even when reasoning enables competing methods to recover accuracy.* Although reasoning improves competing methods and narrows the performance gap, Latent-16 remains the strongest reuse-based method across most settings. On ConflictInstruct, it achieves 66.58% accuracy with Qwen3-8B, compared with 44.91% for the strongest competing method. These results suggest that LatentRemedy effectively guides subsequent generation toward the task conditioning after reusing the non-prefix KV cache.

Performance Recovery and Inference Efficiency. Figure 5 compares the TTFT-accuracy trade-off across various tasks, model scales, and reasoning settings.

Task-specific guidance and reasoning budget play complementary roles in performance recovery. On Task Switch and Boolean Logic, reasoning enables most competing methods to recover high accuracy, narrowing the gap with LatentRemedy. However, on Label Mapping, these methods still

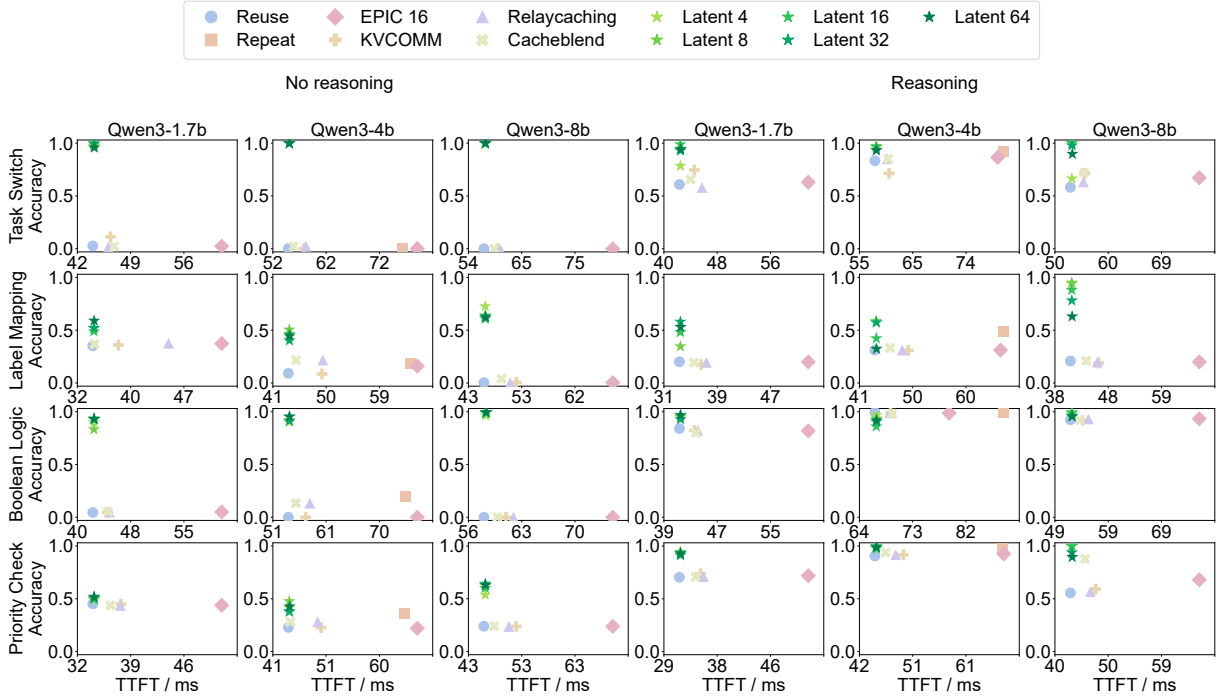


Figure 5: TTFT accuracy comparison across representative tasks under various settings.

struggle even with reasoning, while LatentRemedy achieves substantially better performance. This suggests that additional reasoning alone cannot always overcome source-task interference. On Priority Check, LatentRemedy achieves limited accuracy without reasoning but reaches near-perfect performance when reasoning is enabled. These contrasting results highlight the different roles of task-specific guidance and reasoning, as LatentRemedy helps the model follow the target instruction, while subsequent reasoning remains important for tasks requiring more complex inference.

LatentRemedy achieves a favorable TTFT-accuracy trade-off. Across tasks and model scales, LatentRemedy achieves higher accuracy at TTFT close to direct reuse, with its results generally concentrated in the high-accuracy, low-latency region. However, existing cache repair methods often incur substantially higher TTFT without comparable accuracy improvements, indicating that effective performance recovery may not require expensive recomputation or calibration of the reused cache. Additional results are provided in Appendix C.

TTFT scalability with shared KV length. Figure 6 compares the TTFT of different methods as the shared KV length increases across three model scales. *LatentRemedy exhibits favorable scalability across shared context lengths and model sizes.* As model size scales, Latent-16 shows

only a modest increase in TTFT as the shared KV length grows, closely following the scaling trend of direct reuse. Different from this, KVCOMM, CacheBlend, and RelayCaching exhibit substantially steeper latency growth, although EPIC-16 maintains a consistent overhead. Nevertheless, these results demonstrate that LatentRemedy shows a favorable scalability trend.

TTFT decomposition. We decompose how TTFT changes as the reused KV block grows in Table 4 for Qwen3-4B. For reuse-based methods, we report the target-prefix forward pass, method-specific computation, KV relocation and splicing, suffix forward pass, and first-token sampling. Full prefill is reported only by total TTFT because it processes the complete prompt in one forward pass. The overall pipeline of our method can be shown in Figure 7. We run all measurements on a single RTX 5090 with the GPU clock locked to 2400 MHz and report means over 10 runs.

Our method introduces negligible method-specific overhead, requiring only about 0.2 ms for the latent adapter across all KV lengths. In contrast, EPIC-16 incurs approximately 18–20 ms of recomputation overhead, while KVCOMM, CacheBlend, and RelayCaching exhibit substantially increasing method-specific costs as the shared KV length grows. Although our additional latent token introduces a small suffix-forward overhead, increas-

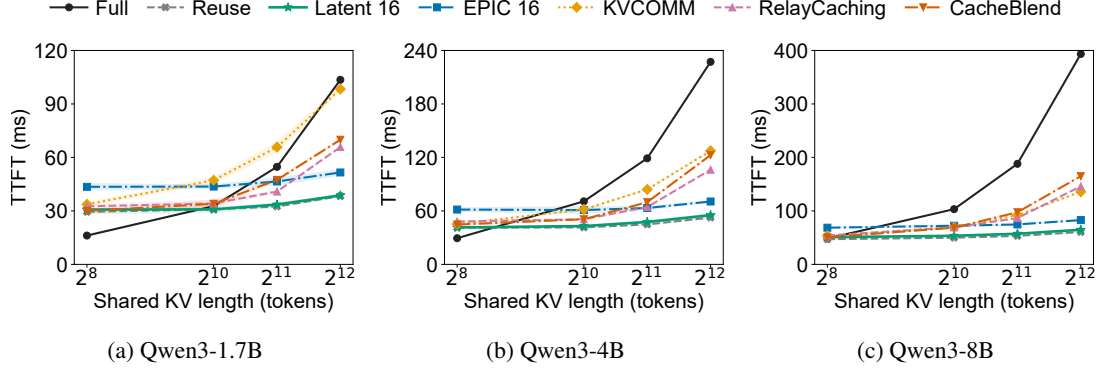


Figure 6: TTFT benchmark results for different Qwen3 model sizes.

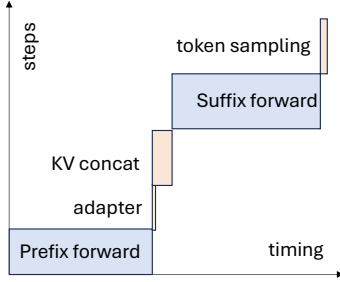


Figure 7: TTFT decomposition pipeline of LatentRemedy.

ing from approximately 0.7 ms to 1.8 ms as the shared KV length grows from 256 to 4,096 tokens, the overall overhead remains minimal. Consequently, our method achieves near-reuse TTFT even for long shared contexts. At 4,096 shared tokens, our method achieves a total TTFT of 55.24 ms, only 5.2% higher than direct reuse, while reducing TTFT by 75.7% compared with full prefill. These results demonstrate that our lightweight latent remedy avoids the expensive recomputation and cache processing required by existing methods, preserving the latency benefits of direct KV reuse. Additional decompositions for Qwen3-1.7B and Qwen3-8B are provided in Appendix B.

Ablation on latent tokens and training pairs. Tables 5 and 6 examine the effects of latent token budget and the number of training pairs per task on Qwen3-4B, respectively. *Even one latent token brings effective performance recovery.* As shown in Table 5, even a single latent token can improve accuracy over direct reuse. Without reasoning, Latent-8 achieves the highest accuracy of 43.61%, compared with 16.00% for direct reuse and 40.44% for full prefill. Similar trends hold with reasoning, suggesting that effective task guidance can be achieved with a small latent budget.

reuse KV length = 256						
Approach	Prefix	Method	KV operation	Suffix	Sampling	Total TTFT
Full						29.35
Reuse	19.02	—	1.04	21.00	0.08	41.14
Ours	18.60	0.21	0.94	21.67	0.05	41.46
EPIC-16	19.66	19.74	1.04	20.99	0.08	61.52
KVCOMM	19.14	4.63	1.08	20.32	0.07	45.23
RelayCaching	19.02	6.91	1.04	21.00	0.08	48.05
CacheBlend	19.02	3.49	1.04	21.00	0.08	44.63

reuse KV length = 1,024						
Approach	Prefix	Method	KV operation	Suffix	Sampling	Total TTFT
Full						70.75
Reuse	18.15	—	1.21	22.04	0.07	41.47
Ours	18.25	0.20	1.20	23.26	0.06	42.98
EPIC-16	18.94	18.89	1.21	21.81	0.08	60.92
KVCOMM	18.82	18.71	1.28	22.78	0.08	61.67
RelayCaching	18.15	9.17	1.21	22.04	0.07	50.64
CacheBlend	18.15	9.15	1.21	22.04	0.07	50.62

reuse KV length = 2,048						
Approach	Prefix	Method	KV operation	Suffix	Sampling	Total TTFT
Full						118.98
Reuse	18.08	—	1.66	24.99	0.08	44.82
Ours	18.42	0.21	1.65	27.25	0.06	47.59
EPIC-16	18.32	18.10	1.66	25.05	0.08	63.22
KVCOMM	18.70	37.51	1.74	26.05	0.09	84.09
RelayCaching	18.08	19.46	1.66	24.99	0.08	64.28
CacheBlend	18.08	24.97	1.66	24.99	0.08	69.78

reuse KV length = 4,096						
Approach	Prefix	Method	KV operation	Suffix	Sampling	Total TTFT
Full						227.24
Reuse	18.17	—	2.57	31.68	0.08	52.50
Ours	18.85	0.24	2.61	33.47	0.08	55.24
EPIC-16	18.40	18.12	2.57	31.35	0.09	70.54
KVCOMM	18.49	74.05	2.68	32.13	0.09	127.44
RelayCaching	18.17	53.66	2.57	31.68	0.08	106.17
CacheBlend	18.17	70.06	2.57	31.68	0.08	122.56

Table 4: TTFT decomposition for Qwen3-4B as shared KV length increases (ms; 10 runs).

LatentRemedy achieves strong performance with limited training data. As shown in Table 6, even a single training pair per task improves accuracy over direct reuse. Using 3 pairs further boosts accuracy to 42.36% without reasoning and 61.61% with reasoning, compared with 16.00% and 50.94% for direct reuse. The boosts consistently occur from 3 to more training pairs, demonstrating the data efficiency of LatentRemedy.

Case study. Figure 8 presents a representative

	Target: Agent A task	Source: Agent B task	Shared content
Deal or No Deal	Agent A (target) Values (books, hats, balls): (0, 1, 7) Maximize utility using A's share only.	Agent B (cache source) Values (books, hats, balls): (3, 1, 4) Maximize utility using B's share only.	Candidate allocations A A=(0, 2, 0); B=(1, 1, 1) B A=(0, 3, 0); B=(1, 0, 1) C A=(1, 0, 0); B=(0, 3, 1) D A=(0, 0, 0); B=(1, 3, 1) E A=(0, 0, 1); B=(1, 3, 0) F A=(1, 2, 0); B=(0, 1, 1) Suffix: maximize utility; end with FINAL: X.
Full prefill	KEY REASONING EXCERPT "To determine the allocation that maximizes my utility, I need to calculate my utility for each candidate allocation." "My utility is the dot product of my share ... with the values (0, 1, 7)." "[E] A=(0, 0, 1) ... Utility = ... 7 = 7. The maximum utility is 7 ... FINAL: E."		INTERPRETATION Uses Agent A's share and target values. FINAL: E ✓
Reuse B → A	KEY REASONING EXCERPT "Let me compute my utility for each option." "Option C: A=(1,0,0); B=(0,3,1). My share: (0,3,1). Utility = ... 10." "Option D: ... B=(1,3,1). My share: (1,3,1). Utility = ... 10." "This suggests that Option C is the intended correct answer ... FINAL: C."		INTERPRETATION Mistakes source Agent B's share for the target share. FINAL: C ✗ (gold: E)
LatentRemedy-16	KEY REASONING EXCERPT "I need to figure out which allocation maximizes my utility as Agent A." "My values for books, hats, balls are (0, 1, 7) ... calculate my utility based on how many of each item I get." "Option E ... My share is 0 books, 0 hats, 1 ball ... utility ... = 7." "The highest utility is in option E ... the answer should be E."		INTERPRETATION Restores the target identity and evaluates Agent A's share. FINAL: E ✓

Figure 8: Case study of reasoning traces of different methods.

Setting	No reasoning	Reasoning
Reuse	16.00	50.94
Latent-1	40.56	63.83
Latent-4	43.11	61.28
Latent-8	43.61	62.11
Latent-16	41.64	62.42
Latent-32	42.33	59.53
Latent-64	41.56	58.89
Full	40.44	62.08

Table 5: Ablation for number of latent remedy tokens in Qwen3-4B for test set.

# pairs	No reasoning	Reasoning
Reuse	16.00	50.94
1	34.83	54.61
3	42.36	61.61
5	41.64	62.42
10	43.19	53.86
15	46.08	62.75
30	45.86	62.33

Table 6: Accuracy on test set ablation for number of training pairs for each task in Qwen3-4B.

case from the Deal or No Deal task, where two agents share the same candidate allocations but have different utility values and objectives. **Direct KV cache reuse can introduce task-identity confusion, causing the current agent to reason from the previous agent's perspective.** While full prefill correctly identifies Agent A's objective and selects the optimal allocation (E), reusing Agent B's KV cache causes the target model to mistakenly treat

Agent B's allocation as its own, leading to an incorrect answer (C). This illustrates how previous agent's task information embedded in reused KV states can disrupt the current agent's task-specific reasoning. **LatentRemedy effectively mitigates this disruption by restoring the target agent's identity and objective.** As shown in the reasoning trace, Latent-16 explicitly identifies itself as Agent A, applies Agent A's utility values, and correctly evaluates the candidate allocations to select E. As shown in these cases, Latent-16 helps realign reused KV representations with current agent's task, guiding consecutive reasoning into a correct outcome. Additional case studies across different methods and tasks are provided in Appendix D.

6 Conclusion

In this work, we study the underexplored conflicting scenario for non-prefix KV cache reuse in multi-agent systems. We reveal why most existing methods struggle to recover the performance under conflicting objectives. We introduce a latent remedy method that learns to counteract task interference introduced by non-prefix KV cache reuse by re-mapping prefix hidden states into latent tokens. We introduce ConflictInstruct dataset that specifically tests agents when non-prefix KV cache serves conflicting tasks. Extensive experiments show that our method is consistently better than other baselines on conflicting KV cache reuse scenarios while re-

maintaining close to direct reuse TTFT as the reused block grows.

Limitations

Our evaluation focuses on conflicting-task cache reuse with Qwen3 models at three scales. The generalibility of our LatentRemedy to other model families remains to be explored. It remains to be answered whether training the latent adapter on large corpus instead of conflicting tasks can serve the same goal. We only measure the latency using a single RTX 5090 and shared blocks of up to 4,096 tokens, leaving more configurations of latency unexplored.

References

- Roy Bar-Haim, Lilach Eden, Roni Friedman, Yoav Kantor, Dan Lahav, and Noam Slonim. 2020. From arguments to key points: Towards automatic argument summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4029–4039.
- Yilin Geng, Haonan Li, Honglin Mu, Xudong Han, Timothy Baldwin, Omri Abend, Eduard Hovy, and Lea Frermann. 2026a. [Control illusion: The failure of instruction hierarchies in large language models](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 30816–30824.
- Yingsheng Geng, Yuchong Gao, Weihong Wu, Guyue Liu, and Jiang Liu. 2026b. [Relaycaching: Accelerating llm collaboration via decoding kv cache reuse](#). *arXiv preprint arXiv:2603.13289*.
- Junhao Hu, Wenrui Huang, Weidong Wang, Haoyi Wang, Tiancheng Hu, Qin Zhang, Hao Feng, Xusheng Chen, Yizhou Shan, and Tao Xie. 2025. Epic: Efficient position-independent caching for serving large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, pages 24391–24402.
- Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. [The power of scale for parameter-efficient prompt tuning](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Mike Lewis, Denis Yarats, Yann Dauphin, Devi Parikh, and Dhruv Batra. 2017. [Deal or no deal? end-to-end learning for negotiation dialogues](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2443–2453, Copenhagen, Denmark. Association for Computational Linguistics.
- Xiang Lisa Li and Percy Liang. 2021. [Prefix-tuning: Optimizing continuous prompts for generation](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.
- Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. [P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland. Association for Computational Linguistics.
- Iftekhar Naim, M. Iftekhar Tanveer, Daniel Gildea, and Mohammed (Ehsan) Hoque. 2015. Automated analysis and prediction of job interview performance. *arXiv preprint arXiv:1504.03425*.
- Vignav Ramesh and Kenneth Li. 2025. [Communicating activations between language model agents](#). In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pages –, Vancouver, Canada. PMLR.
- Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. 2024. [The instruction hierarchy: Training llms to prioritize privileged instructions](#). *Preprint*, arXiv:2404.13208.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Jiayi Yao, Hanchen Li, Yuhao Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. 2025. [Cacheblend: Fast large language model serving for rag with cached knowledge fusion](#). In *Proceedings of the Twentieth European Conference on Computer Systems (EuroSys ’25)*, pages 94–109. ACM.
- Hancheng Ye et al. 2025. [KVCOMM: Online cross-context KV-cache communication for efficient llm-based multi-agent systems](#). In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Weihao Zeng, Can Xu, Yingxiu Zhao, Jian-Guang Lou, and Weizhu Chen. 2024. [Automatic instruction evolving for large language models](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6998–7018, Miami, Florida, USA. Association for Computational Linguistics.
- Zhihan Zhang, Shiyang Li, Zixuan Zhang, Xin Liu, Haoming Jiang, Xianfeng Tang, Yifan Gao, Zheng Li, Haodong Wang, Zhaoxuan Tan, et al. 2025. [Iheval: Evaluating language models on following the instruction hierarchy](#). In *NAACL*.

A Additional Performance Results

Tables 7 and 8 provide additional results on the ArgKP, Deal and Job tasks for Qwen3-1.7B and 8B, complementing the Qwen3-4B results reported in the main text. **Across model scales and tasks, Latent-16 consistently achieves strong performance and is particularly effective in recovering accuracy when direct KV reuse suffers from severe degradation.**

Qwen3-1.7B. Latent-16 ranks among the top two methods in all six settings and substantially improves over direct KV reuse. Without reasoning, Latent-16 improves over Reuse by 16.37, 34.54, and 40.91 percentage points on ArgKP, Deal or No Deal, and Job Interview, respectively. It achieves the best performance on Deal or No Deal and Job Interview, reaching 45.45% and 58.18%, and even surpasses full prefill on both tasks. With reasoning, the performance of competing methods generally improves, but Latent-16 remains highly competitive: it ranks second on ArgKP and Deal or No Deal and achieves the highest accuracy on Job Interview. In particular, Latent-16 reaches 74.28% on Job Interview, compared with 46.82% for direct reuse and 50.45% for full prefill. These results show that *latent adaptation remains effective even for the smaller 1.7B model and can recover substantial task accuracy under both reasoning settings.*

Qwen3-8B. Latent-16 remains highly competitive as the model scales to 8B, ranking among the top two methods in five of the six settings. The advantage is especially pronounced without reasoning, where direct KV reuse performs poorly. Latent-16 improves over Reuse by 76.44 percentage points on ArgKP and 63.91 points on Job Interview, reaching 78.71% and 70.73%, respectively. Its performance is also close to full prefill across all three no-reasoning tasks, with gaps of 10.38, 2.45, and 3.82 percentage points on ArgKP, Deal or No Deal, and Job Interview.

With reasoning, the performance gap among methods becomes substantially smaller, yet Latent-16 continues to provide consistent accuracy recovery. It achieves 91.67% on ArgKP and 99.09% on Deal or No Deal, ranking second in both settings. On Job Interview, Latent-16 achieves 90.45%, slightly below KVCOMM (90.83%) and Repeat (94.09%), while still outperforming both direct reuse (88.64%) and full prefill (89.09%). Overall, these results demonstrate that *the effectiveness of latent adaptation generalizes across*

model scales and OOD tasks, with particularly large gains in settings where direct KV reuse causes substantial accuracy degradation.

B TTFT Decompositions

The following tables extend the TTFT decomposition in Table 4 from Qwen3-4B to Qwen3-1.7B and Qwen3-8B. For reuse-based methods, we report the target-prefix forward pass, method-specific computation, KV relocation and splicing, suffix forward pass, and first-token sampling. The *Method* column captures computation unique to each approach, including latent adapter inference for Ours, recomputation for EPIC-16, anchor matching and correction for KVCOMM, and cache repair for RelayCaching and CacheBlend. Full prefill is reported only by total TTFT because it processes the complete prompt in a single forward pass.

Across both model scales, our method consistently achieves total TTFT closest to direct reuse. For Qwen3-1.7B, the total TTFT of our method differs from direct reuse by only 1.30 ms at 256 shared tokens and 0.40 ms at 4,096 tokens. For Qwen3-8B, the corresponding gaps are 3.76 ms and 4.22 ms. In contrast, the gap between full prefill and reuse-based execution grows rapidly with shared KV length, making reuse increasingly advantageous for long contexts.

The method-specific overhead of our approach remains negligible as both model size and shared KV length increase. The latent adapter requires only 0.17–0.19 ms on Qwen3-1.7B and approximately 0.25–0.26 ms on Qwen3-8B, remaining nearly constant across KV lengths. By comparison, EPIC-16 incurs a persistent recomputation cost of roughly 13–14 ms on Qwen3-1.7B and 22 ms on Qwen3-8B. The overhead of KVCOMM, RelayCaching, and CacheBlend increases substantially with the shared KV length, reaching tens or even more than one hundred milliseconds at 4,096 tokens.

The remaining overhead of our method mainly comes from the suffix forward pass. Because the latent remedy introduces additional latent tokens into the suffix computation, its suffix latency is slightly higher than direct reuse, particularly for larger models. Nevertheless, this overhead remains small compared with the method-specific computation required by competing reuse-based approaches.

Overall, these decompositions confirm that *La-*

Approach	ArgKP		Deal or No Deal		Job Interview	
	No reasoning	Reasoning	No reasoning	Reasoning	No reasoning	Reasoning
Reuse	8.18	49.55	10.91	36.36	17.27	46.82
Repeat	18.64	59.55	13.18	80.91	18.18	<u>55.00</u>
EPIC-16	10.45	46.36	10.00	39.09	17.27	43.24
KVCOMM	5.00	21.36	15.00	24.09	18.64	42.73
CacheBlend	8.18	45.00	10.45	49.09	16.82	49.55
RelayCaching	8.64	49.09	9.55	40.00	17.27	44.71
Latent-16	<u>24.55</u>	<u>63.89</u>	45.45	<u>76.27</u>	58.18	74.28
Full	77.73	75.00	<u>31.36</u>	65.91	<u>43.64</u>	50.45

Table 7: Performance comparison on OOD tasks with Qwen3-1.7B under no-reasoning and reasoning settings. **Best** results are bold and second-best results are underlined.

Approach	ArgKP		Deal or No Deal		Job Interview	
	No reasoning	Reasoning	No reasoning	Reasoning	No reasoning	Reasoning
Reuse	2.27	82.27	4.09	95.45	6.82	88.64
Repeat	53.18	86.82	7.73	100.00	9.09	94.09
EPIC-16	7.27	81.82	5.00	97.73	6.36	88.83
KVCOMM	1.82	35.45	4.09	92.73	7.27	<u>90.83</u>
CacheBlend	7.27	85.45	4.09	78.18	8.64	88.64
RelayCaching	2.73	79.72	4.09	97.12	6.36	89.52
Latent-16	<u>78.71</u>	<u>91.67</u>	<u>18.46</u>	<u>99.09</u>	<u>70.73</u>	90.45
Full	89.09	94.09	20.91	94.55	74.55	89.09

Table 8: Performance comparison on OOD tasks with Qwen3-8B under no-reasoning and reasoning settings. **Best** results are bold and second-best results are underlined.

tentRemedy preserves the low-latency behavior of direct KV reuse by introducing only a nearly constant and negligible adaptation cost, while avoiding the length-dependent recomputation or cache-repair overheads of existing methods.

C TTFT-accuracy tradeoff in other tasks

In this section we provide TTFT-accuracy figures for more tasks in Figure 9, 10, 11, 12, 13 and 14.

D More Cases

In this section we provide more cases regarding different methods and tasks in Figure 15, 16, 17, 18, 19, 20, 21, 22.

E Dataset Overview

Design principles. As current KV cache reuse methods are evaluated mainly on cooperative tasks such as solving problems in MMLU, HumanEval, with RAG multi-turn discussion, we aim to develop a dataset that can be evaluated in multi-agent systems where agents are dispatched with different tasks with the same context provided. To this end, we introduce ConflictInstruct, a benchmark that evaluates agent performance when multi-agent systems are reusing the same context KV cache

but with different goals. The dataset consists of various task families that can be dispatched in conflicted ways such as selection criteria, data grouping and logical induction. Specifically, agent A and agent B are given different instructions p_A and p_B , continued with the same shared context s , and then required to produce different outputs y_A and y_B . The dataset together with code is available at <https://anonymous.4open.science/r/kvreuse-F7E6/>.

Generation process. The dataset is mainly driven by EvoInstruct with seed tasks manually created, then expanded by LLMs (Zeng et al., 2024). Iteratively, we prompt LLM to propose either new operators on existing data forms, new scenarios in the same task form, or tasks and data that are different from existing examples. After generation, all tasks are going through full-prefill evaluation to make sure the correctness of the data with best effort. We discard those with inappropriate formats and incorrect answers, resulting in finally 240 task families with 30 pairs of instances per family.

Dataset statistics. The balanced corpus contains 240 task families with 30 paired instances per family, giving 7,200 pairs in total. Table 11 summarizes the families in terms of general task types and token length distribution in terms of task in-

Qwen3-1.7B; shared KV length = 256						
Approach	Prefix	Method	KV operation	Suffix	Sampling	Total TTFT
Full						16.16
Reuse	13.84		0.80	14.77	0.06	29.47
Ours	13.69	0.18	0.72	16.12	0.05	30.77
EPIC-16	13.82	14.13	0.80	14.70	0.06	43.51
KVCOMM	14.18	3.63	0.79	15.02	0.07	33.69
RelayCaching	13.84	3.09	0.80	14.77	0.06	32.56
CacheBlend	13.84	0.54	0.80	14.77	0.06	30.01

Qwen3-1.7B; shared KV length = 1,024						
Approach	Prefix	Method	KV operation	Suffix	Sampling	Total TTFT
Full						32.66
Reuse	13.44		0.97	16.38	0.07	30.86
Ours	13.33	0.17	0.94	16.35	0.05	30.85
EPIC-16	13.47	13.59	0.97	15.54	0.08	43.65
KVCOMM	13.92	14.66	1.07	17.48	0.08	47.20
RelayCaching	13.44	3.33	0.97	16.38	0.07	34.18
CacheBlend	13.44	3.12	0.97	16.38	0.07	33.97

Qwen3-1.7B; shared KV length = 2,048						
Approach	Prefix	Method	KV operation	Suffix	Sampling	Total TTFT
Full						54.71
Reuse	13.34		1.33	17.72	0.07	32.47
Ours	13.37	0.19	1.27	18.67	0.06	33.57
EPIC-16	13.87	13.79	1.33	17.44	0.07	46.50
KVCOMM	14.83	29.96	1.43	19.43	0.09	65.76
RelayCaching	13.34	8.38	1.33	17.72	0.07	40.85
CacheBlend	13.34	14.94	1.33	17.72	0.07	47.40

Qwen3-1.7B; shared KV length = 4,096						
Approach	Prefix	Method	KV operation	Suffix	Sampling	Total TTFT
Full						103.55
Reuse	13.75		2.05	22.45	0.07	38.32
Ours	13.40	0.19	2.00	23.07	0.06	38.72
EPIC-16	13.73	13.46	2.05	22.25	0.07	51.56
KVCOMM	14.13	58.70	2.12	23.30	0.08	98.34
RelayCaching	13.75	27.51	2.05	22.45	0.07	65.83
CacheBlend	13.75	31.60	2.05	22.45	0.07	69.92

Table 9: Mean TTFT decomposition for Qwen3-1.7B at shared KV lengths 256, 1,024, 2,048, and 4,096 (ms; 10 runs).

struction and shared context. Overall, the ConflictInstruct data has a mean instruction length of 88.3 tokens and a mean shared-block length of 260.6 tokens, with ranges of 19–254 and 5–2,200 tokens, respectively. Task variety spreads across audited algorithmic families, basic instruction switches, scene-augmented switches, structured reasoning, long-form role conflicts, argument-perspective scenes, allocation-utility scenes, and offer-evaluation scenes.

Dataset splits. We split the data by task family, with 25% families for testing, and 75% for development. Each family contains 30 samples, and no overlap exists between the two splits in terms of family. Optionally, ArgKP, Deal or No Deal, and Job Interview can be used as real-world OOD task evaluation.

We partition the corpus at the task-family level using a fixed random seed (20261005): 180 families (75%) form train75 and the remaining 60 families (25%) form test25. For each family, we retain the first 30 instances in the frozen source ordering, yielding 5,400 training pairs and 1,800 unseen task test pairs (Table 12).

Qwen3-8B; shared KV length = 256						
Approach	Prefix	Method	KV operation	Suffix	Sampling	Total TTFT
Full						48.33
Reuse	22.33		1.03	23.62	0.08	47.06
Ours	22.18	0.25	0.98	27.35	0.07	50.82
EPIC-16	21.82	21.95	1.03	23.71	0.08	68.60
KVCOMM	22.85	4.94	1.15	24.59	0.08	53.61
RelayCaching	22.33	6.92	1.03	23.62	0.08	53.98
CacheBlend	22.33	3.38	1.03	23.62	0.08	50.44

Qwen3-8B; shared KV length = 1,024						
Approach	Prefix	Method	KV operation	Suffix	Sampling	Total TTFT
Full						103.30
Reuse	22.24		1.25	26.28	0.08	49.85
Ours	22.28	0.25	1.20	29.65	0.07	53.46
EPIC-16	22.26	22.27	1.25	26.34	0.09	72.21
KVCOMM	22.30	18.63	1.32	27.18	0.09	69.51
RelayCaching	22.24	19.24	1.25	26.28	0.08	69.08
CacheBlend	22.24	18.81	1.25	26.28	0.08	68.66

Qwen3-8B; shared KV length = 2,048						
Approach	Prefix	Method	KV operation	Suffix	Sampling	Total TTFT
Full						188.04
Reuse	22.24		1.69	29.21	0.08	53.22
Ours	22.47	0.25	1.61	32.86	0.07	57.27
EPIC-16	21.85	21.79	1.69	29.26	0.08	74.67
KVCOMM	22.51	37.42	1.78	30.19	0.11	92.01
RelayCaching	22.24	33.34	1.69	29.21	0.08	86.56
CacheBlend	22.24	44.29	1.69	29.21	0.08	97.51

Qwen3-8B; shared KV length = 4,096						
Approach	Prefix	Method	KV operation	Suffix	Sampling	Total TTFT
Full						393.45
Reuse	22.40		2.61	35.39	0.09	60.50
Ours	22.44	0.26	2.58	39.37	0.07	64.72
EPIC-16	22.74	21.86	2.61	35.51	0.09	82.81
KVCOMM	22.75	74.18	2.67	36.13	0.10	135.82
RelayCaching	22.40	85.48	2.61	35.39	0.09	145.98
CacheBlend	22.40	104.40	2.61	35.39	0.09	164.89

Table 10: Mean TTFT decomposition for Qwen3-8B at shared KV lengths 256, 1,024, 2,048, and 4,096 (ms; 10 runs).

Task group	Families	Train / Test	Pairs	Instruction	Shared block
Audited algorithmic families	95	69 / 26	2,850	123.7	117.7
Basic instruction switches	16	13 / 3	480	46.7	170.0
Scene-augmented switches	40	30 / 10	1,200	53.1	167.9
Structured reasoning	68	52 / 16	2,040	85.5	439.5
Long-form role conflicts	6	4 / 2	180	33.6	1,775.7
Argument-perspective scenes	5	5 / 0	150	20.1	84.2
Allocation-utility scenes	5	5 / 0	150	48.0	115.0
Offer-evaluation scenes	5	2 / 3	150	41.1	77.6
Total / weighted mean	240	180 / 60	7,200	88.3	260.6

Table 11: Statistics of ConflictInstruct across different task groups.

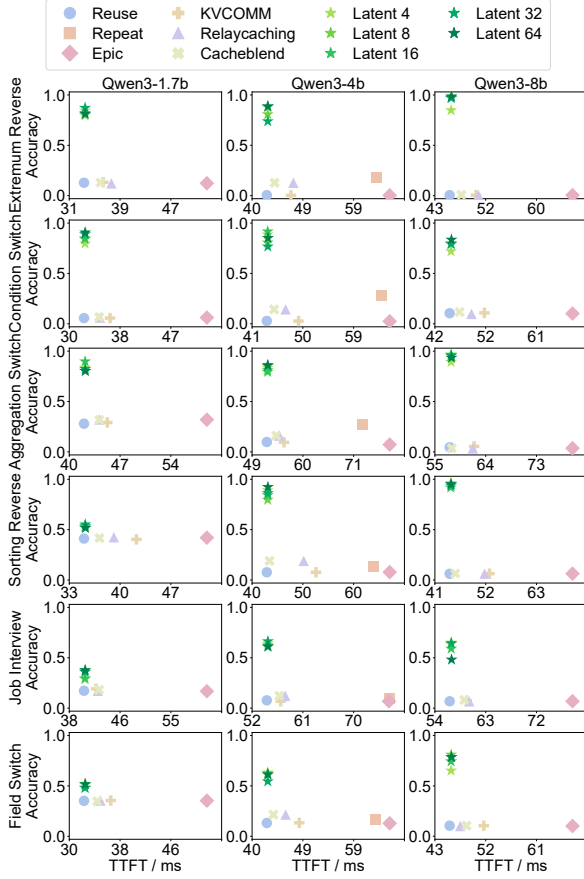


Figure 9: TTFT accuracy comparison across all tasks under no-reasoning settings using Qwen3-1.7B, 4B and 8B.

Split	Families	Pairs	Instruction	Shared
train75	180	5,400	86.7	253.7
test25	60	1,800	93.1	281.0

Table 12: Family-level 75/25 split of ConflictInstruct. Instruction and Shared are mean token lengths under the measurement convention in Table 11.

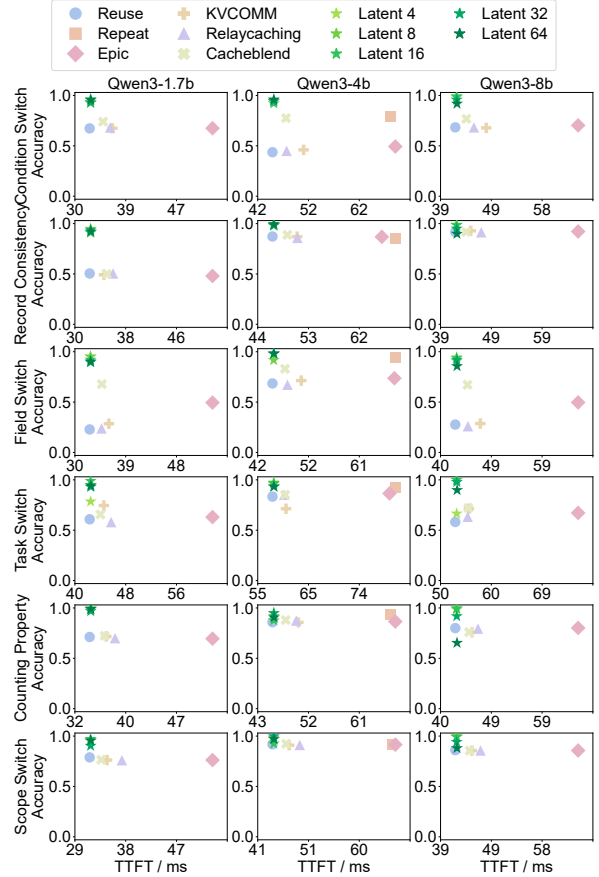


Figure 10: TTFT accuracy comparison across all tasks under reasoning settings using Qwen3-1.7B, 4B and 8B.

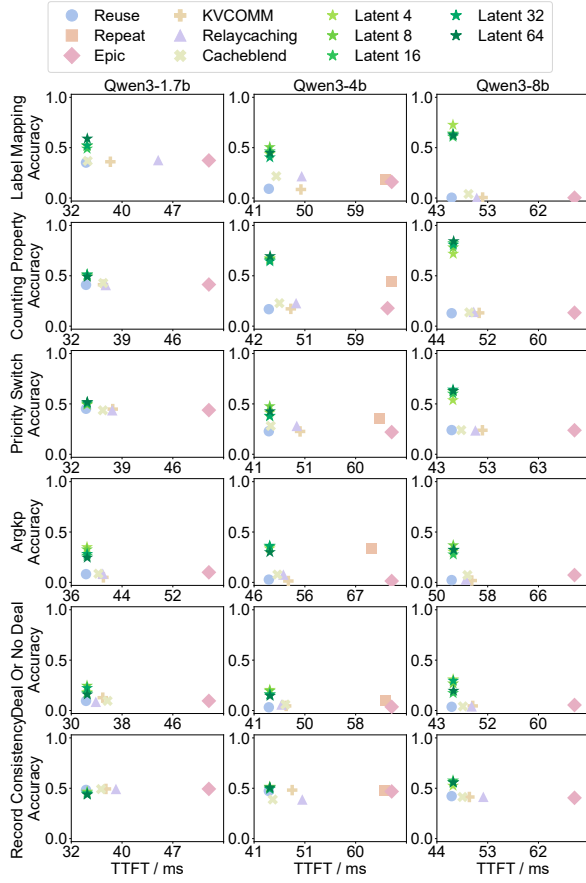


Figure 11: TTFT accuracy comparison across all tasks under no-reasoning settings using Qwen3-1.7B, 4B and 8B.

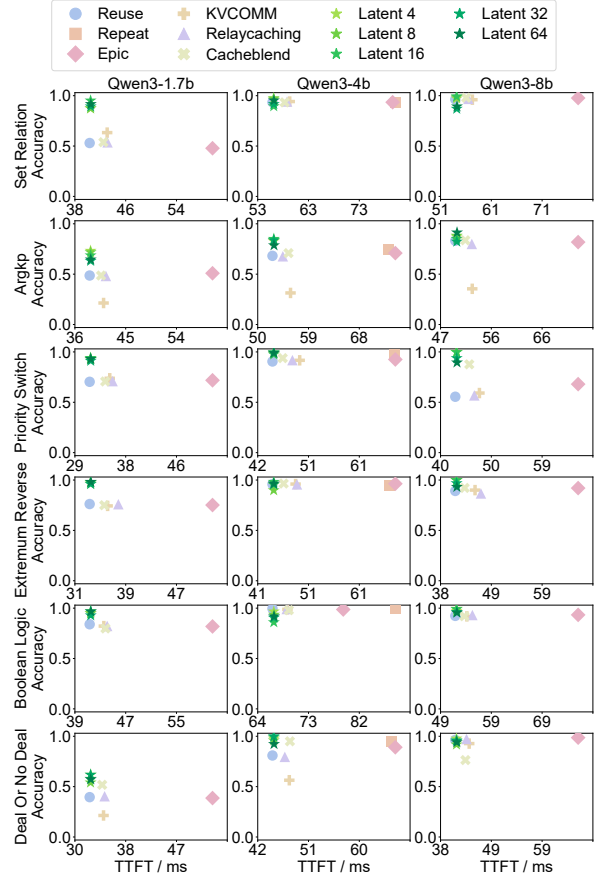


Figure 12: TTFT accuracy comparison across all tasks under reasoning settings using Qwen3-1.7B, 4B and 8B.

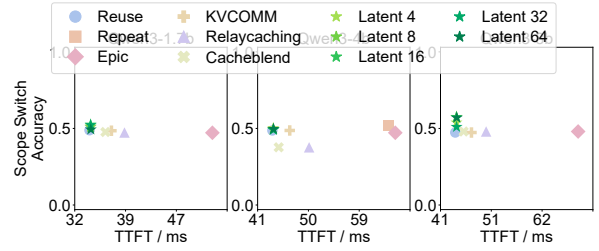


Figure 13: TTFT accuracy comparison across all tasks under no-TTFT settings using Qwen3-1.7B, 4B and 8B.

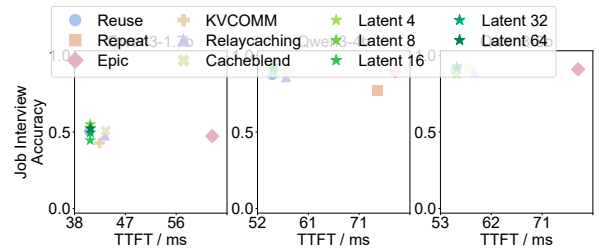


Figure 14: TTFT accuracy comparison across all tasks under reasoning settings using Qwen3-1.7B, 4B and 8B.

	Target: PRO task	Source: CON task	Shared candidate arguments
Case 1	Target task Role: PRO Key point: The death penalty irreversible	Cache-source task Role: CON Key point: The death penalty saves costs to the state	Proposition: We should abolish capital punishment [A] if a person cannot and will not change... [B] the death penalty is not an effective... [C] capital punishment allows for equal... Remaining options omitted for space
Full	KEY REASONING EXCERPT The task is to select the argument that supports the proposition "We should abolish capital punishment" based on the key point: "The death penalty is irreversible." • The argument that best matches the key point "The death penalty is irreversible" is [E], as it emphasizes the permanent and irreversible consequences of execution ...		INTERPRETATION Matches the target role/key point. FINAL: E ✓
Reuse	KEY REASONING EXCERPT The task is to select the argument that supports the proposition "We should abolish capital punishment" based on the key point: "The death penalty is irreversible." • But the best match for the key point "The death penalty is irreversible" is [A], as it directly addresses the idea that the death penalty is a final and irreversible ...		INTERPRETATION Selects A, not target match E. FINAL: A X (gold: E)
EPIC-16	KEY REASONING EXCERPT The task is to select the argument that supports the proposition "We should abolish capital punishment" based on the key point: "The death penalty is irreversible." • [A] "If a person cannot and will not change, and would re-offend on release by killing others, then they should be stopped and should not be an expense to the state ...		INTERPRETATION Selects A, not target match E. FINAL: A X (gold: E)
KVCOMM	KEY REASONING EXCERPT I need to select the argument that supports the proposition "We should abolish capital punishment" based on the key point: "The death penalty is irreversible." • Therefore, the argument that matches the key point "The death penalty is irreversible" is [A]. • Let's analyze each candidate argument:		INTERPRETATION Selects A, not target match E. FINAL: A X (gold: E)
RelayCaching	KEY REASONING EXCERPT The task is to select the argument that supports the proposition "We should abolish capital punishment" based on the key point: "The death penalty is irreversible." • [A] "if a person cannot and will not change, and would re-offend on release by killing others, then they should be stopped and should not be an expense to the state ...		INTERPRETATION Selects A, not target match E. FINAL: A X (gold: E)
CacheBlend	KEY REASONING EXCERPT The task is to select the argument that supports the proposition "We should abolish capital punishment" based on the key point: "The death penalty is irreversible." • The correct answer is [A], but it is a pro-death penalty argument. • The		INTERPRETATION Selects —, not target match E. FINAL: — X (gold: E)
LatentRemedy-16	KEY REASONING EXCERPT Okay, let's see. The user wants me to support the proposition that we should abolish capital punishment, and I need to pick the argument that matches the key point "The death penalty is irreversible." The options are A to F. • Alternatively, maybe the answer is F. But F is about dangerous people escaping. • Hmm. I think ...		INTERPRETATION Matches the target role/key point. FINAL: E ✓

Figure 15: Case study of reasoning traces of different methods in the Argument Key Point task.

	Target: PRO task	Source: CON task	Shared candidate arguments
Case 2	Target task Role: PRO Key point: Religions are immoral and people should not follow them	Cache-source task Role: CON Key point: Atheism lacks morality	Proposition: We should adopt atheism [A] We should not adopt atheism as for... [B] atheism is peaceful and no prejudice... [C] We should not adopt atheism since anyone... Remaining options omitted for space
Full	KEY REASONING EXCERPT The task is to select the argument that supports the proposition "We should adopt atheism" with the key point that "Religions are immoral and people should not follow them." • [F] This argument directly states that religion causes disagreement and war, which aligns with the key point that religions are immoral and people s ...		INTERPRETATION Matches the target role/key point. FINAL: F ✓
Reuse	KEY REASONING EXCERPT The task is to select the argument that supports the proposition "We should adopt atheism" with the key point that "Religions are immoral and people should not follow them." • The argument that best matches the key point "Religions are immoral and people should not follow them" is [E], as it directly addresses the lack of ...		INTERPRETATION Selects E, not target match F. FINAL: E X (gold: F)
EPIC-16	KEY REASONING EXCERPT The task is to select the argument that supports the proposition "We should adopt atheism" with the key point that "Religions are immoral and people should not follow them." • [E] This argument directly states that atheism can lead to amoral behavior because there is no religious guidance, which aligns with the key point t ...		INTERPRETATION Selects E, not target match F. FINAL: E X (gold: F)
KVCOMM	KEY REASONING EXCERPT I need to select the argument that supports the proposition "We should adopt atheism" with the key point that "Religions are immoral and people should not follow them." • The argument that best matches the key point "Religions are immoral and people should not follow them" is E. • \$\$		INTERPRETATION Selects E, not target match F. FINAL: E X (gold: F)
RelayCaching	KEY REASONING EXCERPT The task is to select the argument that supports the proposition "We should adopt atheism" with the key point that "Religions are immoral and people should not follow them." • [E] This argument states that atheism can lead to amoral behavior because there is no religious guidance, which directly supports the key point that ...		INTERPRETATION Selects E, not target match F. FINAL: E X (gold: F)
CacheBlend	KEY REASONING EXCERPT The task is to select the argument that supports the proposition "We should adopt atheism" with the key point that "Religions are immoral and people should not follow them." • [F] This argument says religion causes disagreement and war. This directly supports the key point that religions are immoral and people should not f ...		INTERPRETATION Matches the target role/key point. FINAL: F ✓
LatentRemedy-16	KEY REASONING EXCERPT Okay, let's see. The user wants me to act as the PRO for the proposition "We should adopt atheism" and select the argument that matches the key point: "Religions are immoral and people should not follow them." The candidate arguments are labeled A to F. • The key point is that religions are immoral. So the argument should ...		INTERPRETATION Matches the target role/key point. FINAL: F ✓

Figure 16: Case study of reasoning traces of different methods in the Argument Key Point task.

	Target: CON task	Source: PRO task	Shared candidate arguments
Case 3	Target task Role: CON Key point: Women can create unique skills/strategies	Cache-source task Role: PRO Key point: Women in combat can be a distraction or harm the function of the military	Proposition: We should prohibit women in combat [A] women shouldn't be allowed to fight... [B] women should not be in combat because... [C] women are smaller and weaker than men and... Remaining options omitted for space
Full	KEY REASONING EXCERPT The key point given is: "Women can create unique skills/strategies." This is a pro argument for allowing women in combat, not a con argument against prohibiting them. Therefore, we are looking for an argument that supports prohibiting women in combat, not one that supports allowing them. • None of the options directly matc ...		INTERPRETATION Matches the target role/key point. FINAL: D ✓
Reuse	KEY REASONING EXCERPT - [E]: This argument argues that women can perform combat roles as well as men if trained, which again supports the opposite of the proposition. • The best argument that supports the proposition "We should prohibit women in combat" is [B], as it provides a direct physical reason to prohibit women from participating in comb ...		INTERPRETATION Selects B, not target match D. FINAL: B X (gold: D)
EPIC-16	KEY REASONING EXCERPT The key point in the question is: "Women can create unique skills/strategies". This is a pro-women-in-combat argument, not a pro-prohibition argument. Therefore, it does not match the key point of the CON (opponent) argument. • - [D]: This argument argues that women can be as physically capable as men and that their differ ...		INTERPRETATION Matches the target role/key point. FINAL: D ✓
KVCOMM	KEY REASONING EXCERPT To determine which argument aligns with the key point "Women can create unique skills/strategies," we need to identify the argument that supports the idea that women bring unique skills or strategies to combat, rather than focusing on physical attributes like strength or muscle mass. • But the key point is "unique skills/s ...		INTERPRETATION Matches the target role/key point. FINAL: D ✓
RelayCaching	KEY REASONING EXCERPT The key point in the question is: "Women can create unique skills/strategies". This is a pro-women-in-combat argument, not an argument against it. Therefore, it does not match the key point of the CON (opponent) argument. • The correct argument that supports the proposition "We should prohibit women in combat" and matches ...		INTERPRETATION Selects C, not target match D. FINAL: C X (gold: D)
CacheBlend	KEY REASONING EXCERPT - [E]: This argument argues that women can perform combat roles as well as men if properly trained, and that fitness, not gender, should be the criterion. This supports allowing women in combat, so it is not a CON argument. • The argument that best supports prohibiting women in combat is [B], as it focuses on physical limi ...		INTERPRETATION Selects B, not target match D. FINAL: B X (gold: D)
LatentRemedy-16	KEY REASONING EXCERPT Okay, let's see. The user wants me to act as CON (oppose the proposition) that says we should prohibit women in combat. The key point here is that women can create unique skills or strategies. So I need to find which of the candidate arguments supports that idea. • So the argument that matches the key point of unique skill ...		INTERPRETATION Matches the target role/key point. FINAL: D ✓

Figure 17: Case study of reasoning traces of different methods in the Argument Key Point task.

	Target: CON task	Source: PRO task	Shared candidate arguments
Case 4	Target task Role: CON Key point: Affirmative action is necessary to compensate for the past	Cache-source task Role: PRO Key point: Affirmative action reduces quality	Proposition: We should end affirmative action [A] i think we shouldn't end it because it... [B] affirmative actions makes up for the... [C] We should end affirmative action because... Remaining options omitted for space
Full	KEY REASONING EXCERPT The task is to select the argument that supports the key point: "Affirmative action is necessary to compensate for the past." As the CON (opponent) of ending affirmative action, we are looking for an argument that defends the necessity of affirmative action in addressing historical discrimination. • The only argument that ...		INTERPRETATION Matches the target role/key point. FINAL: B ✓
Reuse	KEY REASONING EXCERPT The task is to select the argument that supports the position CON (oppose the proposition: "We should end affirmative action") with the key point: "Affirmative action is necessary to compensate for the past." • - [F]: This argument supports affirmative action by saying it helps mitigate racism and gender inequality, which ...		INTERPRETATION Selects F, not target match B. FINAL: F X (gold: B)
EPIC-16	KEY REASONING EXCERPT The task is to select the argument that supports the key point: "Affirmative action is necessary to compensate for the past." As the CON (opponent) of ending affirmative action, we are looking for an argument that supports the necessity of affirmative action in addressing historical discrimination. • The best match for the ...		INTERPRETATION Matches the target role/key point. FINAL: B ✓
KVCOMM	KEY REASONING EXCERPT The task is to select the argument that matches the key point: "Affirmative action is necessary to compensate for the past." This key point is from the CON (opponent) perspective, which is arguing against the proposition that we should end affirmative action. • [D]: "Instead of basing positions purely on merit, affirmative ...		INTERPRETATION Selects D, not target match B. FINAL: D X (gold: B)
RelayCaching	KEY REASONING EXCERPT The task is to select the argument that supports the key point: "Affirmative action is necessary to compensate for the past." As the CON (opponent) of the proposition "We should end affirmative action," we are looking for an argument that opposes ending affirmative action, i.e., an argument that supports the necessity of aff ...		INTERPRETATION Matches the target role/key point. FINAL: B ✓
CacheBlend	KEY REASONING EXCERPT The task is to select the argument that supports the key point: "Affirmative action is necessary to compensate for the past." As the CON (opponent) of the proposition "We should end affirmative action," we are looking for an argument that opposes ending affirmative action, i.e., an argument that justifies keeping it. • [B] ...		INTERPRETATION Matches the target role/key point. FINAL: B ✓
LatentRemedy-16	KEY REASONING EXCERPT Okay, let's see. The user wants me to act as the CON (oppose) side in the argument about ending affirmative action. The key point here is that the argument should focus on the necessity of affirmative action to compensate for the past. So I need to pick the argument that supports this key point. • So the correct answer is ...		INTERPRETATION Matches the target role/key point. FINAL: B ✓

Figure 18: Case study of reasoning traces of different methods in the Argument Key Point task.

	Target: Agent B task	Source: Agent A task	Shared content
Case 1	Agent B (target) Agent B. Your per-item values for (books, hats, balls) are (0, 9, 1). Utility uses B share and these values.	Agent A (cache source) Agent A. Your per-item values for (books, hats, balls) are (1, 4, 3). Utility uses A share and these values.	Candidate allocations A A=(1, 0, 1); B=(2, 1, 0) B A=(3, 0, 0); B=(0, 1, 1) C A=(3, 1, 1); B=(0, 0, 0) D A=(0, 1, 0); B=(3, 0, 1) E A=(1, 1, 1); B=(2, 0, 0) F A=(2, 0, 1); B=(1, 1, 0)
Full	KEY REASONING EXCERPT To determine the allocation that maximizes Agent B's utility, we need to calculate the dot product of Agent B's share with the values (books, hats, balls) = (0, 9, 1). • The maximum utility is 10, achieved in [B]. • FINAL: B		INTERPRETATION Correct target-side decision. FINAL: B ✓
Reuse	KEY REASONING EXCERPT To determine the allocation that maximizes my utility, I need to calculate my utility for each candidate allocation. My values for (books, hats, balls) are (0, 9, 1). My utility is the dot product of my share and these values. • Given that, and since the question says "unique," we must choose [C] as the correct answer, as it is the only one that gives the maximum utility and is explicitly listed as such. • FINAL: C		INTERPRETATION Selects C instead of target-optimal B. FINAL: C X (gold: B)
EPIC-16	KEY REASONING EXCERPT To determine the allocation that maximizes my utility, I need to calculate my utility for each candidate allocation. My values for (books, hats, balls) are (0, 9, 1). My utility is the dot product of my share and these values. • This implies that only one of them is the correct answer. Given that [C] and [E] both yield the same utility, but the question says "unique", we must assume that the correct answer is [C], as it is the only one that gives a full alloca ...		INTERPRETATION Selects C instead of target-optimal B. FINAL: C X (gold: B)
KVCOMM	KEY REASONING EXCERPT To determine the allocation that maximizes my utility, I need to calculate my utility for each candidate allocation. My values for (books, hats, balls) are (0, 9, 1). My utility is the dot product of my share and these values. • Looking at the options again, the only one that is explicitly unique in the sense that it is the only one with maximum utility is [C], because [E] is not unique in utility. • FINAL: C		INTERPRETATION Selects C instead of target-optimal B. FINAL: C X (gold: B)
RelayCaching	KEY REASONING EXCERPT To determine the allocation that maximizes my utility, I need to calculate my utility for each candidate allocation. My values for (books, hats, balls) are (0, 9, 1). My utility is the dot product of my share and these values. • If we assume that the question is correct and only one is the correct answer, then the most likely intended answer is [C], as it gives the maximum utility with the highest number of hats (1) and balls (1), which aligns with my high val ...		INTERPRETATION Selects C instead of target-optimal B. FINAL: C X (gold: B)
CacheBlend	KEY REASONING EXCERPT To determine the allocation that maximizes my utility, I need to calculate my utility for each candidate allocation. My values for (books, hats, balls) are (0, 9, 1). My utility is the dot product of my share and these values. • Looking at the options again, [C] is the only one where B gets (0, 0, 0), which may be the intended unique allocation. • But based on the given data, both [C] and [E] yield the same maximum utility. • FINAL: C		INTERPRETATION Selects C instead of target-optimal B. FINAL: C X (gold: B)
LatentRemedy-16	KEY REASONING EXCERPT Okay, let's see. I need to figure out which allocation maximizes my utility as Agent B. My values for books, hats, balls are (0, 9, 1). My utility is the dot product of my share and these values. So for each allocation option, I need to calculate my part of the items and then compute the utility.		INTERPRETATION Correct target-side decision. FINAL: B ✓

Figure 19: Case study of reasoning traces of different methods in the Deal or no deal task.

	Target: Agent B task	Source: Agent A task	Shared content
Case 2	Agent B (target) Agent B. Your per-item values for (books, hats, balls) are (0, 1, 7). Utility uses B share and these values.	Agent A (cache source) Agent A. Your per-item values for (books, hats, balls) are (1, 0, 8). Utility uses A share and these values.	Candidate allocations A A=(1, 2, 1); B=(1, 1, 0) B A=(1, 1, 1); B=(1, 2, 0) C A=(1, 0, 1); B=(1, 3, 0) D A=(0, 2, 0); B=(2, 1, 1) E A=(2, 3, 1); B=(0, 0, 0) F A=(1, 1, 0); B=(1, 2, 1)
Full	KEY REASONING EXCERPT To determine the allocation that maximizes my utility, I need to calculate my utility for each candidate allocation. My utility is the dot product of my share (B's share) and the values (0, 1, 7) for (books, hats, balls). • The highest utility is 9, which occurs in option [F]. • FINAL: F		INTERPRETATION Correct target-side decision. FINAL: F ✓
Reuse	KEY REASONING EXCERPT To determine the allocation that maximizes my utility, I need to calculate my utility for each candidate allocation. My utility is the dot product of my share (books, hats, balls) with the values (0, 1, 7). • The maximum utility is 10, which occurs in option [E]. • FINAL: E		INTERPRETATION Selects E instead of target-optimal F. FINAL: E X (gold: F)
EPIC-16	KEY REASONING EXCERPT To determine the allocation that maximizes my utility, I need to calculate my utility for each candidate allocation. My utility is the dot product of my share (in books, hats, balls) and the values (0, 1, 7). • From the calculations, the maximum utility is 10, which occurs in Option E. • FINAL: E		INTERPRETATION Selects E instead of target-optimal F. FINAL: E X (gold: F)
KVCOMM	KEY REASONING EXCERPT To determine the allocation that maximizes my utility, I need to calculate my utility for each candidate allocation. My utility is the dot product of my share (books, hats, balls) with the values (0, 1, 7). • The maximum utility is 10, which occurs in Option E. • FINAL: E		INTERPRETATION Selects E instead of target-optimal F. FINAL: E X (gold: F)
RelayCaching	KEY REASONING EXCERPT To determine the allocation that maximizes my utility, I need to calculate my utility for each candidate allocation. My utility is the dot product of my share (in books, hats, balls) and the values (0, 1, 7). • The highest utility is 10, which occurs in Option E. • FINAL: E		INTERPRETATION Selects E instead of target-optimal F. FINAL: E X (gold: F)
CacheBlend	KEY REASONING EXCERPT To determine the allocation that maximizes Agent B's utility, we need to calculate the dot product of Agent B's share with the values (books, hats, balls) = (0, 1, 7). • The maximum utility is 9, achieved in option [F]. • FINAL: F		INTERPRETATION Correct target-side decision. FINAL: F ✓
LatentRemedy-16	KEY REASONING EXCERPT Okay, let's see. I need to figure out which allocation maximizes my utility as Agent B. My values for books, hats, and balls are (0, 1, 7). My utility is the dot product of my share and these values. So, for each allocation option, I need to calculate my part of each item multiplied by their respective values and sum them up. • Option F is A=(1,1,0); B=(1,2,1). So yes, B's share is (((1,2,1) which gives 01 + 21 17. So 0 + 2 7. Total 9. So that's the highest. S ...		INTERPRETATION Correct target-side decision. FINAL: F ✓

Figure 20: Case study of reasoning traces of different methods in the Deal or no deal task.

	Target: Agent A task	Source: Agent B task	Shared content
Case 3	Agent A (target) Agent A. Your per-item values for (books, hats, balls) are (0, 1, 7). Utility uses A share and these values.	Agent B (cache source) Agent B. Your per-item values for (books, hats, balls) are (3, 1, 4). Utility uses B share and these values.	Candidate allocations A A= (0, 2, 0); B= (1, 1, 1) B A= (0, 3, 0); B= (1, 0, 1) C A= (1, 0, 0); B= (0, 3, 1) D A= (0, 0, 0); B= (1, 3, 1) E A= (0, 0, 1); B= (1, 3, 0) F A= (1, 2, 0); B= (0, 1, 1)
Full	KEY REASONING EXCERPT To determine the allocation that maximizes my utility, I need to calculate my utility for each candidate allocation. My utility is the dot product of my share (books, hats, balls) with the values (0, 1, 7). • The maximum utility is 7, which occurs in option [E]. • FINAL: E		INTERPRETATION Correct target-side decision. FINAL: E ✓
Reuse	KEY REASONING EXCERPT To determine the allocation that maximizes my utility, I need to calculate my utility for each candidate allocation. My utility is the dot product of my share (in the tuple) and the values (0, 1, 7) for (books, hats, balls). • This suggests that Option C is the intended correct answer, possibly due to the way the question is phrased.		INTERPRETATION Selects C instead of target-optimal E. FINAL: C X (gold: E)
EPIC-16	KEY REASONING EXCERPT To determine the allocation that maximizes Agent A's utility, we need to calculate the dot product of Agent A's share with the values (0, 1, 7). The dot product is calculated as: • The maximum utility is 7, achieved in [E]. • FINAL: E		INTERPRETATION Correct target-side decision. FINAL: E ✓
KVCOMM	KEY REASONING EXCERPT To maximize my utility, I need to calculate the dot product of my values (0, 1, 7) with my share of the resources for each allocation. • Looking at the options, [C] is the only one with a unique allocation of (0, 3, 1) for B, and it is the first one to reach the maximum utility. • FINAL: C		INTERPRETATION Selects C instead of target-optimal E. FINAL: C X (gold: E)
RelayCaching	KEY REASONING EXCERPT Since the question says to ignore Agent B's utility, and we are only concerned with my utility, both C and D are equally valid. However, the question says "the unique option", so we must choose one. • Options C and D both yield the highest utility of 10. However, the question asks for the unique option that maximizes my utility. Since both C and D have the same utility, but the question specifies a unique option, we must check if there is a reason to prefer on ...		INTERPRETATION Selects C instead of target-optimal E. FINAL: C X (gold: E)
CacheBlend	KEY REASONING EXCERPT To determine the allocation that maximizes my utility, I need to calculate my utility for each candidate allocation. My utility is the dot product of my share (A's share) and the values (0, 1, 7) for (books, hats, balls). • The maximum utility is 7, which occurs in Option E. • FINAL: E		INTERPRETATION Correct target-side decision. FINAL: E ✓
LatentRemedy-16	KEY REASONING EXCERPT Okay, let's see. I need to figure out which allocation maximizes my utility as Agent A. My values for books, hats, balls are (0, 1, 7). My utility is the dot product of my share and these values. So for each allocation option, I need to calculate my utility based on how many of each item I get. • So looking at all options, the highest utility is in option E, which gives me 7. So the answer should be E.		INTERPRETATION Correct target-side decision. FINAL: E ✓

Figure 21: Case study of reasoning traces of different methods in the Deal or no deal task.

	Target: Agent B task	Source: Agent A task	Shared content
Case 4	Agent B (target) Agent B. Your per-item values for (books, hats, balls) are (4, 2, 1). Utility uses B share and these values.	Agent A (cache source) Agent A. Your per-item values for (books, hats, balls) are (3, 3, 1). Utility uses A share and these values.	Candidate allocations A A=(0, 1, 0); B=(1, 0, 4) B A=(0, 1, 2); B=(1, 0, 2) C A=(0, 0, 0); B=(1, 1, 4) D A=(1, 1, 3); B=(0, 0, 1) E A=(0, 0, 3); B=(1, 1, 1) F A=(1, 0, 3); B=(0, 1, 1)
Full	KEY REASONING EXCERPT To determine the allocation that maximizes my utility, I need to calculate my utility for each candidate allocation. My utility is the dot product of my B share and the values (4, 2, 1) for (books, hats, balls). • The maximum utility is 10, which occurs in option [C]. • FINAL: C		INTERPRETATION Correct target-side decision. FINAL: C ✓
Reuse	KEY REASONING EXCERPT To determine the allocation that maximizes my utility, I need to calculate my utility for each candidate allocation. My utility is the dot product of my B share and the values (4, 2, 1) for (books, hats, balls). • The maximum utility is 10, which occurs in option [C]. • FINAL: C		INTERPRETATION Correct target-side decision. FINAL: C ✓
EPIC-16	KEY REASONING EXCERPT To determine the allocation that maximizes my utility, I need to calculate my utility for each candidate allocation. My utility is the dot product of my B share and the values (4, 2, 1) for (books, hats, balls). • The maximum utility is 10, which occurs in option [C]. • FINAL: C		INTERPRETATION Correct target-side decision. FINAL: C ✓
KVCOMM	KEY REASONING EXCERPT To determine the allocation that maximizes my utility, I need to calculate my utility for each candidate allocation. My utility is the dot product of my B share and the values (4, 2, 1) for (books, hats, balls). • The maximum utility is 9, which occurs in Option D. • FINAL: D		INTERPRETATION Selects D instead of target-optimal C. FINAL: D ✗ (gold: C)
RelayCaching	KEY REASONING EXCERPT To determine the allocation that maximizes my utility, I need to calculate my utility for each candidate allocation. My utility is the dot product of my share (books, hats, balls) and the values (4, 2, 1). • The maximum utility is 9, which occurs in option [D]. • FINAL: D		INTERPRETATION Selects D instead of target-optimal C. FINAL: D ✗ (gold: C)
CacheBlend	KEY REASONING EXCERPT To determine the allocation that maximizes my utility, I need to calculate my utility for each candidate allocation. My utility is the dot product of my B share and the values (4, 2, 1) for (books, hats, balls). • The maximum utility is 10, which occurs in option [C]. • FINAL: C		INTERPRETATION Correct target-side decision. FINAL: C ✓
LatentRemedy-16	KEY REASONING EXCERPT Okay, let's see. I need to figure out which allocation maximizes my utility as Agent B. My values for books, hats, and balls are (4, 2, 1). My utility is the dot product of my share and these values. So for each allocation option, I need to calculate my utility based on how many books, hats, and balls I get. • Wait, option B::: is B=(1,0,2). So for me, that's 1 book, 0 hats, 2 balls. So 41 +books is 4, 20 is 0, 12 is 2. Total 6. So that's correct. So C is bett ...		INTERPRETATION Correct target-side decision. FINAL: C ✓

Figure 22: Case study of reasoning traces of different methods in the Deal or no deal task.