DISENKV: DISENTANGLING SHARED AND QUERY-SPECIFIC EVIDENCE FOR KV CACHE RETENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing observation-based key-value (KV) cache eviction methods mainly optimize importance scoring, evidence aggregation, or budget allocation while largely taking the observation evidence itself as given. We show that recent queries contain strong shared directional structure rather than independent retention signals. This structure can repeatedly reinforce the same historical tokens and obscure evidence expressed by fewer observations, inducing common-demand bias. We introduce DisenKV, a training-free method built on pre-aggregation query-side evidence disentanglement. Query-specific residuals supply the primary eviction-risk evidence, while raw queries retain shared demand for contextual coverage and late complementary rescue. Leave-one-out projection implements this principle by removing each query's component along a shared direction estimated from the remaining observations. The raw and residual views are aggregated separately before a one-sided correction supplements residual risk. Across 16 LongBench tasks and four model families from 3.8B to 24B, DisenKV improves the strongest reproduced non-layer-aware baseline on every model at 20% KV retention and closes 32.5% of the remaining average gap to Full Cache. Under an identical global layer-wise allocation rule and total cache budget, Layer-DisenKV again improves all four models and closes about 44% of the corresponding gap. Controlled counterfactual interventions show that amplifying shared query structure systematically increases support-associated retention dominance under raw scoring, whereas leave-one-out residualization suppresses this response. These results support disentangling retention evidence before aggregation as a complementary approach to improving importance scoring and budget allocation.

1 INTRODUCTION

Long-context language models reuse historical key-value (KV) states during autoregressive generation, causing KV-cache memory to grow with sequence length. Token eviction reduces this cost by retaining a subset of historical entries (Zhang et al., 2023; Liu et al., 2023; Xiao et al., 2024). Observation-based methods use recent queries to score these entries, with advances in importance estimation, robust aggregation, and adaptive budget allocation (Li et al., 2024; Feng et al., 2025b;a; Qin et al., 2025; Guo et al., 2026). DefensiveKV strengthens evidence aggregation, while AdaKV and CAKE adapt cache allocation. These approaches mainly improve how observation evidence is scored, aggregated, or allocated, while largely taking that evidence as given. This leaves an upstream issue: responses from different observation queries may repeat shared demand instead of supplying distinct evidence for retaining a historical token.

We examine this dependence at the query level. Across the evaluated tasks and model families, approximately 68–84% of query energy aligns with a direction estimated from the remaining queries (Figure 1). Observation queries thus contain substantial shared directional structure and cannot simply be treated as independent retention signals. When the shared component aligns with the same historical keys across observations, it repeatedly reinforces their retention scores, reducing the relative influence of evidence expressed by fewer queries. We call this failure mode *common-demand bias*. It originates in the evidence entering the scorer: shared structure affects both mean aggregation and the response magnitudes entering max- or upper-tail-based risk estimates. Controlled interventions test this scoring effect in Appendix E.6; observational and cross-layer analyses provide complementary evidence in Appendix E.5.

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

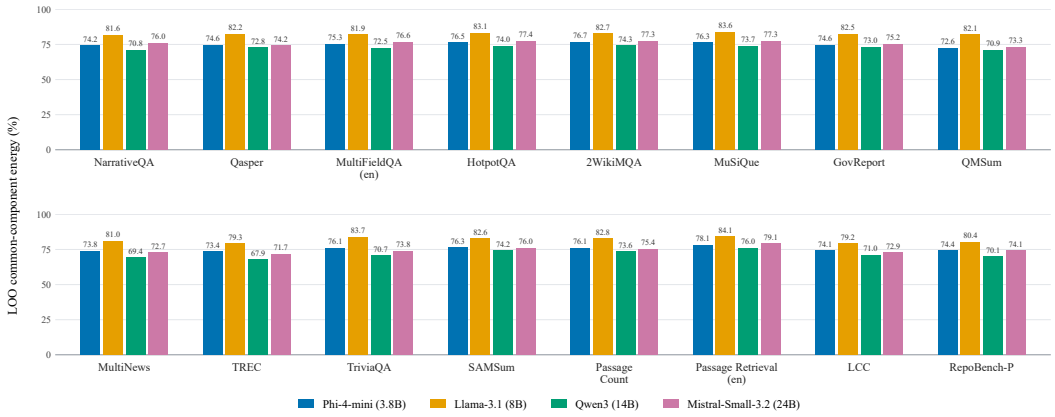


Figure 1: Shared directional structure in observation queries. Across 16 LongBench tasks and four model families, approximately 68–84% of query energy aligns with a leave-one-out direction estimated from the remaining observation queries. For each observation query, we quantify alignment by $\eta_w = (q_w^\top u_{-w})^2 / \max(\|q_w\|_2^2, \epsilon)$, where u_{-w} is estimated from the other $W - 1$ queries as in Section 3.1. Reported values average η_w over the evaluated observation queries for each task–model pair; this statistic measures geometric alignment rather than semantic relevance.

This diagnosis motivates a design principle: *disentangle retention evidence before aggregating it*. Shared demand carries useful context, but its repeated expression should not dominate eviction-risk estimation. We therefore assign query-specific evidence and raw evidence containing shared demand different roles. **DisenKV** implements this principle as a training-free KV-cache eviction method. For each observation query, the remaining queries define a leave-one-out (LOO) shared direction. Subtracting the query's projection onto this direction yields a residual that supplies the primary risk signal. Raw attention protects broadly attended entries and contributes a one-sided complementary rescue after the two views have been aggregated separately. Projection and LOO estimation provide the implementation; the central design is query-side evidence disentanglement and the asymmetric use of the resulting views.

DisenKV acts at the input to retention scoring: it changes the evidence carried by observation queries while preserving generation queries and cached KV states. This makes query disentanglement compatible with layer-wise cache management. **Layer-DisenKV** combines DisenKV scores with the same global allocation rule and total budget as Layer-DefensiveKV (Appendix E.3). This comparison tests the benefit of query-side evidence construction within a shared allocation framework.

Across 16 LongBench tasks (Bai et al., 2024) and four model families from 3.8B to 24B, DisenKV improves over DefensiveKV on every model at 20% retention. Layer-DisenKV also improves all four models, raising the four-model macro-average from 47.82 to 49.07. On Llama-3.1-8B, the advantage persists at 10% retention and across observation windows of 16, 32, and 64 queries. Matched ablations distinguish the benefit of residual direction from norm scaling and quantify the contributions of coverage and late correction. RULER (Hsieh et al., 2024) exposes contrasting behavior on retrieval and distributed frequency aggregation, and efficiency measurements assess the cost of the additional scoring pass.

Our contributions are threefold:

- We expose dependence in the observation evidence used for KV eviction. Across tasks and model families, a substantial fraction of query energy aligns with directions shared by neighboring observations. This shared structure shows that repeated attention support can reflect common contextual demand rather than distinct retention evidence, motivating query-side evidence construction before aggregation.
- We identify common-demand bias and test its scoring mechanism. Controlled counterfactual interventions show that increasing the relative strength of the shared query component monotonically increases support-associated risk dominance under raw scoring. LOO resid-

ualization reduces the response slope on all 16 LongBench tasks and reverses its aggregate direction, linking shared query structure to the behavior of retention scores.

- We introduce pre-aggregation query-side evidence disentanglement through DisenKV. This training-free method uses query-specific residuals for eviction-risk estimation and raw evidence containing shared demand for contextual coverage and complementary rescue. At 20% KV retention, DisenKV recovers 32.5% of the gap to Full Cache relative to the strongest reproduced non-layer-aware baseline, while Layer-DisenKV recovers about 44% under matched layer-wise allocation and improves all four model families. This advantage also extends to 32K long-context retrieval, where DisenKV raises NIAH-MK2 on RULER from 86.80 to 99.50.

2 Related Work

2.1 KV Cache Compression

Quantization reduces the precision of retained states, as in KIVI and KVQuant (Liu et al., 2024b; Hooper et al., 2024); MiniCache instead exploits redundancy across adjacent layers (Liu et al., 2024a). Token eviction changes which historical states remain available. H_2O and Scissorhands exploit persistent importance (Zhang et al., 2023; Liu et al., 2023), while StreamingLLM retains attention sinks and recent tokens (Xiao et al., 2024). PyramidInfer and PyramidKV adapt retention to layer-dependent attention patterns (Yang et al., 2024; Cai et al., 2024). Recent methods also address information lost by eviction: SmallKV uses small-model assistance to compensate compressed attention, while IndexMem combines learned importance prediction with a latent memory for evicted content (Zhao et al., 2025; Yang et al., 2026). DisenKV addresses the selection stage by constructing query-side evidence that distinguishes eviction risk from broad contextual coverage.

2.2 Observation- and Query-Aware Eviction

SnapKV scores historical positions from a recent observation window (Li et al., 2024); AdaKV and CAKE adapt cache budgets across heads or layers (Feng et al., 2025a; Qin et al., 2025), while CriticalKV and DefensiveKV refine token-importance estimation and aggregation (Feng et al., 2025b; Guo et al., 2026). These observation-based approaches mainly improve scoring, aggregation, or allocation while largely taking the observation evidence as given. DisenKV examines the dependence among the queries supplying that evidence and separates their shared and query-specific demand before aggregation. Query-aware methods such as Quest, FIER, and Lookahead Q-Cache modify retrieval or query construction (Tang et al., 2024; Wang et al., 2025a;b). DisenKV's distinction is the construction of retention evidence within an existing observation window: residual queries supply eviction-risk evidence, while raw queries preserve contextual coverage and complementary support. This addresses common-demand bias at the input to retention scoring and complements downstream aggregation and allocation.

2.3 Projection-Based Residualization

Projection-based methods remove directional components from representations (Bolukbasi et al., 2016; Ravfogel et al., 2020; Belrose et al., 2023). In KV compression, AnDPro projects cached values onto an output-derived anchor for importance scoring (Geng et al., 2025), whereas DisenKV residualizes observation queries along a leave-one-out shared direction before attention scoring. Projection itself is established; here it implements pre-aggregation evidence disentanglement motivated by common-demand bias. A detailed comparison with AnDPro is provided in Appendix D.3.

3 Method

DisenKV follows the principle that retention evidence should be disentangled before it is aggregated. Query-specific residuals supply the primary eviction-risk evidence, while raw queries retain shared demand for contextual coverage and complementary rescue. LOO residualization implements this separation in the observation queries; each view then undergoes independent attention scoring and risk aggregation before the late correction supplements residual risk.

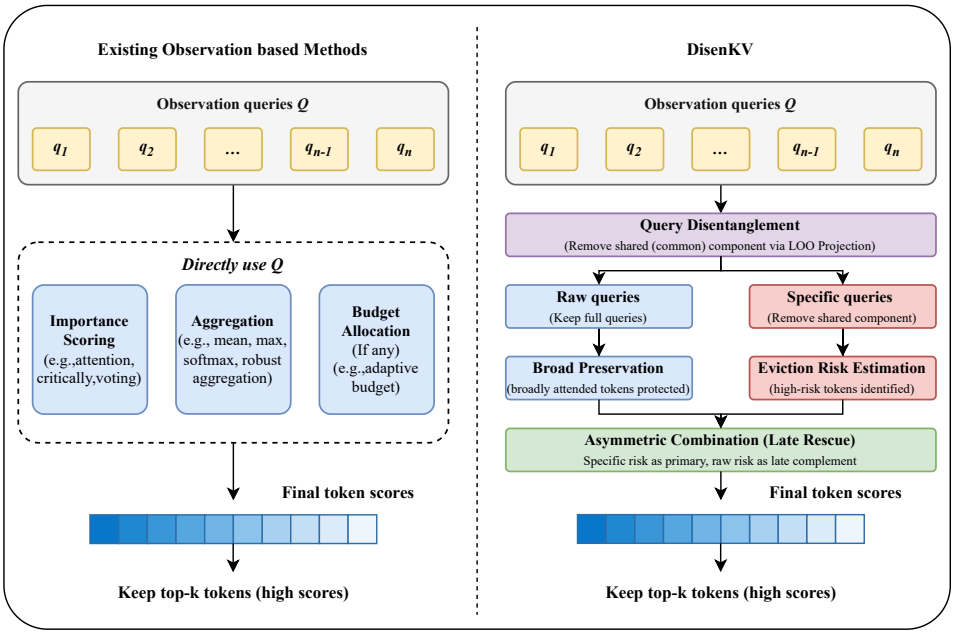


Figure 2: DisenKV overview. LOO residuals supply primary eviction-risk evidence; raw queries supply coverage and late complementary support.

DisenKV scores KV entries once, after context prefill and before decoding (Figure 2). For a context of length N, the last $W > 1$ query states serve as observations for the $T = N - W$ historical positions. The method first constructs LOO residual queries, then aggregates residual and raw attention independently. Eviction-risk scores proxy the cost of removing an entry; higher scores favor retention. Residual risk drives selection, while raw attention supplies coverage and a late correction. These auxiliary scoring queries leave generation queries unchanged. Appendix B gives the numerical scoring and cache-selection details.

3.1 Leave-One-Out Query Disentanglement

Let $Q = \{q_w\}_{w=1}^{W}$, with $q_w \in \mathbb{R}^d$, denote observation queries after the model's query projection and any native query normalization, but before RoPE. Figure 4 in Appendix E.1 illustrates the construction: the other queries estimate a shared direction for each q_w, and subtracting its projection yields a residual in the same query space while leaving the cached keys and values intact. For each query head independently, we construct

$$\bar{q}_{-w} = \frac{1}{W-1}\sum_{j \neq w} q_j, \qquad u_{-w} = \frac{\bar{q}_{-w}}{\max(\|\bar{q}_{-w}\|_2, \epsilon)},$$
$$q_w^{\text{spec}} = q_w - (q_w^\top u_{-w})u_{-w}.$$

Excluding q_w prevents its direct inclusion in its own shared-direction estimate. Under an illustrative additive model with independent, zero-mean query deviations, the LOO mean also removes the conditional self-inclusion term present in the full-window mean (Appendix C.1). When $\|\bar{q}_{-w}\|_2 \geq \epsilon$, the residual is orthogonal to that direction and its norm contracts as

$$\|q_w^{\text{spec}}\|_2^2 = \|q_w\|_2^2 - (q_w^\top u_{-w})^2 \leq \|q_w\|_2^2.$$

Projection changes both query direction and residual-logit scale. The norm-matched control in Section 4.5 isolates the contribution of direction at matched query norms. Small-norm handling and additional geometric properties are given in Appendices B.1 and C.2.

3.2 Asymmetric Evidence Estimation

Both views attend to the same cached keys, with different primary roles illustrated in Figure 5 (Appendix E.2). Raw queries provide broad-context coverage, whereas query-specific residuals provide the primary eviction-risk evidence. Appendix E.4 further visualizes how residualization changes observation attention on real LongBench examples. The raw view additionally supplies the late risk correction described in Section 3.3. Let R_w be the model's positional rotation at the observation's position, $\hat{k}_i$ a cached key after its native positional transformation, s the native attention scale, and M the causal mask. Suppressing batch, layer, and head indices, we compute

$$A^x_{\mathrm{full},w,i} = \mathrm{Softmax}_i\Big(s(R_w q^x_w)^\top \hat{k}_i + M_{w,i}\Big), \qquad x \in \{\mathrm{raw}, \mathrm{spec}\},$$

where $q^{\mathrm{raw}}_w = q_w$. Softmax includes all causally visible context positions; only afterward do we restrict to the historical positions to obtain A^x. For $q^{\mathrm{com}}_w = q_w - q^{\mathrm{spec}}_w$, linearity of RoPE gives the following decomposition of the unmasked logits:

$$\ell^{\mathrm{raw}}_{w,i} = \ell^{\mathrm{spec}}_{w,i} + s(R_w q^{\mathrm{com}}_w)^\top \hat{k}_i.$$

The residual view therefore removes the estimated shared component's logit contribution before softmax, whereas the raw view retains it. Preserving the raw view lets coverage and late correction retain useful context carried by this shared logit contribution.

Reintroducing the grouped query-head index $g \in \{1, \ldots, G\}$, where $G = H_q/H_{\mathrm{kv}}$ query heads share one KV head, we denote the corresponding historical attention by $A^x_{g,w,i}$. Raw attention supplies the average-support coverage score

$$c_i = \frac{1}{GW} \sum_{g=1}^{G} \sum_{w=1}^{W} A^{\mathrm{raw}}_{g,w,i}.$$

To implement risk scoring, both views use the same denoising, local smoothing, and historical-mass rescaling to produce $\widetilde{A}^x$ (Appendix B.2). We average the strongest m observations and combine grouped heads:

$$\rho^x_{g,i} = \frac{1}{m} \sum_{w \in \mathrm{Top}_m(\widetilde{A}^x_{g,:,i})} \widetilde{A}^x_{g,w,i},$$

$$r^x_i = \sqrt{\left(\max_g \rho^x_{g,i}\right)\left(\frac{1}{G}\sum_g \rho^x_{g,i}\right)}.$$

Here $m = \max\{1, \min\{\lceil (1-\alpha)W \rceil, W\}\}$, and $\alpha \in [0,1]$ controls the upper-tail size ($m = 2$ for $W = 32$, $\alpha = 0.95$). Upper-tail aggregation retains responses concentrated in a few observations; the head maximum and mean balance strong responses with support across heads. Using the same operator in both branches isolates their query inputs. Raw coverage averages all observations, so repeated responses directly accumulate coverage mass. Upper-tail risk depends on the strongest m responses: shared structure affects their magnitudes and ordering even after the number of contributing observations is capped. Residualization acts before this aggregation, reweighting the attention evidence that determines which entries receive high risk scores.

The residual branch adds one auxiliary observation-attention pass with $\mathcal{O}(H_q W T d_h)$ time per layer; constructing the LOO residuals themselves costs $\mathcal{O}(H_q W d_h)$.

3.3 Late Complementary Fusion and Cache Eviction

We use r^{spec} as the primary risk and introduce raw risk only after independent aggregation. Within each batch element and KV head, we match the historical-token means and apply a one-sided correction:

$$\tilde{r}^{\mathrm{raw}}_i = \frac{\mathrm{Mean}_j(r^{\mathrm{spec}}_j)}{\max(\mathrm{Mean}_j(r^{\mathrm{raw}}_j), \epsilon)} r^{\mathrm{raw}}_i,$$

$$r^{\mathrm{pre}}_i = r^{\mathrm{spec}}_i + \lambda_{\mathrm{raw}}[\tilde{r}^{\mathrm{raw}}_i - r^{\mathrm{spec}}_i]_+.$$

Table 1: Performance comparison on LongBench under 20% KV cache retention. Layer-DefensiveKV and Layer-DisenKV share the global allocation rule and total cache budget. The best and second-best results among all compressed configurations are highlighted in **bold** and underline, respectively. Full Cache is the uncompressed reference and is excluded from ranking.

	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		
Method	NrtvQA	Qasper	MF-en	Hotpot	2WikiMQA	Musique	GovRep	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	Lcc	RB-P	**Avg.**
Phi-4-mini-instruct-3.8B, 20% Cache Size																	
Full Cache	24.97	45.21	54.12	52.13	36.56	27.49	34.93	24.03	27.04	71.50	83.07	44.57	3.50	95.00	60.88	55.57	46.29
StreamingLLM	14.46	18.43	26.05	31.60	23.77	12.71	27.28	19.10	21.76	42.67	77.16	38.95	1.46	22.00	52.42	50.70	30.03
SnapKV	18.12	25.97	30.70	39.82	30.39	20.62	27.17	19.67	21.99	48.17	82.66	44.13	1.93	72.62	62.50	54.77	37.58
CAKE	17.97	27.95	34.30	38.79	31.56	20.55	27.39	19.72	22.52	50.83	82.87	43.54	2.73	76.75	61.54	53.78	38.30
AdaKV	19.62	26.30	32.04	42.02	30.43	20.81	26.78	19.92	22.29	53.00	82.63	43.87	2.50	74.00	63.39	54.83	38.40
CriticalKV	22.72	26.52	31.54	43.24	31.84	22.91	28.50	21.51	22.71	49.00	**83.41**	43.47	**3.50**	83.50	**63.85**	54.50	39.55
DefensiveKV	23.33	34.92	44.98	50.94	35.98	26.89	28.92	21.61	24.00	63.50	82.63	44.06	2.50	93.00	63.22	**55.57**	43.50
DisenKV	22.96	**38.78**	47.42	50.68	36.77	29.38	31.59	22.43	24.43	66.50	82.57	**45.35**	3.00	92.50	63.61	55.27	44.58
Layer-DefensiveKV	**25.55**	38.50	48.30	**52.82**	40.11	25.79	29.84	22.86	24.62	65.83	82.36	43.81	2.54	93.21	63.07	55.08	44.64
Layer-DisenKV	25.39	38.56	**48.88**	50.37	**41.26**	**29.96**	**33.05**	**23.38**	**24.92**	**67.00**	**83.41**	44.61	3.00	**94.00**	62.07	55.15	**45.31**
Llama-3.1-8B-Instruct, 20% Cache Size																	
Full Cache	29.55	44.68	55.82	57.59	48.89	32.61	34.40	25.51	26.83	73.00	92.36	43.27	7.38	99.50	63.44	52.36	49.20
StreamingLLM	22.05	19.83	23.87	39.44	20.97	15.46	27.76	20.63	22.27	53.50	89.97	40.04	4.00	29.50	65.61	**60.66**	34.72
SnapKV	25.64	28.23	29.71	46.17	29.64	22.07	27.09	21.51	22.46	48.50	**92.21**	44.08	5.08	79.50	67.17	54.36	40.21
CAKE	26.29	30.54	33.28	46.03	32.08	24.73	27.77	22.16	22.91	51.50	91.86	43.56	6.50	92.50	65.46	52.50	41.85
AdaKV	27.07	28.69	32.85	49.64	30.89	21.57	26.70	21.85	22.67	55.50	91.30	43.89	7.30	80.50	66.44	55.43	41.39
CriticalKV	29.81	32.58	34.96	52.34	36.24	26.37	28.35	23.52	23.24	56.50	90.80	43.37	**8.89**	93.00	67.05	54.99	43.88
DefensiveKV	29.97	40.46	46.23	52.20	38.40	28.06	29.96	23.89	24.11	68.00	91.58	43.17	8.28	**100.00**	67.17	55.40	46.68
DisenKV	29.86	41.57	51.20	**56.97**	41.50	28.77	31.57	23.90	24.46	68.00	91.94	43.58	7.75	99.50	69.44	59.23	48.08
Layer-DefensiveKV	30.10	42.91	52.94	55.03	**44.07**	27.00	30.99	24.95	24.42	69.00	91.30	43.54	8.38	**100.00**	67.60	56.00	48.01
Layer-DisenKV	**30.79**	**43.37**	**54.90**	56.73	42.96	**28.96**	**33.49**	**25.13**	**25.94**	**71.20**	91.53	**45.21**	8.86	**100.00**	**69.86**	57.93	**49.18**
Qwen3-14B, 20% Cache Size																	
Full Cache	30.90	45.59	53.55	63.53	54.16	40.33	32.70	24.56	24.84	75.50	92.25	43.90	8.50	100.00	70.74	65.00	51.63
StreamingLLM	17.89	22.11	25.12	36.25	26.52	15.21	27.71	19.65	20.45	58.02	83.09	37.68	7.15	28.25	**70.30**	50.85	34.14
SnapKV	21.15	30.74	28.89	48.17	33.61	26.03	28.13	20.07	20.15	55.17	88.89	42.44	6.03	74.38	66.54	60.02	40.65
CAKE	19.20	33.12	31.43	45.11	33.94	24.61	27.74	19.53	20.78	55.17	88.92	41.57	7.37	79.15	64.83	59.47	40.75
AdaKV	21.99	28.15	29.22	49.61	32.95	26.24	27.89	19.95	20.59	58.50	89.25	42.27	6.03	72.09	69.85	59.33	40.87
CriticalKV	25.58	32.42	32.61	54.02	38.83	30.24	29.65	21.53	20.98	61.83	89.60	42.83	7.03	92.30	67.49	59.28	44.14
DefensiveKV	26.84	40.15	45.80	57.35	46.08	35.50	30.17	22.77	21.69	67.54	90.13	42.30	6.36	97.17	67.92	62.39	47.51
DisenKV	**27.62**	41.32	47.13	59.02	**47.42**	36.53	31.05	23.43	22.32	69.50	**92.75**	**43.53**	6.55	**100.00**	69.89	64.20	48.89
Layer-DefensiveKV	27.11	**43.69**	49.39	59.04	47.26	35.20	30.68	**23.78**	21.81	71.54	90.09	41.72	6.86	97.30	65.84	61.03	48.27
Layer-DisenKV	26.32	42.98	**51.34**	**63.08**	46.97	**39.87**	**31.63**	23.75	**23.23**	**74.50**	**92.75**	43.28	**9.15**	**100.00**	70.19	**65.09**	**50.26**
Mistral-Small-3.2-24B, 20% Cache Size																	
Full Cache	34.40	51.33	55.95	68.10	63.09	49.85	34.16	23.48	26.75	76.50	93.14	47.51	12.50	100.00	76.25	74.64	55.48
StreamingLLM	30.53	16.19	18.93	50.57	38.02	28.34	23.95	19.33	18.86	19.00	84.58	42.50	9.97	31.50	41.39	57.61	33.20
SnapKV	34.14	23.50	23.26	55.50	40.31	33.09	23.86	19.75	19.69	43.50	93.11	48.60	11.97	86.00	73.07	70.67	43.75
CAKE	35.31	25.23	27.98	55.89	40.99	31.87	24.22	19.78	20.15	48.50	94.06	48.33	10.97	80.50	73.27	69.91	44.19
AdaKV	36.23	26.10	23.82	57.05	40.02	33.93	23.32	20.27	19.92	47.50	93.57	48.20	11.47	91.50	73.82	70.20	44.81
CriticalKV	**36.37**	29.37	31.58	57.21	44.11	34.54	24.54	21.31	20.65	54.00	93.83	48.58	11.08	93.50	74.19	69.13	46.50
DefensiveKV	33.28	34.48	42.72	60.55	44.30	39.37	27.90	21.99	21.59	61.00	93.81	48.73	10.50	**100.00**	74.55	**72.63**	49.21
DisenKV	32.40	37.72	43.68	60.79	49.75	40.36	28.65	22.26	21.59	70.00	93.56	48.49	11.50	**100.00**	73.87	72.56	50.45
Layer-DefensiveKV	34.14	39.22	42.37	61.68	**49.85**	39.44	**29.08**	**23.68**	**23.03**	63.00	93.31	48.88	10.43	**100.00**	**75.76**	71.91	50.36
Layer-DisenKV	33.94	**39.52**	**48.03**	**62.37**	49.11	**41.79**	28.88	22.39	22.26	**74.50**	**94.13**	**49.07**	**12.50**	**100.00**	73.68	72.52	**51.54**

For $0 \leq \lambda_{\text{raw}} \leq 1$, this increases a token's score only when its matched raw risk is higher, with $r_i^{\text{spec}} \leq r_i^{\text{pre}} \leq \max\{r_i^{\text{spec}}, \tilde{r}_i^{\text{raw}}\}$. Appendix C.3 derives this bound and characterizes exactly when the correction can change the top-K ranking. Mean matching calibrates the two risks before comparing them, so the correction responds to token-level disagreement between the views. With $\lambda_{\text{raw}} = 0.1$, each score increases by one tenth of the positive raw–residual risk gap. Scores remain unchanged where residual risk already equals or exceeds matched raw risk.

After late fusion, historical risks are lower-bounded by their per-KV-head mean and reweighted using value/output magnitude. Raw-query coverage then prioritizes a capped set of historical positions whose cumulative normalized coverage reaches a target mass, while the final W observation positions receive the highest retention priority. The remaining entries are ranked by the resulting scores under the configured retention budget, retaining each selected key and value together. Appendix B.3 gives the exact thresholds, normalization, and budget interaction. We use $\lambda_{\text{raw}} = 0.1$; all other defaults are listed in Appendix B.

Table 2: LongBench with Llama-3.1-8B at 10% KV cache retention. Layer-DefensiveKV and Layer-DisenKV share the global allocation rule and total cache budget. Bold marks the best result among all compressed configurations for each task and macro-average, including ties. Full Cache is the uncompressed reference and is excluded from ranking.

	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		
Method	NrtvQA	Qasper	MF-en	Hotpot	2WikiMQA	MuSiQue	GovRep	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	LCC	RB-P	Avg.
Full Cache	29.55	44.68	55.82	57.59	48.89	32.61	34.40	25.51	26.83	73.00	92.36	43.27	7.38	99.50	63.44	52.36	49.20
SnapKV	23.25	20.77	23.54	43.11	22.29	18.57	24.13	19.51	20.43	43.00	92.01	43.18	7.35	51.50	64.80	54.17	35.73
CAKE	24.25	22.79	24.69	44.49	23.58	20.30	24.70	20.75	20.38	44.50	**92.36**	43.44	5.15	76.00	65.52	52.95	37.87
AdaKV	24.54	22.69	25.47	41.86	22.59	16.74	24.52	20.17	20.81	46.00	92.25	43.36	7.23	50.50	66.25	54.84	36.24
CriticalKV	27.62	25.02	28.78	43.49	23.50	20.67	25.69	21.39	21.10	48.50	92.17	42.84	7.30	63.50	66.61	54.09	38.27
DefensiveKV	27.23	25.32	35.30	46.29	27.15	18.23	25.42	22.39	20.88	54.00	91.96	**44.13**	5.90	98.50	67.46	54.15	41.52
DisenKV	28.04	30.29	39.98	47.75	35.27	23.28	**31.57**	22.37	21.66	55.50	91.30	42.92	5.16	**99.50**	67.96	**59.19**	43.86
Layer-DefensiveKV	27.30	28.53	38.09	49.12	30.80	22.44	25.76	22.42	21.41	52.00	92.01	43.94	**8.68**	98.00	67.66	56.01	42.76
Layer-DisenKV	**29.23**	**35.76**	**43.08**	**51.47**	**38.35**	**24.10**	28.83	**23.36**	**22.20**	**63.50**	91.50	43.08	8.34	**99.50**	**69.92**	58.47	**45.67**

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluate 16 LongBench tasks on Phi-4-mini-instruct-3.8B (Abouelenin et al., 2025), Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen3-14B (Yang et al., 2025), and Mistral-Small-3.2-24B (Mistral AI, 2025), against reproduced StreamingLLM, SnapKV, CAKE, AdaKV, CriticalKV, and DefensiveKV baselines, with Full Cache as reference. Llama-3.1-8B is additionally evaluated on 13 RULER tasks at 32K context length (Hsieh et al., 2024). Dataset statistics and evaluation details are provided in Appendix A.

The default retention fraction is 20% ($\chi = 0.8$), with 10% retention ($\chi = 0.9$) used for tighter-budget evaluation. DisenKV uses $W = 32$, $k = 5$, $\alpha = 0.95$, and $\lambda_{\text{raw}} = 0.1$; $W \in \{16, 32, 64\}$ is evaluated for window sensitivity. Layer-DisenKV and Layer-DefensiveKV use the same global allocation rule and total cache budget. Macro-averages weight all 16 LongBench tasks equally. Implementation, window-analysis, and ablation details are given in Appendix B, Section 4.6, and Appendix D, respectively.

4.2 LONGBENCH PERFORMANCE

DisenKV improves all four model families at 20% KV retention (Table 1). Relative to DefensiveKV, the strongest reproduced standard baseline by model-level average, the gains are 1.08, 1.40, 1.38, and 1.24 points on Phi-4-mini, Llama-3.1-8B, Qwen3-14B, and Mistral-Small-3.2-24B, respectively. These gains extend across the evaluated model families and scales.

Query disentanglement also improves retention under global layer-wise allocation. With the same allocation rule and total budget as Layer-DefensiveKV, Layer-DisenKV gains 0.67, 1.17, 1.99, and 1.18 points on the four models, raising the macro-average from 47.82 to 49.07. This recovers about 44% of the remaining gap to Full Cache, which averages 50.65. Gains occur on 13/16, 14/16, 12/16, and 9/16 tasks, respectively. The consistent model-level gains show that residual evidence also benefits retention when the allocator can redistribute capacity across layers.

4.3 RETENTION BUDGET

The advantage grows under a tighter cache budget. At 10% retention on Llama-3.1-8B, DisenKV reaches 43.86 versus 41.52 for DefensiveKV, a 2.34-point gain (Table 2). Under the shared global allocation rule, Layer-DisenKV improves over Layer-DefensiveKV from 42.76 to 45.67 ($+2.91$ points). Relative to Full Cache at 49.20, this recovers 45.2% of the remaining gap. Both gains exceed their 20%-retention counterparts, showing the value of query-specific evidence when fewer KV slots can be retained. At 40% retention, standard DisenKV remains ahead of DefensiveKV (49.32 vs. 49.21), while Layer-DisenKV improves over Layer-DefensiveKV from 49.46 to 50.24 ($+0.78$ points) (Appendix A.1). Together, these results locate the largest standard-method gains in the aggressive-compression regime, where competition for retained slots is strongest.

Table 3: All 13 RULER tasks at 32K with Llama-3.1-8B and 20% retention. Avg. is the macro-average; MQ denotes multi-query. Bold marks the best compressed result, including ties; Full Cache is a reference.

			NIAH								QA			
Method	CWE	FWE	MK1	MK2	MK3	MQ	MV	S1	S2	S3	1	2	VT	Avg.
Full Cache	45.22	94.13	99.60	99.60	99.40	98.75	99.10	100.00	100.00	100.00	79.80	54.80	99.24	89.97
CriticalKV	**26.80**	88.80	91.60	29.40	19.40	95.00	93.60	**100.00**	99.60	42.40	40.80	40.20	97.76	66.57
DefensiveKV	22.94	90.00	**99.80**	86.80	97.00	98.65	97.90	**100.00**	**100.00**	97.40	**68.80**	45.80	**98.76**	84.91
DisenKV	20.13	**90.40**	99.60	**99.50**	**98.80**	**98.75**	**98.30**	**100.00**	**100.00**	**99.60**	68.30	**46.90**	97.98	**86.02**

Table 4: Component ablations on LongBench with Llama-3.1-8B, 20% retention, and $W = 32$. Avg. is the unweighted mean over 16 tasks; bold marks the best result per column, including ties.

	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		
Variant	NrtvQA	Qasper	MF-en	Hotpot	2WikiMQA	MuSiQue	GovRep	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	LCC	RB-P	Avg.
Full DisenKV	29.86	41.57	51.20	**56.97**	**41.50**	**28.77**	31.57	**23.90**	**24.46**	68.00	91.94	43.58	**7.75**	99.50	69.44	59.23	**48.08**
Matched raw-only	**30.25**	38.98	49.86	53.27	39.52	28.61	28.70	23.44	23.99	66.00	91.83	43.88	7.17	99.00	68.63	58.86	47.00
Norm-matched raw	29.36	41.48	49.11	51.51	38.25	27.50	30.28	23.36	23.93	67.50	91.94	43.97	6.28	**100.00**	68.49	58.06	46.94
Full-window mean	29.72	41.17	50.30	56.60	40.53	27.19	31.35	23.44	24.43	68.50	91.16	43.41	6.59	99.50	**69.45**	59.39	47.67
No late correction	29.64	40.49	50.82	56.17	40.41	27.81	**31.61**	23.79	24.34	69.00	**92.15**	43.28	6.69	99.50	69.07	**59.41**	47.76
Residual-only	29.84	**41.92**	**51.43**	56.38	40.93	28.24	30.85	23.69	24.25	**69.50**	89.48	**44.01**	6.06	99.00	63.67	53.41	47.04

4.4 RULER AT 32K

At 32K and 20% retention, DisenKV raises the RULER macro-average from 84.91 with DefensiveKV to 86.02 (Table 3), improving seven tasks and tying two. NIAH-MK2 increases from 86.80 to 99.50. CWE remains a boundary case for distributed frequency evidence: its score is 20.13 versus 22.94 for DefensiveKV.

A matched-budget diagnostic links the CWE regression to reduced retention of its distributed target-word evidence (Appendix E.8): gold KV-slot survival decreases from 25.89% to 24.37% (-1.52 points), whereas FWE survival slightly increases from 18.94% to 19.26%. This highlights a boundary of residual scoring: suppressing repeated demand can be harmful when repetition itself carries task-relevant evidence.

4.5 COMPONENT ABLATIONS

Table 4 compares DisenKV with five controls on Llama-3.1-8B at 20% retention and $W = 32$. The controls retain the aggregation, mean floor, value reweighting, recent-window protection, and cache-selection backend, except for the named changes. Appendix D specifies the controls and evaluation protocol; the table reports all 16 task scores.

Query direction contributes beyond norm scaling. Replacing residual queries with raw queries lowers the average from 48.08 to 47.00, while norm-matched raw queries score 46.94. LOO estimation is also beneficial: excluding the current query from the shared-direction estimate improves 47.67 to 48.08 with gains on 12/16 tasks, supporting both directional residualization and removal of self-contributions.

Raw evidence remains complementary. Removing late correction reduces the average to 47.76, and additionally removing coverage lowers it to 47.04 (Appendix C.3). The residual cue also transfers to all five tested baselines: SnapKV, CAKE, DefensiveKV, AdaKV, and CriticalKV gain 0.79, 0.70, 0.45, 0.17, and 0.20 points, respectively (Appendix D.2).

Controlled interventions further validate the proposed mechanism (Appendix E.6). As shared-component strength increases at matched query norms, the high–low support gap grows from 0.50 to 3.26 under raw scoring but decreases from 0.73 to 0.19 after residualization. Residualization reduces the response slope on all 16 tasks, showing that query-side evidence construction directly controls sensitivity to shared demand.

Table 5: Observation-window sensitivity on LongBench with Llama-3.1-8B at 20% KV retention. Parentheses denote the observation-window size W. Bold marks the best score across all evaluated configurations for each task, including ties. Avg. is the unweighted mean over the 16 tasks.

	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		
Method (W)	NrtvQA	Qasper	MF-en	Hotpot	2WikiMQA	MuSiQue	GovRep	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	LCC	RB-P	Avg.
DefensiveKV (16)	29.75	35.43	46.54	55.51	36.41	25.64	29.20	23.68	23.34	66.00	91.69	44.02	7.31	**100.00**	67.00	55.10	46.04
DefensiveKV (32)	29.97	40.46	46.23	52.20	38.40	28.06	29.96	23.89	24.11	68.00	91.58	43.17	8.28	**100.00**	67.17	55.40	46.68
DefensiveKV (64)	28.98	36.09	45.49	56.69	39.31	27.50	30.63	24.12	23.31	68.00	91.13	43.35	6.77	**100.00**	66.72	55.04	46.45
DisenKV (16)	26.85	40.52	49.81	53.53	39.98	28.19	31.22	24.06	23.57	**72.50**	91.27	43.55	6.59	**100.00**	69.15	58.70	47.47
DisenKV (32)	29.86	41.57	51.20	**56.97**	41.50	28.77	31.57	23.90	24.46	68.00	**91.94**	43.58	7.75	99.50	69.44	**59.23**	48.08
DisenKV (64)	30.34	41.76	49.81	54.98	42.43	27.46	31.30	24.59	24.16	71.00	91.14	43.10	6.66	99.00	68.43	59.10	47.83
Layer-DefensiveKV (16)	29.64	41.23	49.21	54.59	41.31	**29.05**	30.19	24.22	24.25	68.50	91.86	44.04	**9.00**	99.50	68.82	55.00	47.53
Layer-DefensiveKV (32)	30.10	42.91	52.94	55.03	**44.07**	27.00	30.99	24.95	24.42	69.00	91.30	43.54	8.38	**100.00**	67.60	56.00	48.01
Layer-DefensiveKV (64)	30.51	**43.40**	51.42	55.36	40.03	26.23	31.60	24.70	24.29	70.00	91.61	43.21	6.92	**100.00**	68.69	56.46	47.78
Layer-DisenKV (16)	30.41	42.80	54.06	52.00	42.33	27.17	33.16	**25.13**	25.27	69.00	90.53	43.37	7.18	**100.00**	**69.96**	57.80	48.14
Layer-DisenKV (32)	**30.79**	43.37	**54.90**	56.73	42.96	28.96	**33.49**	**25.13**	**25.94**	71.20	91.53	**45.21**	8.86	**100.00**	69.86	57.93	**49.18**
Layer-DisenKV (64)	30.09	43.37	53.04	55.31	42.16	26.11	33.39	24.80	24.75	71.00	91.02	44.62	6.59	99.50	69.40	57.86	48.31

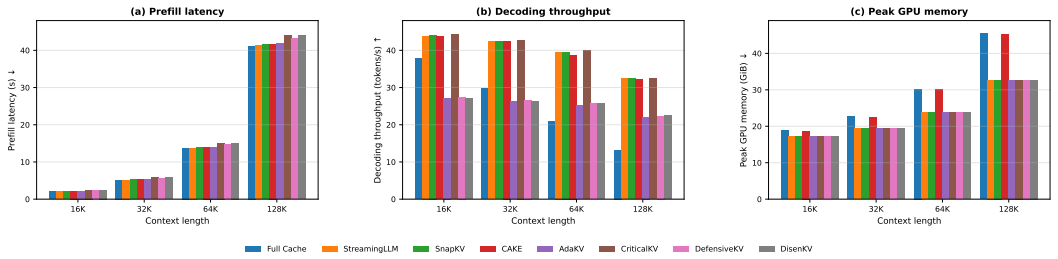


Figure 3: Inference efficiency of Llama-3.1-8B with batch size 1 over 16K–128K contexts: (a) prefill latency, (b) decoding throughput, and (c) peak allocated GPU memory.

4.6 Observation-Window Sensitivity

The benefit persists across observation windows of 16, 32, and 64 queries (Table 5). Standard DisenKV exceeds DefensiveKV by 1.43, 1.40, and 1.38 points, respectively. Layer-DisenKV also improves all three windows under the shared allocator, scoring 48.14, 49.18, and 48.31. Both variants peak at the default $W = 32$; gains across all three windows show robustness to observation-window size.

4.7 Inference Efficiency

Figure 3 evaluates Llama-3.1-8B with batch size 1 over 16K–128K contexts. Relative to DefensiveKV, DisenKV adds only 1.8–3.5% prefill latency while preserving comparable decoding throughput and peak GPU memory. At 128K, it achieves 22.42 tokens/s and 32.73 GiB peak memory, corresponding to a 1.7× decoding speedup and 28.1% lower peak memory than Full Cache. The extra attention pass is used only during prefill-time compression and does not alter decoding.

5 Conclusion

We show that observation-based KV eviction depends not only on how retention evidence is aggregated or allocated, but also on how it is constructed. We identify shared directional structure in observation queries and the resulting common-demand bias, and propose DisenKV to disentangle query-specific retention evidence from shared contextual demand before aggregation. Across four model families, DisenKV consistently improves LongBench performance at 20% retention and remains effective under matched global layer-wise allocation. These results motivate pre-aggregation evidence disentanglement as a complementary direction for KV-cache retention.

AI USE STATEMENT

Generative AI tools were used only to assist with manuscript revision and were not used to generate experimental results or measurements. The authors retain responsibility for the methods, reported measurements, and final manuscript.

REFERENCES

Mohamed Abouelenin, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, Dong Chen, et al. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*, 2025.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 3119–3137, 2024.

Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, and Stella Biderman. Leace: Perfect linear concept erasure in closed form. *Advances in Neural Information Processing Systems*, 36:66044–66063, 2023.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *Advances in neural information processing systems*, 29, 2016.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, et al. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. 38, Main Conference:113152–113188, 2025a. doi: 10.52202/085713-3773. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/a40ff56daab9f4808b1e18350c8a11ce-Paper-Conference.pdf.

Yuan Feng, Junlin Lv, Haoyu Guo, Yukun Cao, S Kevin Zhou, and Xike Xie. Criticalkv: Optimizing kv cache eviction from an output perturbation perspective. *arXiv preprint arXiv:2502.03805*, 2025b.

Zijie Geng, Jie Wang, Ziqi Liu, Feng Ju, Yiming Li, Xing Li, Mingxuan Yuan, Jianye Hao, Defu Lian, Enhong Chen, and Feng Wu. Accurate KV cache eviction via anchor direction projection for efficient LLM inference. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Haoyu Guo, Junlin Lv, S Kevin Zhou, Xike Xie, et al. Defensivekv: Taming the fragility of kv cache eviction in llm inference. In *International Conference on Learning Representations*, volume 2026, pp. 43407–43432, 2026.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun S Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303, 2024.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. Minicache: Kv cache compression in depth dimension for large language models. *Advances in Neural Information Processing Systems*, 37:139997–140031, 2024a.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36:52342–52364, 2023.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024b.

Mistral AI. Mistral-small-3.2-24b-instruct-2506. `https://huggingface.co/mistralai/Mistral-Small-3.2-24B-Instruct-2506`, 2025. Open-weight model, released June 2025.

Ziran Qin, Yuchen Cao, Mingbao Lin, Wen Hu, Shixuan Fan, Ke Cheng, Weiyao Lin, and Jianguo Li. Cake: Cascading and adaptive kv cache eviction with layer preferences. In *International Conference on Learning Representations*, volume 2025, pp. 89945–89977, 2025.

Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. Null it out: Guarding protected attributes by iterative nullspace projection. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 7237–7256, 2020.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-aware sparsity for efficient long-context LLM inference. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 47901–47911. PMLR, 21–27 Jul 2024.

Dongwei Wang, Zijie Liu, Song Wang, Yuxin Ren, Jianing Deng, Jingtong Hu, Tianlong Chen, and Huanrui Yang. Fier: Fine-grained and efficient kv cache retrieval for long-context llm inference. Association for Computational Linguistics, 2025a.

Yixuan Wang, Shiyu Ji, Yijun Liu, Yuzhuang Xu, Yang Xu, Qingfu Zhu, and Wanxiang Che. Lookahead q-cache: Achieving more consistent kv cache eviction via pseudo query. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 34146–34162, 2025b.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Dongjie Yang, Xiaodong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. PyramidInfer: Pyramid KV cache compression for high-throughput LLM inference. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3258–3270, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.195.

Xintong Yang, Hao Gu, Binxing Xu, Lujun Li, Bei Liu, Jiacheng Liu, Qiyuan Zhu, Yike Guo, and Sirui Han. Indexmem: Learned kv-cache eviction with latent memory for long-context llm inference. *arXiv preprint arXiv:2605.25475*, 2026.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in neural information processing systems*, 36:34661–34710, 2023.

Yi Zhao, Yajuan Peng, Nguyen Cam-Tu, Zuchao Li, Wang Xiaoliang, hai zhao, and Xiaoming Fu. SmallKV: Small model assisted compensation of KV cache compression for efficient LLM inference. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=0BVrpXMr5Y.

Table 6: Details of the 16 LongBench tasks used in our experiments. Avg. Len denotes the average input length, and Sample Num. denotes the number of evaluated examples.

Task	Task Type	Eval. Metric	Avg. Len	Language	Sample Num.
NarrativeQA	Single-Doc. QA	F1	18,409	EN	200
Qasper	Single-Doc. QA	F1	3,619	EN	200
MultiFieldQA-en	Single-Doc. QA	F1	4,559	EN	150
HotpotQA	Multi-Doc. QA	F1	9,151	EN	200
2WikiMultihopQA	Multi-Doc. QA	F1	4,887	EN	200
MuSiQue	Multi-Doc. QA	F1	11,214	EN	200
GovReport	Summarization	Rouge-L	8,734	EN	200
QMSum	Summarization	Rouge-L	10,614	EN	200
MultiNews	Summarization	Rouge-L	2,113	EN	200
TREC	Few-shot Learning	Accuracy	5,177	EN	200
TriviaQA	Few-shot Learning	F1	8,209	EN	200
SAMSum	Few-shot Learning	Rouge-L	6,258	EN	200
PassageCount	Synthetic	Accuracy	11,141	EN	200
PassageRetrieval-en	Synthetic	Accuracy	9,289	EN	200
LCC	Code	Edit Sim	1,235	Python/C#/Java	500
RepoBench-P	Code	Edit Sim	4,206	Python/Java	500

Table 7: Detailed LongBench results on Llama-3.1-8B-Instruct with 40% KV cache retention. Layer-DefensiveKV and Layer-DisenKV share the global allocation rule and total cache budget. Bold marks the best result among compressed configurations for each task and macro-average, including ties. Full Cache is the uncompressed reference and is excluded from ranking.

	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		
Method	NrtvQA	Qasper	MF-en	Hotpot	2WikiMQA	MuSiQue	GovRep	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	LCC	RB-P	Avg.
Full Cache	29.55	44.68	55.82	57.59	48.89	32.61	34.40	25.51	26.83	73.00	92.36	43.27	7.38	99.50	63.44	52.36	49.20
StreamingLLM	24.32	28.77	28.74	43.75	30.71	18.55	30.26	21.86	24.88	65.50	92.24	41.65	2.92	46.00	66.54	**61.42**	39.26
SnapKV	27.72	36.42	38.31	54.92	40.02	26.95	30.60	23.33	24.31	56.00	92.31	**43.92**	7.62	96.50	65.95	53.77	44.92
CAKE	30.43	37.57	45.51	57.13	40.08	25.95	30.33	23.80	24.96	61.00	91.83	43.46	6.70	**100.00**	65.56	52.66	46.06
AdaKV	28.36	37.58	41.35	54.80	41.47	29.02	30.18	23.72	24.68	63.50	91.73	43.57	7.27	95.00	64.93	54.75	45.74
CriticalKV	30.10	40.14	49.03	55.95	46.22	30.42	31.49	24.34	25.15	67.50	**92.39**	43.20	8.08	99.00	64.68	55.08	47.67
DefensiveKV	30.07	**46.37**	54.90	57.50	45.97	28.85	33.70	24.69	26.20	71.50	91.78	43.69	**9.88**	**100.00**	66.25	55.97	49.21
DisenKV	30.46	45.76	53.85	56.16	46.72	29.84	33.59	24.99	25.13	70.00	91.47	43.34	8.75	99.50	69.11	60.45	49.32
Layer-DefensiveKV	30.94	43.84	55.01	56.36	**49.14**	29.88	34.09	25.71	26.64	**72.00**	91.49	42.96	8.56	99.50	67.30	57.98	49.46
Layer-DisenKV	**31.54**	45.88	**55.67**	**57.82**	48.76	**31.16**	**36.13**	**25.89**	**26.82**	71.50	91.41	43.01	9.40	99.50	**69.71**	59.61	**50.24**

A LongBench Task Details

Table 6 summarizes the 16 LongBench tasks used in our evaluation, including task category, evaluation metric, average input length, language, and number of evaluated samples. The task-specific metrics follow LongBench (Bai et al., 2024). In the main tables, NrtvQA, MF-en, PCount, PR-en, and RB-P abbreviate NarrativeQA, MultiFieldQA-en, PassageCount, PassageRetrieval-en, and RepoBench-P, respectively.

A.1 LongBench Results at 40% KV Retention

Table 7 reports detailed LongBench results on Llama-3.1-8B-Instruct with 40% KV retention. Under this less aggressive compression setting, standard DisenKV improves over DefensiveKV (49.32 vs. 49.21), while Layer-DisenKV improves the matched Layer-DefensiveKV configuration from 49.46 to 50.24, with gains on 12 of 16 tasks. This result further supports the compatibility of query-side disentanglement with layer-wise cache management across retention budgets.

B Implementation Details

This appendix specifies the operations supporting Section 3. Let N be the prefill length, W the observation window, $T = N - W$ the historical length, and $G = H_q/H_{\text{kv}}$ the number of query heads sharing one KV head. Unless stated otherwise, operations are applied independently across batch elements, layers, and KV heads.

The default settings are $W = 32$, pooling kernel $k = 5$, tail parameter $\alpha = 0.95$, noise-floor coefficient $\beta = 0.1$, rescue coefficient $\lambda_{\text{raw}} = 0.1$, and $\epsilon = 10^{-8}$. We assume $N > W > 1$.

Implementation and hardware. All experiments are implemented with PyTorch 2.12.1, Transformers 5.15.1, and CUDA 13.0 using BF16 precision. All benchmark evaluations and efficiency measurements are conducted on a single NVIDIA RTX PRO 6000 GPU under the same software environment. DisenKV is training-free and requires no model-parameter updates.

B.1 Query Construction and Attention

We project the last W hidden states using the model's query projection or the query slice of a fused QKV projection, followed by the model's native query normalization when present. The LOO means and projection subtraction are computed in FP32 before casting the residual queries back to the model query dtype. The residual queries are not separately normalized.

The shared-direction denominator is clamped as in Section 3.1. When the LOO mean is exactly zero, the projection subtraction is zero. For a nonzero mean whose norm is below ϵ, the resulting direction is not unit length; therefore, the exact orthogonality identities in Appendix C.2 do not apply in this clamped regime.

The model's RoPE, including partial RoPE when applicable, is applied independently to the raw and residual queries. Both views attend to the same cached keys and use the model's native attention scale and causal mask. Each softmax is evaluated in FP32 over all causally visible context positions and then cast back to the query dtype. Only afterward are the historical positions selected, without renormalizing away the attention mass assigned to the recent observation window.

For batch size B and H_q query heads, let $\mathcal{W}$ denote the final W observation positions. The average raw-query attention mass assigned to this recent window, used by the coverage safeguard, is

$$b_{\text{win}} = \frac{1}{BH_qW} \sum_{b=1}^{B} \sum_{h=1}^{H_q} \sum_{w=1}^{W} \sum_{j \in \mathcal{W}} A^{\text{raw}}_{\text{full},b,h,w,j}.$$

Causally masked entries contribute zero.

B.2 Risk Aggregation for Both Views

Both views use the same downstream aggregation operator. For $x \in \{\text{raw}, \text{spec}\}$, define the original historical attention mass as

$$a^x_{g,w} = \sum_{i=1}^{T} A^x_{g,w,i}.$$

Weak responses are first filtered relative to their per-row mean. We then apply one-dimensional average pooling over historical positions and rescale the pooled responses to restore the original historical attention mass:

$$\begin{aligned} B^x_{g,w,i} &= \left[A^x_{g,w,i} - \beta \frac{a^x_{g,w}}{T} \right]_+, \\ P^x &= \text{AvgPool}_k(B^x), \\ \widetilde{A}^x_{g,w,i} &= \frac{P^x_{g,w,i}}{\max\left(\sum_j P^x_{g,w,j}, \epsilon \right)} a^x_{g,w}. \end{aligned}$$

Pooling uses stride 1 and zero padding of $\lfloor k/2 \rfloor$. With the default odd kernel size, the historical sequence length is therefore preserved. The rescaling restores the original historical attention mass whenever the pooled mass exceeds the numerical floor and does not create mass from an all-zero response row.

The number of observations included in the upper-tail aggregation is

$$m = \max\{1, \min\{\lceil (1-\alpha)W \rceil, W\}\}.$$

For each historical position and grouped query head, we average the m largest responses across the observation window:

$$\rho^x_{g,i} = \frac{1}{m} \sum_{w \in \mathrm{Top}_m\left(\widetilde{A}^x_{g,:,i}\right)} \widetilde{A}^x_{g,w,i}.$$

We then combine the strongest grouped-head response with the average grouped-head response using their geometric mean:

$$r^x_i = \sqrt{\left(\max_g \rho^x_{g,i}\right)\left(\frac{1}{G}\sum_g \rho^x_{g,i}\right)}.$$

Under the default settings $W = 32$ and $\alpha = 0.95$, we obtain $m = 2$. The maximum preserves a strong grouped-head response, while the mean reduces the influence of isolated head-wise peaks. The resulting nonnegative scores enter the late-fusion stage without a mean floor; the mean floor is applied only after fusion.

B.3 Value Weighting, Coverage Protection, and Cache Selection

After late complementary fusion, the historical risk scores are lower-bounded by their mean within each batch element and KV head:

$$r^{\mathrm{floor}}_{b,h,i} = \max\left\{r^{\mathrm{pre}}_{b,h,i}, \mathrm{Mean}_j\left(r^{\mathrm{pre}}_{b,h,j}\right)\right\}.$$

Output-aware value weighting. Let $W_O^{(h,g)} \in \mathbb{R}^{d_h \times d_{\mathrm{model}}}$ denote the block of the attention output projection corresponding to query head (h, g), where $g \in \{1, \ldots, G\}$ indexes the $G = H_q/H_{\mathrm{kv}}$ query heads sharing KV head h. For a cached value $v_{b,h,i} \in \mathbb{R}^{d_h}$, the implementation computes

$$\nu_{b,h,g,i} = \left\| v^\top_{b,h,i} W_O^{(h,g)} \right\|_1.$$

This is the quantity evaluated by the fused value–output projection norm operator used in the implementation. Because each KV head is shared by G query heads, these magnitudes are averaged within each group:

$$\bar{\nu}_{b,h,i} = \frac{1}{G}\sum_{g=1}^{G} \nu_{b,h,g,i}.$$

For each token position, the resulting magnitudes are normalized across KV heads:

$$z_{b,h,i} = \frac{\bar{\nu}_{b,h,i}}{\max\left(\sum_{h'=1}^{H_{\mathrm{kv}}} \bar{\nu}_{b,h',i}, \epsilon\right)}.$$

The value-weighted risk is then

$$r^{\mathrm{value}}_{b,h,i} = r^{\mathrm{floor}}_{b,h,i} z_{b,h,i}.$$

Thus, the value magnitude is normalized across KV heads for a fixed token position, rather than across historical positions.

Raw-query coverage protection. Coverage protection uses the average raw-query support score $c_{b,h,i}$ defined in Section 3.2. We normalize the score over historical positions:

$$p_{b,h,i} = \frac{c_{b,h,i}}{\max\left(\sum_j c_{b,h,j}, \epsilon\right)}, \qquad \tau = 0.9 - b_{\text{win}}.$$

For each batch element and KV head, historical positions are sorted in decreasing order of $p_{b,h,i}$. The implementation selects the shortest prefix whose cumulative normalized coverage reaches τ, including the position at which the threshold is first reached. The number of coverage-protected historical positions is capped by

$$M_{\text{cov}} = \text{int}\left(N(1-\chi) - W\right),$$

where χ denotes the configured compression ratio. The cap reserves sufficient capacity for the final W observation positions under the fixed cache budget. When the cap is active, the selected prefix need not reach the target coverage mass. If the cumulative threshold is never reached, the coverage-protected set is empty.

Coverage-selected historical positions are assigned the maximal finite score representable by the score dtype. The final W observation positions are subsequently appended with the same maximal finite priority. These safeguards do not create additional cache capacity; all positions remain subject to the configured final top-K cache budget.

Head-adaptive selection for standard DisenKV. For standard DisenKV, the final scores of all KV heads within a layer are flattened into

$$\mathbf{s}_b \in \mathbb{R}^{H_{\text{kv}}N}.$$

For compression ratio χ, the layer retains exactly

$$K = \lfloor N(1-\chi)H_{\text{kv}} \rfloor$$

KV slots. The retained set is

$$\mathcal{S}_b = \text{TopK}\left(\mathbf{s}_b, K\right).$$

Selection is therefore joint over KV heads and token positions rather than assigning an equal budget to each KV head. The retained length of KV head h is

$$K_{b,h} = \sum_{j \in \mathcal{S}_b} \mathbf{1}\left[\text{head}(j) = h\right],$$

with

$$\sum_{h=1}^{H_{\text{kv}}} K_{b,h} = K.$$

Keys and values are gathered using the same selected indices, ensuring that every retained key remains paired with its corresponding value.

The implementation uses `torch.topk` and retains exactly K indices. No additional tie-expansion rule is applied at the selection boundary. Consequently, equal-score boundary ties follow the native `torch.topk` index selection, and stable ordering among tied entries is not assumed.

B.4 LAYER-WISE GLOBAL BUDGET ALLOCATION

Layer-DisenKV and Layer-DefensiveKV use the same global layer-wise allocation rule and the same total cache budget; they differ only in the KV-slot scores supplied to this allocator.

Let L denote the number of Transformer layers. During prefill, each layer ℓ first produces a flattened score vector

$$\mathbf{s}^{(\ell)} \in \mathbb{R}^{H_{\mathrm{kv}}N}.$$

The layer-wise backend stores these score vectors and performs cache compression only after scores from all L layers have been collected. Thus, the current implementation reallocates retained cache capacity across layers after prefill; it is not a cascaded compression scheme for reducing peak prefill memory.

For global compression ratio χ, the total retained-slot budget is

$$K_{\mathrm{total}} = \lfloor N(1-\chi)H_{\mathrm{kv}}L \rfloor .$$

The score vectors from all layers are concatenated:

$$\mathbf{s}^{\mathrm{global}} = \mathrm{Concat}\left(\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(L)}\right) \in \mathbb{R}^{LH_{\mathrm{kv}}N}.$$

The global allocator then selects

$$\mathcal{S}_{\mathrm{global}} = \mathrm{TopK}\left(\mathbf{s}^{\mathrm{global}}, K_{\mathrm{total}}\right).$$

The effective cache budget assigned to layer ℓ is the number of globally selected KV slots originating from that layer:

$$K_\ell = \sum_{j \in \mathcal{S}_{\mathrm{global}}} \mathbf{1}\left[\mathrm{layer}(j) = \ell\right],$$

so that

$$\sum_{\ell=1}^{L} K_\ell = K_{\mathrm{total}}.$$

Each layer then performs a second top-K_ℓ selection over all of its KV heads and token positions:

$$\mathcal{S}_\ell = \mathrm{TopK}\left(\mathbf{s}^{(\ell)}, K_\ell\right).$$

Consequently, both the per-layer budget K_ℓ and the retained lengths of individual KV heads within each layer adapt to the score distribution. In the absence of boundary ties, this second selection recovers the within-layer subset implied by the global ranking while materializing the allocated budget using the layer-local cache layout.

The realized per-layer budgets of Layer-DefensiveKV and Layer-DisenKV need not be identical because the two methods supply different KV-slot scores to the shared allocator. The matched comparison therefore fixes the global allocation mechanism and total cache budget, rather than the resulting per-layer budgets.

Both the global and within-layer selections use `torch.topk` with the exact requested counts. No additional tie-expansion rule is applied; equal-score boundary ties therefore follow the native `torch.topk` index selection, and stable ordering among tied entries is not assumed.

C Statistical and Geometric Properties

This section gives the derivations supporting the design discussion. The statistical results use an illustrative local model; the geometric identities concern the estimated direction, not recovery of latent factors.

C.1 Leave-One-Out Estimation

Consider $q_w = c + s_w$, where $c \in \mathbb{R}^d$ is a fixed window-level component and the deviations $s_1, \ldots, s_W$ are mutually independent with zero mean and finite covariance Σ. The full-window and LOO means are

$$\bar{q} = c + \frac{s_w}{W} + \frac{1}{W}\sum_{j \neq w} s_j, \qquad \bar{q}_{-w} = c + \frac{1}{W-1}\sum_{j \neq w} s_j.$$

Conditioning on s_w yields

$$\mathbb{E}[\bar{q} \mid s_w] = c + \frac{s_w}{W}, \qquad \mathbb{E}[\bar{q}_{-w} \mid s_w] = c.$$

Thus LOO removes the current deviation's direct contribution to the mean used to estimate its shared direction. Both estimators are unconditionally unbiased under this model; the distinction is conditional self-inclusion, not unconditional bias.

Independence also gives

$$\begin{aligned}\mathbb{E}\|\bar{q}_{-w} - c\|_2^2 &= \frac{1}{(W-1)^2}\sum_{j,k \neq w} \mathbb{E}[s_j^\top s_k] \\ &= \frac{\operatorname{tr}(\Sigma)}{W-1}.\end{aligned}$$

For comparison, the full mean has mean-squared error $\operatorname{tr}(\Sigma)/W$. LOO removes direct self-inclusion at the cost of one observation; it is not a lower-variance mean estimator.

Real neighboring queries can be correlated. In that case, $\mathbb{E}[s_j \mid s_w]$ need not vanish, and cross-covariance terms enter the mean-squared error. LOO still excludes q_w algebraically, but does not ensure statistical independence or eliminate all dependence on its distinctive information. Moreover, unbiasedness of the unnormalized mean does not imply unbiasedness of the normalized direction, particularly when $\|c\|_2$ is small.

C.2 Projection Geometry

Fix an observation and write $u = u_{-w}$, $q = q_w$, and

$$q^{\text{spec}} = (I - uu^\top)q.$$

Assume $\|u\|_2 = 1$, as holds when the LOO mean norm is at least ϵ in exact arithmetic.

Since $I - uu^\top$ is the orthogonal projector onto $u^\perp$,

$$u^\top q^{\text{spec}} = 0, \qquad q^{\text{spec}} = \arg\min_{z:\, u^\top z = 0} \|z - q\|_2.$$

Indeed, for any feasible z, $q - z = (u^\top q)u + (q^{\text{spec}} - z)$ is a sum of orthogonal vectors. The distance is minimized at $z = q^{\text{spec}}$, and the same decomposition gives

$$\|q\|_2^2 = (q^\top u)^2 + \|q^{\text{spec}}\|_2^2.$$

This identity motivates measuring the query energy aligned with the estimated direction. It does not assign semantic relevance to that energy.

For any reference unit direction v, let $\theta = \angle(u, v)$. Symmetry of the projector and Cauchy–Schwarz imply

$$|v^\top q^{\mathrm{spec}}| = |((I - uu^\top)v)^\top q| \le \|(I - uu^\top)v\|_2 \|q\|_2 = \|q\|_2 \sin\theta.$$

If v represents an ideal shared direction, the bound relates residual alignment with v to the angular error of u. It does not establish that this angular error is small in real queries. Finite-precision computation can introduce additional numerical error; the non-unit clamped regime is described in Appendix B.1.

C.3 One-Sided Late Score Correction

For one historical slot, write $a = r_i^{\mathrm{spec}}$, $b = \tilde{r}_i^{\mathrm{raw}}$, and

$$f = a + \lambda_{\mathrm{raw}}[b - a]_+, \qquad 0 \le \lambda_{\mathrm{raw}} \le 1.$$

Then

$$f = \begin{cases} a, & b \le a, \\ (1 - \lambda_{\mathrm{raw}})a + \lambda_{\mathrm{raw}} b, & b > a, \end{cases} \qquad a \le f \le \max\{a, b\}.$$

The correction leaves residual risk unchanged when it dominates and otherwise interpolates toward matched raw risk. Unlike a symmetric average of the two aggregated risks, it does not reduce a larger residual score. This comparison is at the aggregated-score level; it is not an algebraic equivalence to averaging queries or attention maps before aggregation.

The bound does not guarantee preservation of rankings or the retained set. Other tokens may receive larger increments and displace a previously retained token under a fixed budget. For example, residual scores $(1.01, 1)$ and matched raw scores $(0.01, 2)$ have the same mean. With $\lambda_{\mathrm{raw}} = 0.1$, the corrected scores are $(1.01, 1.1)$, reversing their order. This is an arithmetic counterexample, not an experimental result. Subsequent reweighting and retention priorities further modify the scores.

When can the correction change score ordering? Define

$$d_i = [b_i - a_i]_+, \qquad f_i = a_i + \lambda_{\mathrm{raw}} d_i,$$

where $a_i = r_i^{\mathrm{spec}}$ and $b_i = \tilde{r}_i^{\mathrm{raw}}$. For any two historical slots i and j,

$$f_i - f_j = (a_i - a_j) + \lambda_{\mathrm{raw}}(d_i - d_j).$$

Suppose $a_i > a_j$, so slot i is ranked above j by residual risk. Their ordering is reversed by the late correction if and only if

$$\lambda_{\mathrm{raw}}(d_j - d_i) > a_i - a_j.$$

Thus, a lower-ranked slot can overtake a higher-ranked one only when it receives a sufficiently larger raw-risk increment to overcome the original residual-risk margin. When $d_j > d_i$, this condition is equivalently

$$\lambda_{\mathrm{raw}} > \frac{a_i - a_j}{d_j - d_i}.$$

A smaller λ_{raw} therefore requires stronger raw–residual disagreement to reverse a residual-risk ordering.

More generally, let $\mathcal{S}_a$ denote the top-K set induced by the residual scores a at this stage. Ignoring exact boundary ties, the top-K set after late correction remains unchanged if and only if

$$a_i - a_j \ge \lambda_{\mathrm{raw}}(d_j - d_i) \qquad \text{for every } i \in \mathcal{S}_a,\ j \notin \mathcal{S}_a.$$

The correction therefore changes the late-stage top-K ordering only when the additional raw-risk increment is large enough to overcome an existing residual-risk margin.

This characterization also exposes the two possible semantic outcomes. The correction can be beneficial when a task-relevant slot is under-ranked by residual risk but receives sufficient complementary raw support to cross a competing slot. Conversely, it can be harmful when the same condition holds for a less relevant slot, which may displace task-relevant evidence under a fixed budget. The score rule itself cannot distinguish these semantic cases; it provides a one-sided opportunity for raw evidence to modify the residual ranking. Subsequent value reweighting and retention safeguards can further alter the final cache selection.

Table 8: Transferability of the auxiliary LOO-residual cue on LongBench with Llama-3.1-8B at 20% KV retention. For each method, the first row is the original baseline and the second row adds the same residual cue. Bold marks the better result within each original/residual pair, including ties. Avg. is recomputed from the displayed 16 task scores; Δ is the difference between the two unrounded means before display rounding.

Method	NrtvQA	Qasper	MF-en	Hotpot	2Wiki	MuSiQue	GovRep	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	LCC	RB-P	Avg.	Δ
SnapKV	25.64	**28.23**	29.71	**46.17**	29.64	22.07	27.09	21.51	**22.46**	48.50	92.21	**44.08**	5.08	79.50	67.17	54.36	40.21	–
SnapKV + Residual	**26.10**	27.57	**29.99**	45.69	**30.55**	**23.83**	**27.18**	**21.96**	21.46	**49.70**	**92.73**	43.94	**6.25**	**86.50**	**67.44**	**55.23**	**41.01**	+0.79
AdaKV	27.07	28.69	**32.85**	49.64	**30.89**	21.57	26.70	**21.85**	**22.67**	**55.50**	**91.30**	43.89	7.30	**80.50**	66.44	55.43	41.39	–
AdaKV + Residual	**27.45**	**29.32**	32.27	**50.19**	30.63	**22.68**	**27.03**	21.42	22.27	55.00	91.25	**44.76**	**8.40**	79.00	**67.11**	**56.17**	**41.56**	+0.17
CAKE	26.29	30.54	33.28	46.03	**32.08**	**24.73**	27.77	22.16	22.91	**51.50**	**91.86**	43.56	6.50	92.50	65.46	52.50	41.85	–
CAKE + Residual	**26.73**	**32.43**	**34.13**	**47.49**	31.17	24.15	**28.13**	**22.35**	**23.43**	**51.50**	91.56	**44.13**	**6.65**	**96.50**	**67.11**	**53.42**	**42.56**	+0.70
CriticalKV	**29.81**	32.58	**34.96**	52.34	**36.24**	26.37	**28.35**	23.52	23.24	**56.50**	90.80	43.37	8.89	93.00	**67.05**	54.99	43.88	–
CriticalKV + Residual	29.11	**33.32**	34.53	**53.54**	35.87	**26.47**	28.18	**23.61**	**23.29**	55.00	**91.75**	**44.00**	**9.11**	**95.50**	66.12	**55.82**	**44.08**	+0.20
DefensiveKV	**29.97**	40.46	46.23	52.20	**38.40**	**28.06**	29.96	23.89	24.11	**68.00**	91.58	43.17	8.28	**100.00**	67.17	55.40	46.68	–
DefensiveKV + Residual	29.59	**41.48**	**48.37**	**55.64**	37.69	27.94	**30.42**	**23.92**	**24.23**	64.00	**92.09**	**43.80**	**9.18**	**100.00**	**68.24**	**57.49**	**47.13**	+0.45

D Ablation Protocol

Tables 4 and 5 report the complete per-task ablation and observation-window results in the main text. All averages weight the 16 LongBench tasks equally. This appendix specifies the ablation controls and evaluation protocol.

D.1 Ablation Definitions and Protocol

The matched raw-only control substitutes q_w for the residual query in the risk branch. The norm-matched control instead uses

$$q_w^{\mathrm{ctrl}} = q_w \frac{\|q_w^{\mathrm{spec}}\|_2}{\max(\|q_w\|_2, \epsilon)}$$

before RoPE, preserving the raw direction while matching the residual norm. Both leave the separate raw branch unchanged.

The full-window control estimates the direction from $W^{-1}\sum_j q_j$, including the query being residualized. The no-late-correction control sets $\lambda_{\mathrm{raw}} = 0$ while retaining raw-coverage priority. Residual-only disables both raw-risk correction and raw-coverage priority while retaining the remaining scoring and cache-selection pipeline. It does not replace the model's generation queries with residual queries.

The ablation runner uses Llama-3.1-8B-Instruct, seed 42, $W = 32$, and compression ratio 0.8 (20% retention). It evaluates 200 examples per task, except MultiFieldQA-en (150), LCC (500), and RepoBench-P (500), for 3,750 examples in total. The evaluator retains the model context limit and task-specific generation defaults.

D.2 Transferability of Query-Specific Residual Evidence

To examine the transferability of LOO-derived query-specific evidence beyond the full DisenKV pipeline, we inject the same residual cue into five existing KV-eviction methods on Llama-3.1-8B at 20% retention. For each method, the original raw-query pathway and downstream cache-selection rule are preserved. The auxiliary branch applies the same LOO decomposition as DisenKV and is combined using a fixed one-sided correction,

$$s_i^{\mathrm{plug}} = s_i^{\mathrm{raw}} + \lambda_{\mathrm{spec}}[\widetilde{s}_i^{\mathrm{spec}} - s_i^{\mathrm{raw}}]_+,$$

where $\widetilde{s}^{\mathrm{spec}}$ is mean-matched to the original score and $\lambda_{\mathrm{spec}} = 0.1$ is fixed across all methods. Thus, residual queries provide complementary evidence rather than replacing the original raw-query scorer.

As shown in Table 8, the same auxiliary LOO-residual cue improves all five baselines without replacing their original raw-query pathways. SnapKV improves from 40.21 to 41.01 (+0.79), CAKE from 41.85 to 42.56 (+0.70), DefensiveKV from 46.68 to 47.13 (+0.45), AdaKV from 41.39 to 41.56 (+0.17), and CriticalKV from 43.88 to 44.08 (+0.20). The gains occur on 12/16, 12/16, 11/16, 9/16, and 10/16 tasks, respectively.

Table 9: Comparison with AnDPro on LongBench using Llama-3.1-8B-Instruct and $W = 32$. Compression ratios $\chi = 0.8$ and $\chi = 0.9$ correspond to 20% and 10% KV retention, respectively. Bold indicates the better result between the two methods under each retention budget.

	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		
Method	NrtvQA	Qasper	MF-en	Hotpot	2WikiMQA	MuSiQue	GovRep	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	LCC	RB-P	Avg.
20% KV Retention																	
AnDPro	27.66	32.35	35.73	50.91	35.76	20.88	29.46	22.29	23.56	60.00	**92.10**	**44.49**	4.42	98.50	**69.83**	55.12	43.94
DisenKV	**29.86**	**41.57**	**51.20**	**56.97**	**41.50**	**28.77**	**31.57**	**23.90**	**24.46**	**68.00**	91.94	43.58	**7.75**	**99.50**	69.44	**59.23**	**48.08**
10% KV Retention																	
AnDPro	25.63	23.84	26.02	41.27	25.74	16.94	26.34	21.21	21.18	52.00	**92.16**	**44.22**	**5.47**	85.00	66.42	55.34	39.30
DisenKV	**28.04**	**30.29**	**39.98**	**47.75**	**35.27**	**23.28**	**31.57**	**22.37**	**21.66**	**55.50**	91.30	42.92	5.16	**99.50**	**67.96**	**59.19**	**43.86**

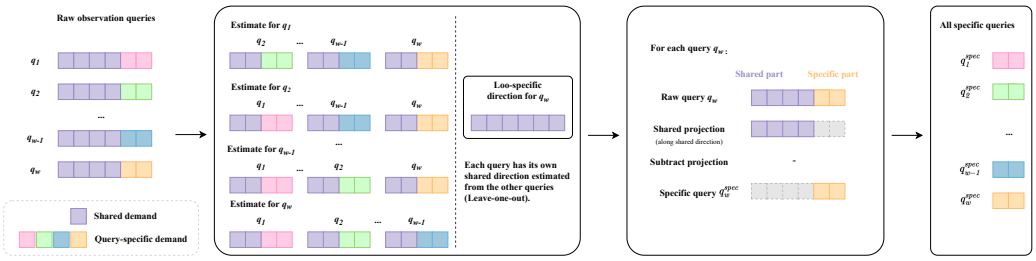


Figure 4: LOO query disentanglement. The other queries estimate a shared direction whose projection is subtracted from q_w. Colors are schematic, not explicit feature dimensions.

LOO-derived residual evidence transfers beyond DisenKV: a single fixed fusion rule improves all five tested observation-based retention methods. The CriticalKV gain is smaller than the gains for SnapKV, CAKE, and DefensiveKV, indicating that the effect of the auxiliary cue varies with the downstream scorer. This result supports treating residual attention as a reusable complementary signal whose integration should respect the original scorer.

D.3 Comparison with AnDPro

AnDPro (Geng et al., 2025) is particularly relevant because it also uses projection in KV-cache retention. However, the two methods apply projection to different representations and for different purposes. AnDPro projects cached values onto an output-derived anchor for importance estimation, whereas DisenKV residualizes observation queries along a leave-one-out shared direction before retention evidence is aggregated. We therefore additionally compare the two methods under the same Llama-3.1-8B-Instruct setting with $W = 32$ at both 20% and 10% KV retention.

DisenKV outperforms AnDPro on 13 of the 16 LongBench tasks under both retention budgets. At 20% retention, the macro-average increases from 43.94 to 48.08 (+4.14 points); at 10% retention, it increases from 39.30 to 43.86 (+4.56 points). The larger gap under the tighter budget suggests that separating query-specific retention evidence from repeated shared demand becomes particularly useful when fewer KV entries can be retained. Together with the conceptual distinction in Section 2, these results show that DisenKV is not simply an alternative projection-based importance metric: projection is used to reconstruct the observation evidence before retention scoring, rather than to score cached values directly.

E Additional Mechanism Analysis

E.1 Query Residualization Illustration

Figure 4 expands the query-construction step of Figure 2. Each query uses a direction estimated from the other queries; colors illustrate the construction rather than identified semantic factors or literal feature coordinates.

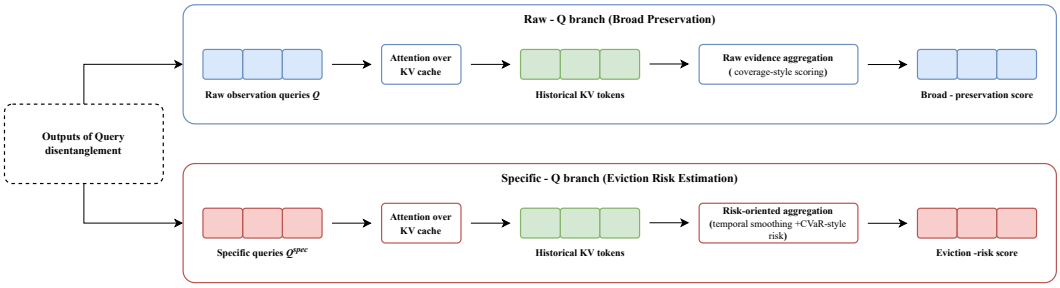


Figure 5: Primary roles of the raw and residual attention branches: broad-context coverage and eviction-risk estimation, respectively. Both score the same historical KV entries. The complementary raw-risk path for late correction and the final safeguards are omitted.

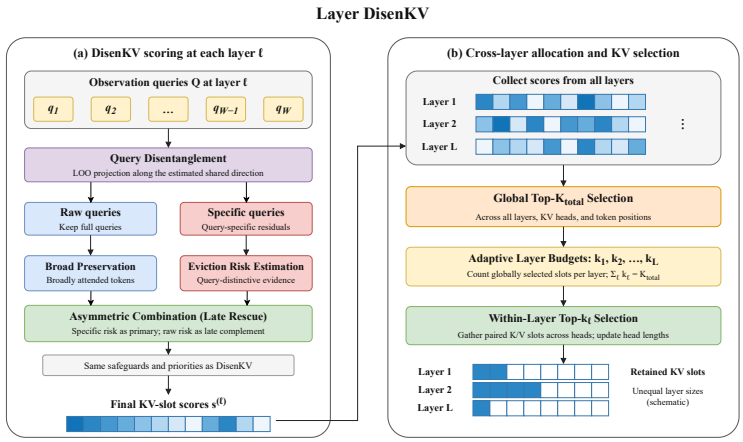


Figure 6: **Integration of DisenKV with layer-wise cache allocation.** DisenKV first produces KV-slot scores independently at each layer using disentangled observation evidence (left). The layer-wise backend then compares scores across layers, derives adaptive per-layer budgets, and selects the retained KV slots within each layer (right). Layer-DisenKV and Layer-DefensiveKV use the same global allocation rule and total cache budget, while their realized per-layer budgets may differ because they provide different KV-slot scores.

E.2 Asymmetric Evidence Illustration

Figure 5 expands the evidence-estimation stage. The upper branch averages raw attention into coverage scores c_i; the lower branch aggregates residual attention into risks r_i^{spec}. Both read the same KV cache; the historical-token blocks indicate the entries being scored, not new KV states produced by attention. The complementary raw-risk path r_i^{raw} used for late correction is omitted from this schematic.

E.3 Integration with Layer-Wise Cache Allocation

Figure 6 illustrates how DisenKV is combined with layer-wise cache management. Each layer first constructs raw and residual observation evidence using the same DisenKV scoring procedure. The resulting KV-slot scores are then collected across layers by the layer-wise allocation backend. A global selection determines the effective cache budget of each layer, after which the selected budget is applied within that layer.

Layer-DisenKV and Layer-DefensiveKV use the same global layer-wise allocation rule and total cache budget; they differ in the KV-slot scores supplied to this allocator. Because the two methods provide different KV-slot scores, their realized per-layer budgets may differ; the matched comparison holds the allocation mechanism, rather than the resulting layer budgets, fixed.

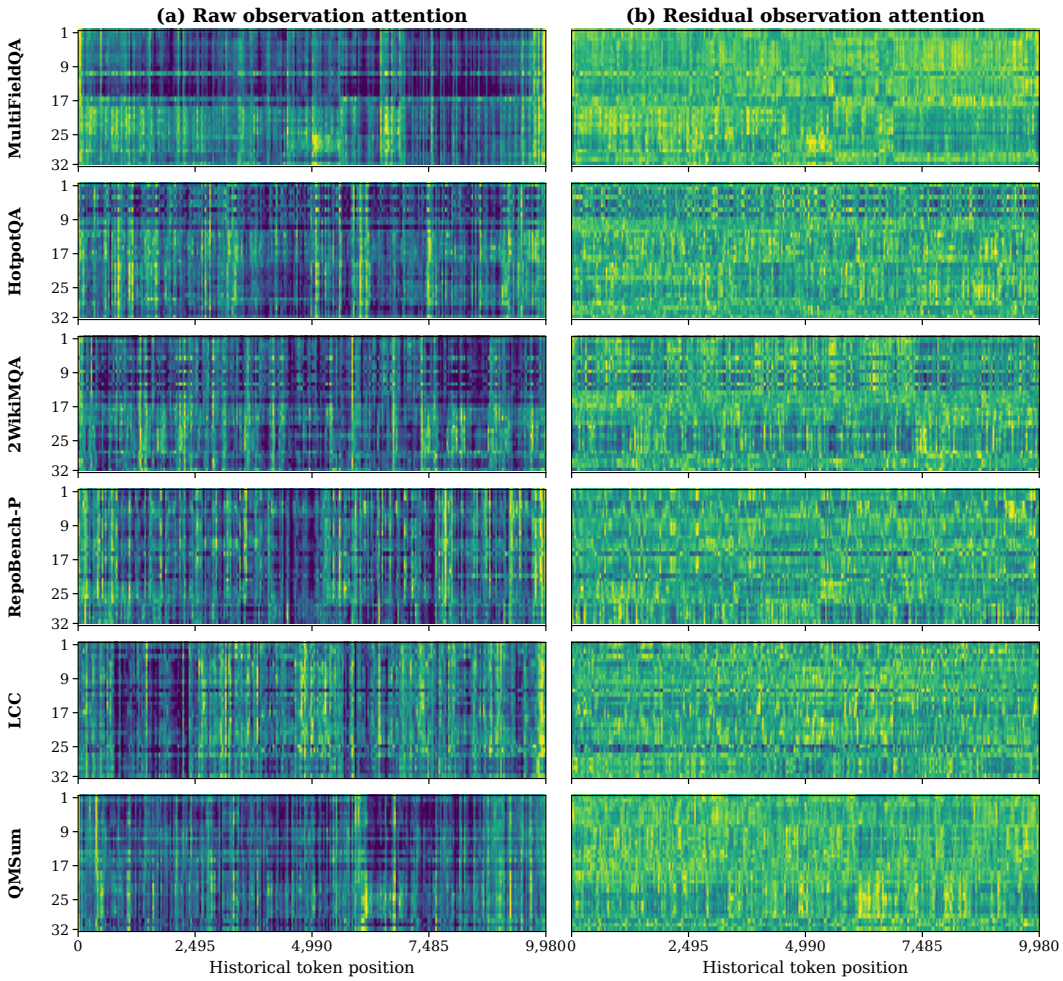


Figure 7: **Raw and residual observation attention across LongBench tasks.** We show MultiFieldQA, HotpotQA, 2WikiMQA, RepoBench-P, LCC, and QMSum using Llama-3.1-8B-Instruct at Layer 20 and KV Head 3. Each row compares attention from the original observation queries (left) with attention after LOO residualization (right), averaged over the grouped query heads sharing the selected KV head. Raw and residual maps within each task use the same logarithmic color scale; scales are selected independently across tasks for visualization. Across these examples, residualization visibly weakens high-contrast repeated attention structure and redistributes observation-derived attention evidence.

E.4 Raw and Residual Observation Attention

To visualize how query disentanglement changes the evidence entering retention scoring, Figure 7 compares raw and residual observation attention on six LongBench tasks. For each task, we use the first evaluated example and fix Llama-3.1-8B-Instruct, Layer 20, KV Head 3, and $W = 32$. The attention maps are averaged over the grouped query heads sharing the selected KV head.

E.5 Observational Support–Risk Coupling

We first quantify the observational relationship between repeated support from raw observation queries and retention-risk dominance, and whether this dependence is reduced after query residualization. For each historical token i, we define its raw-query support count as

$$S_i = \sum_{w=1}^{W} \mathbf{1}\left[i \in \text{Top20\%}\left(A^{\text{raw}}_{w,:}\right)\right].$$

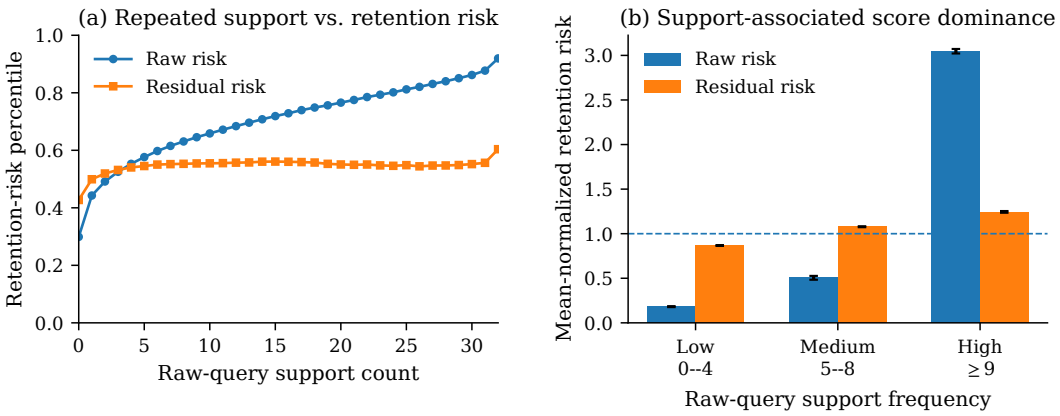


Figure 8: **Repeated raw-query support is associated with retention-risk dominance.** (a) Historical tokens supported by more observation queries receive progressively higher raw-query retention-risk percentiles, whereas the residual branch exhibits substantially weaker dependence on the same raw-derived support count. (b) Grouping historical tokens by raw-query support frequency shows that the high-to-low support gap in mean-normalized risk decreases from 2.87 under raw scoring to 0.38 after LOO query residualization, corresponding to an 86.8% attenuation. Support count denotes the number of the $W = 32$ observation queries for which a historical token lies in the top 20% of raw-attention positions. Statistics are first averaged over the eight KV heads within each sample; error bars denote 95% confidence intervals across the resulting 640 samples.

The same raw-derived support count S_i is used to analyze both the raw and residual risk branches, so the horizontal variable remains fixed between the two comparisons.

To compare relative score dominance across branches, we mean-normalize the historical retention risk within each KV head:

$$\hat{r}_i^x = \frac{r_i^x}{\max\big(\mathrm{Mean}_j(r_j^x), \epsilon\big)}, \qquad x \in \{\mathrm{raw}, \mathrm{spec}\}.$$

We define the support-associated dominance gap as

$$D^x = \mathbb{E}\left[\hat{r}_i^x \mid S_i \geq 9\right] - \mathbb{E}\left[\hat{r}_i^x \mid S_i \leq 4\right].$$

With $W = 32$ and a top-20% support criterion, the nominal expected support under independent uniform selection is $32 \times 0.2 = 6.4$. We therefore partition support counts into low (0–4), medium (5–8), and high (≥ 9) groups.

Figure 8 aggregates 40 examples from each of the 16 LongBench tasks using Llama-3.1-8B-Instruct at Layer 20. Statistics are first averaged over the eight KV heads within each sample and then aggregated across the resulting 640 samples.

Repeated raw-query support is strongly associated with increasing raw retention risk, whereas this dependence is substantially attenuated after LOO residualization. Specifically, the high-to-low normalized-risk gap decreases from 2.87 to 0.38, corresponding to an 86.8% reduction. Because the raw and residual branches use the same downstream risk aggregation operator, the comparison isolates the effect of changing the query-side evidence entering that operator. Appendix E.6 then tests the proposed mechanism directly with a controlled shared-component intervention. These analyses measure the influence of repeated support on scores; token relevance is examined separately in Appendix F.

E.6 Controlled Intervention on Shared Query Demand

The observational analysis in Appendix E.5 shows a strong association between repeated raw-query support and retention-risk dominance. We test the scoring response by varying shared directional evidence in counterfactual observation queries while leaving the cached keys, support groups, and downstream risk aggregation unchanged.

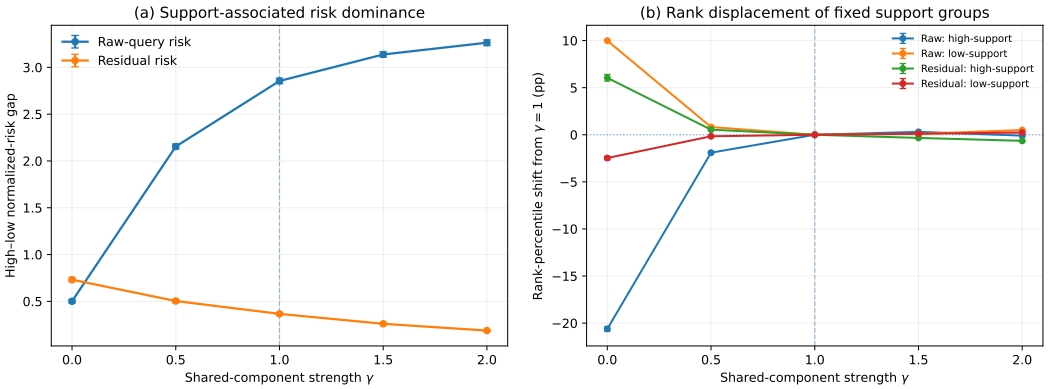


Figure 9: **Controlled intervention on shared observation-query demand.** Counterfactual observation queries are constructed as $q_w(\gamma) = q_w^{\text{spec}} + \gamma q_w^{\text{shared}}$ while preserving the original query norm. Cached keys, downstream risk aggregation, and support groups remain fixed. (a) Increasing shared-component strength monotonically amplifies the high–low support retention-risk gap under raw-query scoring, whereas the corresponding gap decreases after LOO residualization. The dashed vertical line at $\gamma = 1$ marks the unperturbed queries. (b) Mean rank-percentile shifts of fixed high-support ($S_i \geq 9$) and low-support ($S_i \leq 4$) groups relative to $\gamma = 1$. Positive values indicate promotion toward higher retention-risk ranks and negative values indicate demotion. Error bars denote 95% confidence intervals across samples.

For each original observation query, we reuse the LOO decomposition $q_w = q_w^{\text{spec}} + q_w^{\text{shared}}$ and construct

$$q_w(\gamma) = q_w^{\text{spec}} + \gamma q_w^{\text{shared}}, \qquad \gamma \in \{0, 0.5, 1, 1.5, 2\}.$$

The intervention axes are estimated once from the original queries and then held fixed across γ. We additionally rescale each counterfactual query to match the norm of its original query. The intervention therefore varies the relative weight of shared and residual components at a fixed total query norm. The raw-derived support count S_i is also computed once at the unperturbed point $\gamma = 1$ and held fixed for every intervention level. We use the same D^x metric and risk operator as in Appendix E.5, with Llama-3.1-8B-Instruct, Layer 20, $W = 32$, and 40 examples from each of the 16 LongBench tasks.

Figure 9(a) shows the scoring response to the intervention. As γ increases from 0 to 2, the raw support-associated dominance gap increases monotonically from 0.50 to 3.26, whereas the residual gap decreases from 0.73 to 0.19. Linear response slopes over the evaluated intervention levels are $+1.30$ for raw risk and -0.27 for residual risk, yielding a raw–residual response contrast of 1.57. At the unperturbed point $\gamma = 1$, the analysis reproduces the observational pattern: the gap is 2.85 under raw scoring and 0.37 after residualization, corresponding to an 87.2% reduction. The qualitative response is consistent across tasks: the raw slope is positive on all 16 LongBench tasks, and the residual slope is smaller than the raw slope on all 16 tasks.

Panel (b) examines the corresponding ranking changes without defining groups from residual risk. Both high- and low-support groups are fixed using only the untreated raw-query support count. Removing shared demand ($\gamma = 0$) substantially demotes high-support entries under raw scoring and promotes low-support entries, while residual scoring exhibits a markedly weaker reordering. For $\gamma > 1$, rank shifts are comparatively small even though the raw score gap continues to widen, indicating that stronger shared demand primarily increases score dominance once many high-support entries are already ranked near the top.

The intervention separates the scoring responses of the two query views. Increasing the relative shared-component strength amplifies support-associated dominance under raw scoring, while residual scoring reduces the response slope across all tasks and reverses its aggregate direction. The manipulated variables are auxiliary scoring queries, and the measured outcomes are risk scores and ranks. Establishing their downstream accuracy effect requires a separate end-to-end intervention.

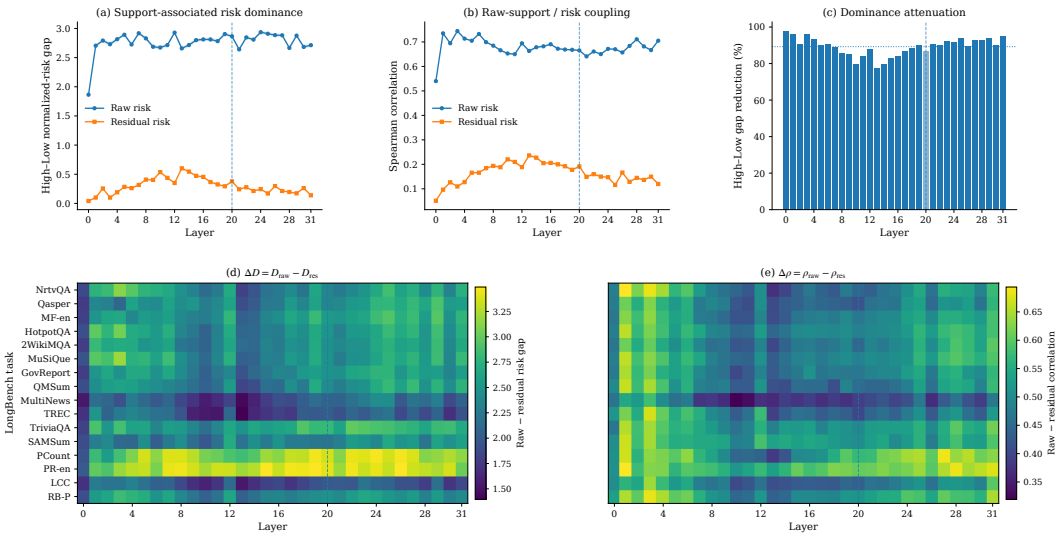


Figure 10: **Cross-layer consistency of support-associated retention-risk dominance.** (a) High–low mean-normalized retention-risk gap across layers for raw and residual query evidence. (b) Spearman correlation between raw-query support count and retention risk across layers. (c) Relative attenuation of the high–low risk gap after LOO query residualization. (d) Task–layer map of $\Delta D = D_{\text{raw}} - D_{\text{res}}$, where positive values indicate stronger support-associated risk dominance under raw scoring. (e) Task–layer map of $\Delta\rho = \rho_{\text{raw}} - \rho_{\text{res}}$, where positive values indicate stronger coupling between raw-query support and raw retention risk. The vertical dashed line marks Layer 20, used in Figure 8; the dotted horizontal line in panel (c) denotes the mean attenuation across layers. Results use Llama-3.1-8B-Instruct with $W = 32$, 40 examples per LongBench task, and all eight KV heads.

E.7 Cross-Layer Consistency of Support–Risk Coupling

The analysis in Appendix E.5 uses Layer 20 to quantify the relationship between repeated raw-query support and retention risk. We further examine whether this behavior is specific to the selected layer. Figure 10 extends the same analysis to all 32 layers of Llama-3.1-8B-Instruct, using 40 examples from each of the 16 LongBench tasks. For each example, statistics are first computed for individual KV heads and then aggregated at the sample level.

Across the network, raw-query scoring remains substantially more sensitive to repeated support than residual scoring. Both the high–low risk-gap analysis and the support–risk correlation show the same qualitative separation across layers, while the task–layer maps demonstrate that this effect is not restricted to the Layer 20 case used in the preceding analysis. Across all 32 layers of Llama-3.1-8B-Instruct, these results support the interpretation that LOO residualization reduces support-associated score dominance rather than correcting an isolated layer-specific behavior.

E.8 Frequency-Evidence Retention on CWE and FWE

The RULER results expose an informative boundary case for query disentanglement. At 32K context length and 20% KV retention, CWE decreases from 22.94 with DefensiveKV to 20.13 with DisenKV, whereas FWE increases slightly from 90.00 to 90.40. We therefore examine whether these different task-level outcomes are accompanied by different retention behavior for the frequency evidence required by the two tasks.

Diagnostic protocol. We analyze all 500 CWE examples and all 500 FWE examples using Llama-3.1-8B-Instruct under the same 20% KV-retention budget as the main RULER evaluation. For each example, we take the gold answer words provided by the benchmark and locate every lexical occurrence of those words in the original context. Character spans are mapped to their BPE positions and then to the corresponding production KV positions using the same tokenizer alignment as in the evaluation pipeline.

Table 10: Frequency-evidence retention on RULER CWE and FWE under a matched 20% KV-retention budget. Target-occurrence survival measures the fraction of KV slots associated with occurrences of the gold answer words that remain in cache. Layer/head survival counts an occurrence as surviving at a layer/head when at least one of its BPE positions is retained.

Task	Method	Target-occ.	Layer/head	Task score
CWE	DefensiveKV	25.89%	27.61%	22.94
	DisenKV	24.37%	26.32%	20.13
FWE	DefensiveKV	18.94%	28.68%	90.00
	DisenKV	19.26%	29.66%	90.40

For a gold target-word occurrence o, let $\mathcal{P}(o)$ denote the set of BPE positions occupied by that occurrence. We define its KV-slot survival rate as

$$s_o = \frac{1}{LH_{\text{kv}}|\mathcal{P}(o)|} \sum_{\ell=1}^{L} \sum_{h=1}^{H_{\text{kv}}} \sum_{p \in \mathcal{P}(o)} \mathbf{1}\left[p \text{ is retained at layer } \ell \text{ and KV head } h\right],$$

where L is the number of Transformer layers and H_{kv} is the number of KV heads.

For each example, we average s_o over all occurrences of its gold target words and then macro-average across the 500 examples. We refer to this quantity as *target-occurrence survival*. We additionally report *layer/head survival*, where an occurrence is considered preserved at a particular layer and KV head if at least one of its BPE positions remains in the cache. Both methods retain exactly 20% of the KV slots overall, so the comparison isolates differences in which slots are retained rather than differences in cache budget.

CWE exhibits reduced preservation of distributed evidence. On CWE, DisenKV reduces target-occurrence survival from 25.89% to 24.37%, a difference of -1.52 percentage points. The secondary layer/head survival metric shows the same direction, decreasing from 27.61% to 26.32%. Thus, under an identical overall cache budget, the remaining CWE score gap coincides with lower preservation of KV slots associated with the distributed occurrences needed for frequency aggregation.

FWE provides a matched contrast. The same retention loss does not occur on FWE. Target-occurrence survival increases slightly from 18.94% with DefensiveKV to 19.26% with DisenKV, corresponding to a paired difference of $+0.32$ percentage points. Layer/head survival similarly increases from 28.68% to 29.66%. This contrasts with CWE and is consistent with the much smaller task-level change on FWE (90.00 to 90.40).

Interpretation. The CWE–FWE contrast identifies where repeated evidence matters for retention. On CWE, the loss of target-occurrence KV slots accompanies a task-score decrease; on FWE, target-occurrence survival increases slightly and the score changes little. This is consistent with the tradeoff addressed by DisenKV: shared demand can dominate retention scores, yet also carry context that the task needs. The raw coverage and late-correction paths preserve part of that evidence; the CWE result shows that this balance remains imperfect for distributed frequency aggregation.

The diagnostic links sample-level retention patterns to task behavior. It does not isolate the causal contribution of individual evictions to prediction errors.

F Support–Relevance Case Study

Figure 11 presents three real token-level cases observed during benchmark inference under raw-only eviction. In each case, broadly supported historical tokens receive higher retention priority, while selected gold-answer-supporting tokens with substantially sparser query support are assigned lower retention scores and may be evicted. These observed cases reveal a mismatch between repeated query support and answer relevance at the actual retention boundary.

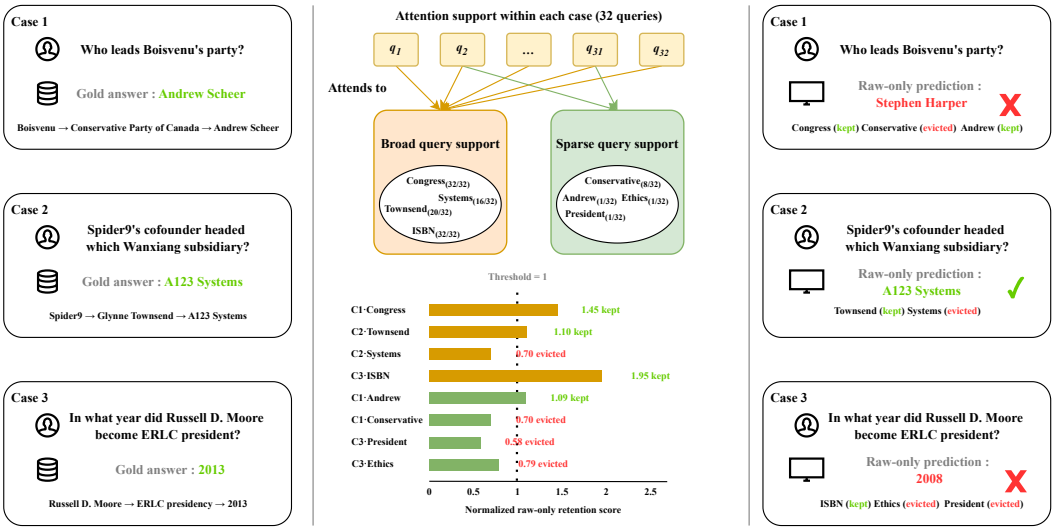


Figure 11: **Real token-level cases of support–relevance mismatch.** Three examples observed during raw-only eviction with Llama-3.1-8B-Instruct at 20% KV retention. Support count measures how many of the $W = 32$ observation queries place a historical token within the top 20% of raw-attention positions, while retention scores are normalized by their corresponding selection cutoffs. Across the three cases, broadly supported tokens can remain above the retention cutoff whereas selected gold-answer-supporting tokens with sparser support receive lower scores and may be evicted. These cases locate support–relevance mismatch at the retention cutoff in actual benchmark inference. A low-scored gold-supporting slot does not necessarily imply an incorrect final answer, since other retained evidence may still be sufficient for generation. Slots use zero-based Layer 1 and KV Head 3; C1–C3 denote the three cases.

G LIMITATIONS

The CWE regression shows that repeated demand can also carry useful task evidence, highlighting a boundary of residualization. In addition, the component and controlled-intervention analyses are conducted on one model, and the interventions establish effects on retention scoring rather than downstream task accuracy. The layer-aware experiments also evaluate a single global allocation rule. These limitations leave broader validation of the proposed mechanism across models, allocation strategies, and downstream causal outcomes to future work.