ClueKV: Self-Guided KV Cache Compression via Adaptive Head Prioritization

Anonymous ACL submission

Abstract

KV cache compression is essential for scaling large language models to long-context inference. However, existing methods excel at explicit retrieval tasks but often degrade on tasks that require multi-step reasoning over dispersed evidence. We present **ClueKV**, a training-free KV cache compression framework that better preserves the reasoning-critical clues under tight cache budgets. The core design of ClueKV is a *self-guided, adaptive headwise token reweighting* mechanism. Specifically, ClueKV first uses contrastive probability shifts between the query-conditioned context and a context-free reference to construct an unsupervised clue signal, which highlights context tokens supportive of downstream reasoning under the current input. This signal then serves as self-generated guidance for compression. Based on this signal, ClueKV dynamically identifies, for each input instance, the attention heads that are more aligned with reasoning-relevant context regions, and reweights token importance head by head before KV eviction. Across LongReason, LongBench, and RULER, ClueKV consistently outperforms strong KV compression baselines, with the largest gains on complex reasoning tasks. On LongReason, it retains 99.21% of full-KV accuracy at a 50% cache budget, and remains more stable than prior methods as context length increases to 64k tokens. ClueKV also introduces only marginal prefill overhead, which making it practical for long-context deployment. [1]

1 Introduction

Large language models (LLMs) have shown strong performance on long-context tasks such as multi-document question answering, long-form summarization, and code understanding (Achiam et al., 2023; Yang et al., 2025). However, serving LLMs on long context remains expensive because the key-value (KV) cache grows linearly with context length, creating a major memory bottleneck during autoregressive decoding (Shah et al., 2024).

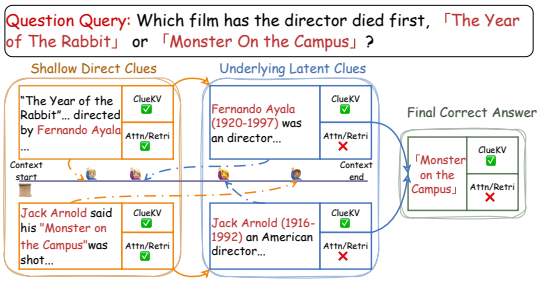

Figure 1: A typical long context reasoning example. Classic attention-based compression tends to miss some latent clue tokens, leading to incorrect reasoning outcomes, which motivates ClueKV's clue-aware design.

To alleviate this bottleneck, a growing line of work studies KV cache compression, aiming to retain only a subset of KV pairs while preserving downstream accuracy. Existing methods predominantly rely on generic attention-based heuristics (Li et al., 2024; Feng et al., 2024; Devoto et al., 2025) to rank tokens for compression. While effective for tasks where relevant information is explicitly presented (e.g., fact retrieval), these approaches often face challenges in long-context reasoning scenarios due to their design lacking any mechanism for reasoning-critical information awareness. Specifically, in the ideal case, we expect a compression method to be able to identify and preserve all the *clue tokens* that are essential to forming a correct and complete deduction chain in long-context reasoning. However, among these tokens, some are implicitly connected through latent logical chains and do not necessarily receive the strongest direct attention scores or exhibit the most apparent surface relevance to the query. As a result, they are easily overlooked by existing attention-based methods (Fig. 1 shows a typical case). We refer to such easily neglected yet crucial tokens as *latent clue*

[1] Code will be released for reproducibility upon acceptance.

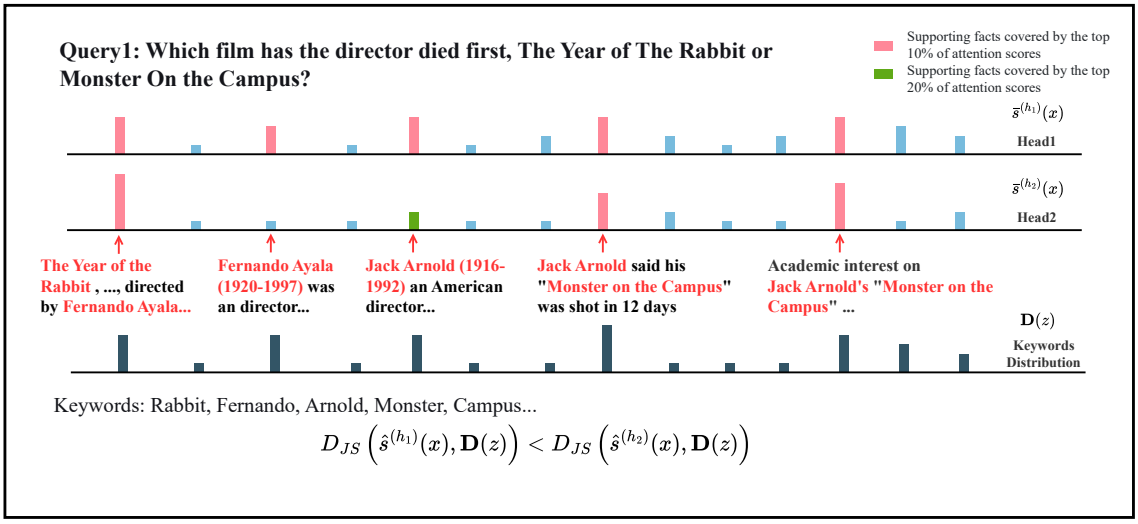

Figure 2: Comparison of attention coverage between a clue-aligned head (*Head 1*) and an irrelevant head (*Head 2*). For a complex query requiring multi-step deduction, Head 1 effectively identifies most necessary supporting facts (highlighted in red) within the top 10% of attention scores. Conversely, Head 2 misses critical information (e.g., death dates) even with a larger budget. This observation motivates our strategy to dynamically prioritize heads that align with the latent clue distribution.

tokens, as they often play an irreplaceable role in bridging the full reasoning path.

From the methodological perspective, the above limitation can be attributed to two common design choices in existing KV compression methods, both of which exhibit inherent limitations in complex reasoning. (1) *Uniform head treatment attention.* These methods (Li et al., 2024; Cai. et al., 2024b; Devoto et al., 2025) often aggregate attention scores across attention heads or treat all heads uniformly when evicting tokens, implicitly assuming that all heads contribute equally. However, in practice, only a subset of attention heads are aligned with reasoning-relevant regions. Uniform manners therefore dilute these signals. (2) *Head-aware token eviction but without reasoning awareness.* More recent methods (Feng et al., 2024; Tang et al., 2026) introduce head-level functional differentiation, but their optimization objectives remain agnostic to reasoning structure. As a result, they still remain largely blind to latent clue tokens that are essential for bridging multi-step inference chains.

To equip compression with reasoning awareness, we leverage our observation that clue signals are captured highly unevenly across attention heads, and that quite a few responsive heads vary across input instances, suggesting that effective compression should adaptively prioritize heads rather than rely on static heuristics. Meanwhile, because reasoning clues are often latent and difficult to supervise, self-guided clue signals derived from the input are particularly desirable.

Motivated by this, we propose **ClueKV**, a training-free KV cache compression framework that performs *self-guided, instance-adaptive head-wise token reweighting*. Specifically, ClueKV first applies contrastive probability shifts as a proxy signal for latent clue tokens without external supervision. It then summarizes these clue tokens into a context-level clue distribution, and identifies heads whose attention patterns align with this distribution and reweights token importance accordingly before KV eviction. In this way, ClueKV converts self-derived clue signals into adaptive compression guidance, instead of relying on static heuristics without reasoning awareness.

Our contributions can be summarized as follows:

- **Novel insight for reasoning-oriented KV compression.** We identify a key limitation of existing KV cache compression methods for long-context reasoning: generic attention-based scoring tends to overlook latent clue tokens needed to form multi-step reasoning chains, potentially leading to incomplete reasoning and incorrect predictions.

- **Self-guided, instance-adaptive head prioritization for KV compression.** We propose **ClueKV**, a training-free KV cache compres-

sion framework that uses contrastive probability shifts as self-guidance signals to capture token-level clues relevant to long-context reasoning, and further leverages these signals to perform head prioritization and token reweighting.

- **Empirical effectiveness and practical deployability.** ClueKV consistently outperforms strong baselines under the same cache budgets, with the largest gains on reasoning-intensive tasks. It retains 99.21% of full-KV accuracy on LongReason at a 50% cache budget, remains robust up to 64k context length, and requires no additional training or supervision, making it easy to deploy.

2 Related Works

Token-Eviction KV Cache Compression. Early KV cache compression methods reduce memory usage by evicting tokens estimated to be less important according to heuristic or score-based criteria, such as H2O (Zhang et al., 2023) and TOVA (Oren et al., 2024). Subsequent methods incorporate richer feature-level information to improve token importance estimation, including CAOTE (Goel et al., 2025) and KeyDiff (Park et al., 2025). Other approaches, such as NACL (Chen et al., 2024) and Lookahead Q-Cache (Wang et al., 2025), further develop more advanced strategies to estimate token importance for future decoding.

Head/Layer-Reallocation KV Cache Compression. Another line of work improves compression by exploiting heterogeneity across attention heads or transformer layers. FastGen (Ge et al., 2023) leverages head-specific patterns for adaptive caching. RazorAttention (Tang et al., 2024) and DuoAttention (Xiao et al., 2025) distinguish between full-context "retrieval" heads and local "streaming" heads, while AdaKV (Feng et al., 2024), LU-KV (Tang et al., 2026) and LAVa (Shen et al., 2025) dynamically redistribute cache budgets across heads or layers. LayerKV (Xiong et al., 2024) introduces layer-wise offloading, and KVSharer (Yang et al., 2024) improves efficiency through cross-layer cache sharing.

Context-Adaptive KV Cache Compression. Recent methods adapt compression policies to the input context or evolving generation state. DynamicKV (Zhou et al., 2025) allocates layer-wise budgets according to task-specific patterns, while OjaKV (Zhu et al., 2025) uses online PCA to adapt low-rank representations to changing context distributions. ContextCache (Yan et al., 2025) further incorporates dialogue history for adaptive cache management.

Unlike prior methods, ClueKV uses contrastively identified clue signals to guide instance-adaptive KV cache compression, aiming to better preserve reasoning-critical information in long-context reasoning.

3 Observation

To better understand the limitations of existing KV cache compression methods in complex long-context reasoning and to motivate our approach, we analyze both token-level clue patterns and head-level attention behaviors in LLMs. Our analysis yields three key observations.

Unsupervised contrastive probability shifts provide an effective signal for identifying reasoning-relevant tokens. We find that contrastive probability shifts can reliably surface the reasoning-critical clues required for multi-step reasoning. To illustrate this, we evaluate the effectiveness of contrastive probability shifts on LongReason (Zhan Ling, 2024) by measuring their recall against ground-truth reasoning-critical tokens. As reported in Table 1, contrastive probability shifts consistently outperform attention-based selection and frequency-based heuristics. For a more detailed analysis of the reliability of CPS, please refer to the appendix D.

Recall (%)	16K	32K	64K	128K	Ave.
CPS (Ours)	78.88	71.90	66.15	55.14	68.02
Top-Attn	45.25	47.13	44.21	37.89	43.87
Most-Freq	24.35	20.99	12.15	9.71	16.81

Table 1: Recall of selected tokens against ground-truth reasoning-critical tokens on LongReason. Higher recall indicates better identification of tokens essential for multi-step deduction. contrastive probability shifts (CPS) consistently outperforms attention-based and frequency-based baselines. See Appendix D.1 for detailed experimental setup.

Specific attention heads exhibit notably stronger alignment with latent clues. Given the clue tokens identified above, we analyze how different attention heads interact with them. As shown in Figure 2, attention heads exhibit clear functional specialization. While many heads focus on local or

syntactic patterns, a subset consistently aligns with the latent clue tokens identified by CPS. These *clue-aligned* heads act as reliable carriers of reasoning-relevant information. This suggests that prioritizing such heads is critical for preserving reasoning clues during KV compression.

The set of "clue-aligned" heads varies substantially across inputs. Although clue-aligned heads consistently emerge, their identities vary significantly across different inputs. As shown in Figure 3, there is only limited overlap among the most clue-aligned heads across examples. This indicates that no fixed subset of heads can universally capture reasoning-critical information, highlighting the need for instance-adaptive head selection.

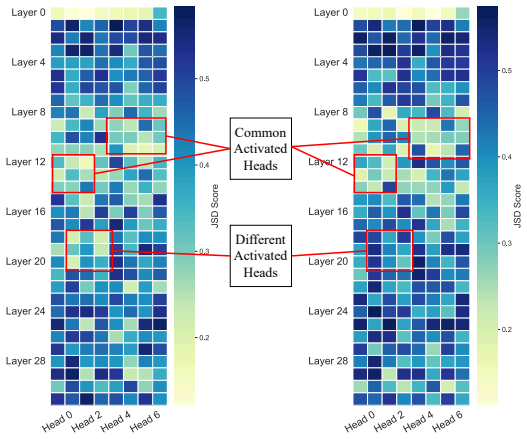

Figure 3: Variability of clue-aligned attention heads. Heatmaps show the JSD between attention and latent clues (lighter = stronger alignment). While some heads are consistently aligned, others are sample-specific, motivating ClueKVs instance-wise dynamic selection over static heuristics.

Based on these observations, we propose CLUEKV, a self-guided KV cache compression framework that uses contrastive signals to identify reasoning clue tokens and dynamically locate clue-aligned heads, thereby allocating cache budgets in a way that better preserves semantic dependencies for long-context reasoning.

4 ClueKV

In this section, we detail the algorithmic workflow of ClueKV, a training-free framework for KV cache compression. The overall workflow is illustrated in Figure 4, consisting of three stages: clue distribution estimation, head prioritization, and head-aware KV compression.

4.1 Estimating the Clues Distribution

Unlike explicit keywords, latent clue tokens are not directly associated with the query and are often overlooked by naive attention-based selection. As motivated by our observations in Section 3, we instead use contrastive probability shifts to estimate their distribution

Contrastive decoding has been shown to improve generation quality and reduce hallucination (Shi et al., 2024; Fang et al., 2024; Kai et al., 2024). We leverage its underlying probability shift signal to localize latent clue tokens. Specifically, we identify clue tokens by measuring changes in conditional decoding probabilities at the first decoding step after prefill, incurring only a one-time marginal overhead. Detailed efficiency analysis is provided in Appendix B. Let C denote the model's builtin default blank chat template, which serves as the contextfree reference (the specific configurations of C for each model are detailed in Appendix A), $C(S)$ be the formatted input sequence containing context S. Let V be the tokenizer vocabulary of the underlying language model. For vocabulary token $t \in V$, $P(t \mid C(S))$ represents the probability assigned by the model to t at the next decoding step given the input $C(S)$. Then, the growth rate of the conditional decoding probability is given by:

$$\Delta P_{\text{cond}}(t \mid S) = \frac{P(t \mid \mathcal{C}(S)) - P(t \mid \mathcal{C}(\varnothing))}{P(t \mid \mathcal{C}(\varnothing)) + \epsilon} \quad (1)$$

where ϵ is a small constant (e.g., 1e-8) for numerical stability. We identify the top-κ tokens with the highest ΔP_{cond} as the candidate set $T_{\text{top-}\kappa}$.

$$\mathcal{T}_{\text{top-}\kappa} = \underset{t \in V, |\mathcal{T}|=\kappa}{\arg\max}\, \Delta P_{\text{cond}}(t|S). \quad (2)$$

After filtering out common stop words using a list S_{stop}, we obtain the refined intent keyword set T_{clue}:

$$\mathcal{T}_{\text{clue}} = \{t \in \mathcal{T}_{\text{top-}\kappa} \mid t \notin \mathcal{S}_{\text{stop}} \wedge \Delta P_{\text{cond}}(t|S) > 0\}. \quad (3)$$

Intuitively, by normalizing against the prior probability $P(t|C)$, our approach filters out high-frequency background noise and amplifies the signals of implicit reasoning-clue nodes that are statistically essential for bridging the reasoning gap. Some examples are shown in Appendix D.3.

Following keyword extraction, we model their spatial density via a probability vector $\mathbf{D}$, computed through segment-wise frequency analysis (Alg. 1). This "semantic heatmap" exploits the inherent sparsity and local clustering of evidence in

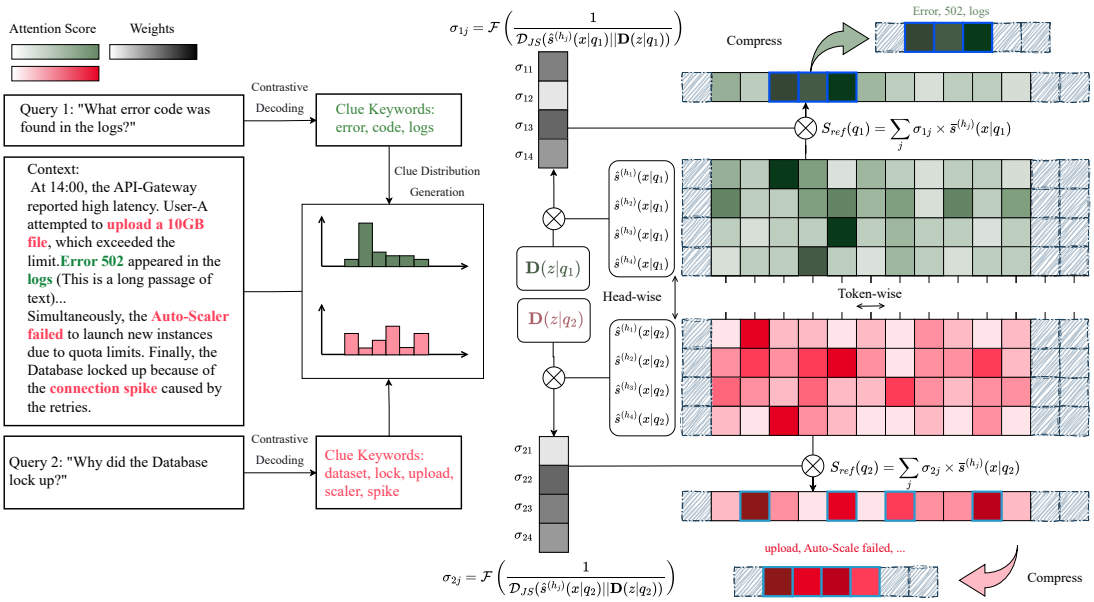

Figure 4: **Overview of the proposed framework.** Given a long context and a specific user query, the model first employs Contrastive Decoding to generate a clue distribution, identifying key semantic anchors (e.g.,"error" vs. "lock"). These clues guide the computation of Head-wise importance weights (σ), which synthesize the attention score matrix for token selection. Finally, a Token-wise selection is performed to prune irrelevant information, resulting in a highly compressed context that retains only inference-critical tokens (e.g., "Error 502" for Query 1,"connection spike" for Query 2).

Algorithm 1 Obtain the Distribution of Clues

Input: Full Context S, Latent Clue Set $\mathcal{T}_{\text{clue}}$, Chunk Size c
Output: Spatial Distribution Vector $\mathbf{D} \in \mathbb{R}^K$ with $K = \lceil \text{len}(S)/c \rceil$
Partition S into K segments $\{s_1, \ldots, s_K\}$
for $k = 1$ **to** K **do**
  $N[k] \leftarrow |\{t \in s_k \mid t \in \mathcal{T}_{\text{clue}}\}|$
end for
$N_{\text{total}} \leftarrow \sum_{k=1}^{K} N[k]$
for $k = 1$ **to** K **do**
  $d[k] \leftarrow N[k] / \max(N_{\text{total}}, 1)$
end for
return $\mathbf{D}$

long-context scenarios. Consequently, $\mathbf{D}$ provides a self-guided signal to prioritize reasoning-critical regions, ensuring that the KV cache compression preserves the connective tissue of the deduction chain.

4.2 Leverage Clue Distribution for Head Prioritization

Motivated by the insights from Section 3, we recognize that dynamically identifying attention heads aligned with latent clues is pivotal for preserving the full chain of reasoning clues needed to reach correct answers. Accordingly, in this section, we propose an algorithm for Head Prioritization via Clue Distribution. To formally describe this algorithm, let $\mathbf{X} \in \mathbb{R}^{L \times d}$ denote the input embedding matrix with sequence length L and model dimension d. We assume N_h attention heads in current layer with a per-head dimension of $d_k = d/N_h$. For each head $h \in \{1, \ldots, N_h\}$, we denote its QKV matrix as $\mathbf{Q}^{(h)}, \mathbf{K}^{(h)}, \mathbf{V}^{(h)} \in \mathbb{R}^{L \times d_k}$. Then, we use an observation window of size L_q to estimate the attention distribution of the current head. Let $\mathbf{Q}_{\text{win}}^{(h)} \in \mathbb{R}^{L_q \times d_k}$ represent the query states within this observation window. We compute the attention weights of these recent tokens attending to the full context:

$$\mathbf{A}^{(h)} = \text{softmax}\left(\frac{\mathbf{Q}_{\text{win}}^{(h)} (\mathbf{K}^{(h)})^\top}{\sqrt{d_k}}\right) \in \mathbb{R}^{L_q \times L}. \quad (4)$$

Then, calculate the average value of the attention within the "observation" window.

$$\bar{\mathbf{s}}^{(h)} = \frac{1}{L_q} \sum_{i=1}^{L_q} \mathbf{A}_i^{(h)} \in \mathbb{R}^L, \quad (5)$$

where $\bar{\mathbf{s}}^{(h)}$ represents the aggregated attention distribution over the context for head h. To align

[2]The data with the best performance are marked in bold, and the second-best ones are marked with an underline.

Table 2: Performance comparison on LongReason with different cache rates and sequence lengths (16k – 64k).[2]

Qwen3-8B		LongReason (16k – 64k)							
Rate	Methods	16k		32k		64k		Ave.	
		w/ CoT	w/o CoT	w/ CoT	w/o CoT	w/ CoT	w/o CoT	w/ CoT	w/o CoT
	Full	71.91	65.87	68.14	56.17	67.38	54.66	69.14	58.90
0.5	**ClueKV**	**70.15**	62.09	**70.03**	55.04	**65.62**	**53.90**	**68.60**	57.01
	AdaKV	69.52	**63.48**	61.96	54.91	62.59	53.02	64.69	**57.14**
	Duo	68.26	62.22	63.22	55.67	58.19	51.13	63.22	56.34
	Expected	68.14	61.71	67.00	**56.55**	54.66	47.98	63.27	55.42
	PyramidKV	69.27	63.35	63.85	56.30	60.71	51.01	64.61	56.88
0.4	**ClueKV**	**70.78**	**64.23**	**65.11**	**57.05**	**62.97**	50.00	**66.29**	**57.09**
	AdaKV	69.14	63.10	62.59	55.04	58.82	49.87	63.52	56.00
	Duo	66.37	60.33	60.71	53.15	56.30	47.23	61.13	53.57
	Expected	65.11	59.07	61.96	54.79	56.55	47.48	61.21	53.78
	PyramidKV	67.00	60.96	62.47	54.91	61.34	**51.76**	63.60	55.88
0.3	**ClueKV**	**69.52**	**62.97**	**66.37**	**55.92**	**57.56**	47.48	**64.48**	**55.46**
	AdaKV	66.37	60.20	63.22	55.67	57.30	**48.49**	62.30	54.79
	Duo	65.74	59.70	60.08	53.02	56.17	45.34	60.66	52.69
	Expected	64.23	59.07	61.34	53.78	55.42	47.73	60.33	53.53
	PyramidKV	66.25	60.33	61.21	54.28	56.93	47.86	61.46	54.15

the granularity of the attention distribution with the clue distribution $\mathbf{D}$, we aggregate the token-wise attention scores $\bar{\mathbf{s}}^{(h)}$ into segment-wise scores $\hat{\mathbf{s}}^{(h)} \in \mathbb{R}^K$ by summing within each segment. Then, we employ the Jensen-Shannon (JS) divergence $D_{JS}(\hat{\mathbf{s}}^{(h)}, \mathbf{D})$ to quantify the alignment between the attention distribution of the h-th head and the latent clue distribution $\mathbf{D}$:

$$\begin{aligned}\mathcal{L}_{\text{align}}^{(h)} &= 1/\left(D_{JS}(\hat{\mathbf{s}}^{(h)} \parallel \mathbf{D}) + \epsilon\right) \\ &= 1/\left(\frac{1}{2}D_{KL}(\hat{\mathbf{s}}^{(h)} \parallel \mathbf{M}) + \frac{1}{2}D_{KL}(\mathbf{D} \parallel \mathbf{M}) + \epsilon\right),\end{aligned} \tag{6}$$

where $\mathbf{M} = \frac{1}{2}(\hat{\mathbf{s}}^{(h)} + \mathbf{D})$ is the mixture distribution, and $D_{KL}(\cdot \parallel \cdot)$ denotes the Kullback-Leibler divergence. Since $D_{JS} \in [0, 1]$, a lower value indicates a stronger distributional alignment, allowing us to prioritize heads that effectively capture the latent clues. For JSD method validation, please refer to D.4.

4.3 Head Prioritized KV Cache Compression.

We normalize the alignment scores via Softmax to derive the importance weight $\sigma^{(h)}$ for each head:

$$\sigma^{(h)} = \text{Softmax}(\mathcal{L}_{\text{align}}^{(h)}) = \frac{\exp(\mathcal{L}_{\text{align}}^{(h)}/\tau)}{\sum_{j=1}^{N_H} \exp(\mathcal{L}_{\text{align}}^{(j)}/\tau)}, \tag{7}$$

where N_H denotes the number of heads in current layer and τ denotes the temperature parameter. Next, we synthesize a global *Reference Attention Score* $\mathbf{S}_{\text{ref}}$ by aggregating head attention distributions:

$$\mathbf{S}_{\text{ref}} = \sum_{h=1}^{N_H} \sigma^{(h)} \cdot \bar{\mathbf{s}}^{(h)}. \tag{8}$$

This mechanism amplifies heads capturing semantic clues while suppressing noise. Finally, we formulate KV cache compression as a subset selection problem. We seek an index set $\mathcal{I}^*$ of size B that maximizes the cumulative reference attention:

$$\mathcal{I}^* = \underset{\mathcal{I} \subset \{1,\ldots,L\}, |\mathcal{I}|=B}{\arg\max} \sum_{t \in \mathcal{I}} \mathbf{S}_{\text{ref}}[t]. \tag{9}$$

Accordingly, the Key and Value matrices are pruned to preserve only the most informative context:

$$\mathbf{K}' = [\mathbf{k}_t]_{t \in \mathcal{I}^*}, \quad \mathbf{V}' = [\mathbf{v}_t]_{t \in \mathcal{I}^*}. \tag{10}$$

5 Experiments

Models. We conduct our experiments using several representative open-source Large Language Models (LLMs): Qwen3-8B (Team, 2025), Llama3.1-8B-Instruction (Grattafiori et al., 2024), and Mistral-7B (Jiang et al., 2023). These models are widely adopted in long-context research and serve as established baselines for evaluating compression methods.

Benchmarks. Experiments are conducted on three established long-context benchmarks: LongBench (Bai et al., 2024), LongReason (Zhan Ling,

Table 3: LongBench Score Performance of Different KV Compression Schemes on Llama3.1-8B-Instruction

Budget	Method	Avg	Single-Document QA				Multi-Document QA			Synthetic		
			NtrvQA	Qasper	MF-en	MF-zh	HotpotQA	2WikiMQA	Musique	PR-en	PR-zh	P-cnt
Llama3.1-8B-Instruction												
	Full KV	52.69	29.69	45.67	55.21	62.12	55.99	44.97	31.34	99.50	93.00	9.41
1024	**ClueKV**	**51.52**	**28.07**	**45.38**	54.76	61.77	**55.72**	43.64	**30.98**	99.50	**93.00**	6.76
	AdaKV	51.16	26.77	43.89	**55.05**	61.08	55.57	43.66	30.80	99.50	92.50	6.81
	SnapKV	51.11	26.71	45.17	54.23	61.73	55.80	43.74	29.50	99.50	92.50	6.60
	PyramidKV	51.14	26.93	43.26	55.03	61.56	55.26	**44.08**	30.38	99.50	92.00	6.76
	Expected	46.82	23.74	38.54	45.39	**61.85**	54.57	43.14	28.42	97.00	68.50	**7.08**
512	**ClueKV**	**50.80**	**27.37**	**42.30**	**54.76**	60.09	**55.94**	**43.79**	29.53	99.50	**92.50**	6.53
	AdaKV	50.15	26.01	40.69	53.51	**60.17**	55.91	42.66	29.93	99.50	90.50	6.98
	SnapKV	50.08	24.93	38.99	54.05	59.88	55.59	42.93	30.31	99.50	91.50	**7.55**
	PyramidKV	50.06	24.89	38.91	53.68	59.88	55.59	42.90	**30.35**	99.50	92.00	7.52
	Expected	42.40	23.27	36.42	41.83	40.75	53.35	40.49	28.86	93.00	59.00	7.08
Mistral-7B-v0.3												
	Full KV	46.77	28.52	37.03	51.94	50.12	50.22	34.38	24.06	98.00	88.50	5.00
1024	**ClueKV**	**45.79**	28.16	**36.76**	**50.49**	**50.52**	**50.07**	34.04	24.04	96.50	87.00	5.00
	AdaKV	45.58	27.09	36.33	49.34	49.46	49.29	33.89	24.58	96.50	**89.50**	4.50
	SnapKV	45.51	**28.38**	35.89	49.65	49.25	48.11	34.00	24.60	**97.00**	88.00	5.00
	PyramidKV	45.25	27.87	35.46	48.67	47.56	48.81	**34.51**	**24.80**	95.50	88.50	**5.50**
	Expected	37.31	25.98	32.61	41.04	36.33	48.39	31.88	23.49	77.00	57.00	3.00
512	**ClueKV**	**44.69**	27.38	**34.51**	**48.63**	**47.79**	**48.74**	**34.02**	23.54	**96.00**	86.00	5.00
	AdaKV	44.50	**27.47**	34.21	48.31	47.63	47.83	33.01	**24.31**	**96.00**	**87.00**	5.00
	SnapKV	44.08	25.10	32.39	48.41	47.51	47.94	33.32	23.83	95.50	86.50	5.00
	PyramidKV	43.96	25.05	31.83	48.31	47.49	47.79	33.34	23.53	95.50	86.00	**5.50**
	Expected	32.94	24.33	29.98	37.29	29.62	46.49	30.52	21.45	63.50	46.00	3.00

2024), and RULER (Hsieh et al., 2024). We evaluate on LongBench for general bilingual multi-task understanding, LongReason for multi-step deduction over scattered evidence, and RULER for controllable, scalable, and structured long-context tasks.

Baselines. We compare ClueKV against established KV compression baselines spanning both head-wise optimization methods (AdaKV (Feng et al., 2024), DuoAttention (Xiao et al., 2025)) and token eviction methods (PyramidKV (Cai. et al., 2024b), ExpectedAttention (Devoto et al., 2025), SnapKV (Li et al., 2024)). ClueKV uses a unified budget across layers. Full experiment settings are in Appendix A.

5.1 LongReason

As illustrated in Table 2, ClueKV yields the closest performance to Full KV across all settings. At a 50% cache rate, it retains 99.21% of the full-model accuracy. Compared to AdaKV and Duo, ClueKV exhibits much lower performance decay as the sequence length scales to 64k, validating that our headwise selection strategy effectively captures the sparse "clues" essential for complex reasoning over long contexts.

5.2 LongBench

We evaluate ClueKV on the LongBench benchmark using Llama3.1-8B-Instruction and Mistral-7B-v0.3 across various compression budgets. As summarized in Table 3, ClueKV consistently achieves the best or highly competitive average performance among all compression methods. These results indicate that ClueKV's headwise selection strategy more effectively preserves critical information compared to other methods. For the complete results of all the subtasks on the LongBench, please refer to appendix F.

5.3 RULER QA

Table 4 evaluates ClueKV on RULER QA (SQuAD (Rajpurkar et al., 2016) and HotpotQA (Yang et al., 2018)) across lengths up to 32k. ClueKV consistently achieves near-lossless performance relative to Full KV. These results confirm ClueKVs superior ability to pinpoint sparse evidence within noisy contexts compared to existing baselines. The main text presents ClueKV's performance on RULER QA; results on longer contexts (e.g., 128k) and the full suite of subtasks are provided in Section G.

Table 4: Average performance comparison on RULER QA under KV cache budgets of 512, 1024 and 2048.

Methods	8k		16k		24k		32k	
	SQuAD	HotpotQA	SQuAD	HotpotQA	SQuAD	HotpotQA	SQuAD	HotpotQA
Llama3.1-8B-Instruction								
Full	75.53	60.87	74.20	57.33	75.13	53.33	74.67	49.80
ClueKV	**75.13**	60.47	74.67	**56.87**	**76.00**	**51.13**	**74.20**	**48.87**
AdaKV	74.67	**60.87**	74.20	53.33	73.80	48.87	73.33	47.13
Expected	58.67	53.33	55.53	45.33	48.47	41.33	44.47	38.20
PyramidKV	71.53	57.33	**76.00**	52.47	73.33	48.87	73.80	46.67
TOVA	64.87	53.80	56.87	45.33	54.67	40.47	55.13	40.87
Mistral-7B-v0.3								
Full	73.80	49.33	70.67	47.13	66.20	43.53	69.33	44.47
ClueKV	**74.20**	**48.87**	**73.80**	46.20	**66.67**	**41.80**	**69.33**	**44.00**
AdaKV	70.20	48.47	69.80	**47.13**	66.20	**41.80**	67.53	43.53
Expected	64.67	40.87	58.87	37.80	56.20	35.13	51.53	36.47
PyramidKV	72.47	47.53	70.20	47.53	66.20	40.47	68.00	42.20
TOVA	71.13	47.53	69.80	43.53	64.87	41.33	66.20	42.67

5.4 Efficiency Analysis

We evaluate ClueKV's computational overhead and efficiency gains across the prefill and decoding stages, as illustrated in Figure 5.

Prefill Overhead. ClueKV incurs a marginal one-time prefill cost from contrastive clue detection and head-wise JS divergence computation. As shown in Figure 5 (Left), this overhead typically accounts for less than 5% of total prefill time, since the contrastive probability shift is computed only for the first decoding token. A detailed complexity analysis is provided in Appendix B.

Decoding Acceleration. The primary inference bottleneck lies in the I/O-bound autoregressive decoding stage. By pruning the KV cache to a compact subset, ClueKV substantially reduces per-token memory traffic. Figure 5 (Right) shows a significant reduction in per-token latency compared to Full KV, with measurements taken at an output length of 16k tokens to reflect long-form decoding. As sequence length grows, the cumulative decoding time saved quickly offsets the initial prefill overhead.

Memory Footprint. ClueKV also yields a deterministic reduction in GPU memory consumption during decoding (see Appendix C).

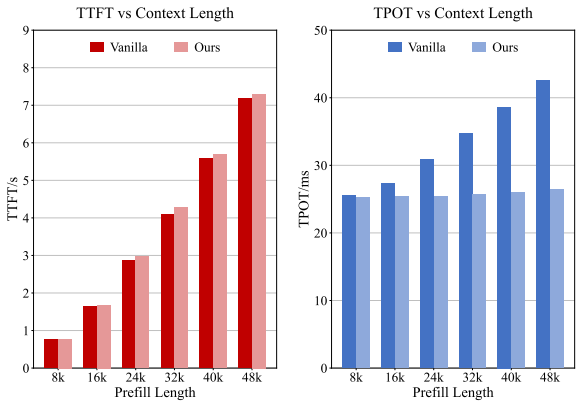

Figure 5: Efficiency of ClueKV (25% KV cache budget) vs. Vanilla (Full KV Cache) across context lengths. ClueKV achieves significantly lower and stable TPOT while maintaining competitive TTFT. Measured on a single NVIDIA A800 80GB GPU.

6 Conclusion

We presented ClueKV, a training-free KV cache compression framework that preserves reasoning-critical information under tight cache budgets. ClueKV introduces a self-guided mechanism to identify inference-critical information via contrastive probability shifts, by dynamically prioritizing attention heads aligned with these clues to retain inference-critical KV pairs. Extensive experiments across diverse benchmarks demonstrate that ClueKV achieves near-lossless performance even under aggressive compression budgets. Notably, ClueKV requires no additional training or external supervision, making it readily integrable into existing inference pipelines with minimal prefill overhead. Our work provides an efficient solution for deploying LLMs in real-world scenarios that require complex deduction.

7 Limitations

We note two limitations of ClueKV that also suggest directions for future improvement. First, its contrastive probability shift signal is most effective when reasoning-critical evidence forms distinct semantic anchors; on tasks requiring holistic understanding such as long-document summarization, the signal becomes less concentrated and ClueKV's advantage narrows (see Appendix F). Second, ClueKV identifies latent clues only once during prefill and reuses them throughout decoding, which may not cover evolving information needs in tasks like Long story generation—a practical simplification that dynamic refinement strategies may help. For a detailed analysis of the limitations of ClueKV, please refer to Appendix D.5.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. *Preprint*, arXiv:2308.14508.

Zefan Cai., Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. 2024b. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *Preprint*, arXiv:2406.02069.

Yilong Chen, Guoxia Wang, Junyuan Shang, Shiyao Cui, Zhenyu Zhang, Tingwen Liu, Shuohuan Wang, Yu Sun, Dianhai Yu, and Hua Wu. 2024. NACL: A general and effective KV cache eviction framework for LLM at inference time. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7913–7926, Bangkok, Thailand. Association for Computational Linguistics.

Alessio Devoto, Maximilian Jeblick, and Simon Jégou. 2025. Expected attention: Kv cache compression by estimating attention from future queries distribution. *Preprint*, arXiv:2510.00636.

Lizhe Fang, Yifei Wang, Zhaoyang Liu, Chenheng Zhang, Stefanie Jegelka, Jinyang Gao, Bolin Ding, and Yisen Wang. 2024. What is wrong with perplexity for long-context language modeling? *arXiv preprint arXiv:2410.23771*.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. 2024. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. *ArXiv*, abs/2407.11550.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. 2023. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*.

Raghavv Goel, Junyoung Park, Mukul Gagrani, Dalton Jones, Matthew Morse, Harper Langston, Mingu Lee, and Chris Lott. 2025. Caote: Kv caching through attention output error based token eviction. *arXiv preprint arXiv:2504.14051*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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

Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *ArXiv*, abs/2310.06825.

Jushi Kai, Tianhang Zhang, Hai Hu, and Zhouhan Lin. 2024. SH2: Self-highlighted hesitation helps you decode more truthfully. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4514–4530, Miami, Florida, USA. Association for Computational Linguistics.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. Snapkv: Llm knows what you are looking for before generation. *Preprint*, arXiv:2404.14469.

Matanel Oren, Michael Hassid, Yossi Adi, and Roy Schwartz. 2024. Transformers are multi-state RNNs. arXiv:2401.06104.

Junyoung Park, Dalton Jones, Matthew J Morse, Raghavv Goel, Mingu Lee, and Chris Lott. 2025. Keydiff: Key similarity-based kv cache eviction for long-context llm inference in resource-constrained environments. *arXiv preprint arXiv:2504.15364*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Conference on Empirical Methods in Natural Language Processing*.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. 2024. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *ArXiv*, abs/2407.08608.

Yiqun Shen, Song Yuan, Zhengze Zhang, Xiaoliang Wang, Daxin Jiang, and Nguyen Cam-Tu. 2025. Lava: Layer-wise kv cache eviction with dynamic budget allocation. *Preprint*, arXiv:2509.09754.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. 2024. Trusting your evidence: Hallucinate less with context-aware decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 783–791, Mexico City, Mexico. Association for Computational Linguistics.

Hanlin Tang, Yang Lin, Jing Lin, Qingsen Han, Shikuan Hong, Yiwu Yao, and Gongyi Wang. 2024. Razorattention: Efficient kv cache compression through retrieval heads. *Preprint*, arXiv:2407.15891.

Ziyao Tang, Pengkun Jiao, Xinhang Chen, Wei Liu, Shiyong Li, and Jingjing Chen. 2026. Predicting future utility: Global combinatorial optimization for task-agnostic kv cache eviction. *arXiv preprint arXiv:2602.08585*.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Yixuan Wang, Shiyu Ji, Yijun Liu, Yuzhuang Xu, Yang Xu, Qingfu Zhu, and Wanxiang Che. 2025. 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*, pages 34158–34174, Suzhou, China. Association for Computational Linguistics.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, junxian guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. 2025. Duoattention: Efficient long-context LLM inference with retrieval and streaming heads. In *The Thirteenth International Conference on Learning Representations*.

Yi Xiong, Hao Wu, Changxu Shao, Ziqing Wang, Rui Zhang, Yuhong Guo, Junping Zhao, Ke Zhang, and Zhenxuan Pan. 2024. Layerkv: Optimizing large language model serving with layer-wise kv cache management. *ArXiv*, abs/2410.00428.

Jianxin Yan, Wangze Ni, Lei Chen, Xuemin Lin, Peng Cheng, Zhan Qin, and Kui Ren. 2025. Contextcache: Context-aware semantic cache for multi-turn queries in large language models. *arXiv preprint arXiv:2506.22791*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yifei Yang, Zouying Cao, Qiguang Chen, Libo Qin, Dongjie Yang, Hai Zhao, and Zhi Chen. 2024. Kvsharer: Efficient inference via layer-wise dissimilar kv cache sharing. *Preprint*, arXiv:2410.18517.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Kai Yan Yifan Yang Weijian Lin Ting-Han Fan Lingfeng Shen Zhengyin Du Jiecao Chen Zhan Ling, Kang Liu. 2024. Longreason: A synthetic long-context reasoning benchmark via context expansion. *arXiv preprint arXiv:2501.15089*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. H2o: heavy-hitter oracle for efficient generative inference of large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Xiabin Zhou, Wenbin Wang, Minyan Zeng, Jiaxian Guo, Xuebo Liu, Li Shen, Min Zhang, and Liang Ding. 2025. DynamicKV: Task-aware adaptive KV cache compression for long context LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 8042–8057, Suzhou, China. Association for Computational Linguistics.

Yuxuan Zhu, David H. Yang, Mohammad Mohammadi Amiri, Keerthiram Murugesan, Tejaswini Pedapati, and Pin-Yu Chen. 2025. Ojakv: Context-aware online low-rank kv cache compression with oja's rule. *Preprint*, arXiv:2509.21623.

A Experiments settings

The settings of ClueKV in the experiment. In all the experiments conducted in this paper, the chunk size of ClueKV was consistently set at 128. The Prioritization of ClueKV on headwise is only carried out within the layer. In this paper, ClueKV does not perform inter-layer optimization. Furthermore, the selection of chat templates for contrastive decoding in ClueKV is shown in Table 6.

The parameter settings for LongBench and RULER. In the experimental setups of Ruler QA and longbench, the observation window size used by AdaKV, SnapKV and PyramidKV was set to 32, and the pooling kernel size was set to 9. The settings for ClueKV in this part were consistent with those of the other three.

The parameter settings for LongReason. In the LongReason dataset, since the length of the final question is longer, the selected observation window size is larger. The size of the observation window is set to the token length of the question, and the common size is around 64. The size of the selected pooling kernel size is still set to 9. Furthermore, since the combination of AdaKV and SnapKV yields the best results, the AdaKV used in the table is all Ada-SnapKV.

The prompts used in CoT and non-CoT sections of the LongReason. To evaluate the reasoning capabilities of models with and without Chain-of-Thought (CoT), we employ the following prompt templates for the LongReason benchmark. The selection of prompts is shown in Table 5.

Table 5: Comparison of Prompt Templates for LongReason

w/o CoT Prompt (Direct Answer)
Carefully read the provided context and answer the following multiple-choice question. Strictly adhere to the output format rules below: (1) If you conclude the answer cannot be determined from the provided context, output exactly: "[I'm not sure.]" (2) If you are confident in the correct answer, output exactly: "The answer is [Option]" (replace [Option] with A, B, C, D, or E). (3) Under no circumstances should you include explanations, reasoning, or any additional content beyond the required output format.

w/ CoT Prompt (Step-by-Step Reasoning)
Based on the given text, answer the question above. Let's think step by step. 1. First, Please analyze the question step by step, and analyze it only once. Do not analyze the question repeatedly. 2. Then, finish your response on the very last line in the exact format: "The answer is [Insert A, B, C, D or E here]"

B Computational overhead analysis

In this section, we provide a theoretical analysis of the computational complexity introduced by ClueKV. We demonstrate that the overhead is almost negligible compared to the standard prefill and decoding phases of Large Language Models (LLMs), particularly as the context length L scales.

B.1 Algorithmic Complexity Breakdown

Let L denote the context length, d the hidden dimension, N_H the number of attention heads, and V the vocabulary size. The total computational overhead of ClueKV, denoted as Ω_{ClueKV}, is composed of three primary stages:

Table 6: Chat Templates for Different Models.

Model	**Template Structure**

Llama-3.1-8B-Instruct	`<|begin_of_text|>`
	`<|start_header_id|>system`
	`<|end_header_id|>`
	
	`Cutting Knowledge Date: December 2023`
	`Today Date: 26 Jul 2024`
	
	`{Context+Query}<|eot_id|>`
	`<|start_header_id|>user`
	`<|end_header_id|>`

Qwen/Qwen3-8B	`<|im_start|>user` `{Context+Query}.<|im_end|>`

Mistral-7B-v0.3	`<s>[INST]`
	`{Context+Query}`
	`[/INST]</s>`

1. Clue Detection (Ω_{clue}): Unlike context-aware decoding methods that require multiple forward passes for the entire sequence, ClueKV calculates the contrastive probability shift ΔP_{cond} only for the *first* generated token.

- Obtaining $P(t|C(S))$ is a byproduct of the standard prefill pass.
- Calculating $P(t|C(\varnothing))$ for the empty template involves a sequence of length $|C(\varnothing)| \ll L$, incurring a constant cost $\mathcal{O}(|C(\varnothing)| \cdot d^2) \approx \mathcal{O}(1)$. And $P(t|C(\varnothing))$ only needs to be calculated once and can be obtained offline, and then it can be reused later.
- The contrastive shift calculation and top-m selection involve a pass over the vocabulary, with complexity $\mathcal{O}(V)$.
- Locating keywords in the context via a hash map or linear scan is $\mathcal{O}(L)$.

Thus, $\Omega_{\text{clue}} \approx \mathcal{O}(V + L)$, which is dominated by the linear embedding lookup cost.

2. Head-wise Alignment (Ω_{align}): For each head $m \in \{1, \ldots, N_H\}$, ClueKV aggregates token-wise attention into $K = L/c$ segments and computes the JS divergence.

- **Aggregation:** Summing scores into segments across all heads takes $\mathcal{O}(N_H \cdot L)$.
- **Alignment Scoring:** Computing D_{JS} for N_H heads takes $\mathcal{O}(N_H \cdot K)$.

Given $K = L/c$, the complexity is $\mathcal{O}(N_H \cdot L)$. Compared to the standard attention complexity $\mathcal{O}(L^2 \cdot d)$, this term is lower-order by a factor of Ld/N_H.

3. Token Selection (Ω_{sel}): The construction of the reference score S_{ref} and the subsequent top-B selection (where B is the KV budget) follows:

$$\Omega_{\text{sel}} = \mathcal{O}(N_H \cdot L + L \log B) \tag{11}$$

The $\mathcal{O}(L \log B)$ term represents the heap-based selection of the most informative KV pairs.

B.2 Asymptotic Negligibility

The total overhead introduced by ClueKV can be summarized as:

$$\Omega_{\text{ClueKV}} = \mathcal{O}(V + L \cdot N_H + L \log B) \tag{12}$$

In a standard Transformer layer, the prefill complexity is dominated by the quadratic attention term, $\Omega_{\text{prefill}} = \mathcal{O}(L^2 \cdot d)$. We evaluate the relative overhead ratio Γ:

$$\Gamma = \frac{\Omega_{\text{ClueKV}}}{\Omega_{\text{prefill}}} = \frac{\mathcal{O}(V + L \cdot N_H + L \log B)}{\mathcal{O}(L^2 \cdot d)} \tag{13}$$

As the context length scales ($L \to \infty$), we observe:

$$\lim_{L \to \infty} \Gamma = \lim_{L \to \infty} \frac{\mathcal{O}(L)}{\mathcal{O}(L^2)} = 0 \tag{14}$$

This derivation proves that ClueKV's overhead becomes theoretically vanishing as the sequence length grows, effectively resolving the "prefill paradox."

B.3 Hardware-Aware Implementation

From an architectural perspective, ClueKV's operations (vector subtractions, JS divergence, and pooling) are *memory-bandwidth bound*, whereas the standard prefill stage is *compute-bound* due to heavy Matrix Multiplications (GEMMs). Since memory-bandwidth bound operations are amenable to kernel fusion, ClueKV's selection logic can potentially be fused with existing operations to reduce overhead. This theoretical analysis suggests that the measured Time-to-First-Token (TTFT) increase can remain extremely low, while the subsequent decoding phase benefits from significantly reduced I/O pressure on the KV cache memory.

B.4 Empirical Latency Analysis under Large-Scale Settings

Our extended experiments (using Qwen3-8B) to 144K context length show that ClueKV's prefill overhead remains minimal and practically negligible compared to the substantial decoding speedup, as shown in Table 7:

Linear Overhead Analysis:

A linear regression on the overhead data yields:

$$\textit{Overhead} \approx 0.0205 \times (\textit{Length}/k) - 0.360 \tag{15}$$

with $R^2 \approx \mathbf{0.978}$, confirming a strong linear relationship between context length and ClueKV's overhead. This translates to only $\sim$20.5 ms per 1k tokens of additional prefill time. In fact, ClueKV introduces only $O(L)$ of additional theoretical overhead, which is dwarfed by the quadratic complexity $O(L^2)$ of the standard attention mechanism. For detailed theoretical derivations, please refer to Appendix B.2.

C Memory efficiency analysis

In this section, we analyze the memory footprint of the KV cache, which is the primary bottleneck for serving long-context Large Language Models (LLMs). Figure 6 illustrates the KV cache memory consumption of ClueKV compared to the Vanilla Transformer as the prefill length scales from 64k to 192k tokens.

C.1 Quantifiable Memory Reduction

As shown in our experimental results, ClueKV with a 25% budget consistently achieves a **75% reduction** in KV cache memory usage across all tested sequence lengths. For instance, at a prefill length of 192k tokens:

- **Vanilla Transformer** consumes approximately **23.6 GB** of GPU memory for the KV cache alone (at FP16 precision, batch size = 1).
- **ClueKV** reduces this footprint to only **5.9 GB**, saving over 17 GB of VRAM.

This substantial saving is critical for multi-user serving environments, as it allows the system to accommodate a $4\times$ larger batch size within the same memory constraints, directly translating to higher system throughput.

Table 7: Prefill latency (seconds) across different context lengths.

Prefill Length	24k	44k	64k	84k	108k	128k	144k
Vanilla	3.34	6.98	12.18	18.85	27.37	36.19	48.03
ClueKV	3.62	7.50	12.88	20.10	29.17	38.55	50.62
Overhead	0.28	0.52	0.70	1.25	1.80	2.36	2.59

C.2 Breaking the Memory Wall

The memory consumption of the Vanilla KV cache grows linearly with the sequence length L, following $M_{vanilla} \propto L \cdot d_{model} \cdot n_{layers}$. In contrast, ClueKV effectively modifies the growth constant by a factor of α (where $\alpha =$ cache retention rate).

More importantly, ClueKV substantially reduces the sustained KV cache memory footprint during the autoregressive decoding phase. As shown in Figure 6, with a 25% budget, ClueKV achieves a 75% reduction in decoding-stage KV cache memory. For instance, at a prefill length of 192k tokens, the full KV cache consumes approximately 23.6 GB of GPU memory, whereas ClueKV requires only 5.9 GB. While the peak memory during prefill remains unchanged, this decoding-stage saving is critical for multi-user serving environments, as it allows the system to accommodate a $4\times$ larger batch size within the same memory constraints, directly translating to higher system throughput.

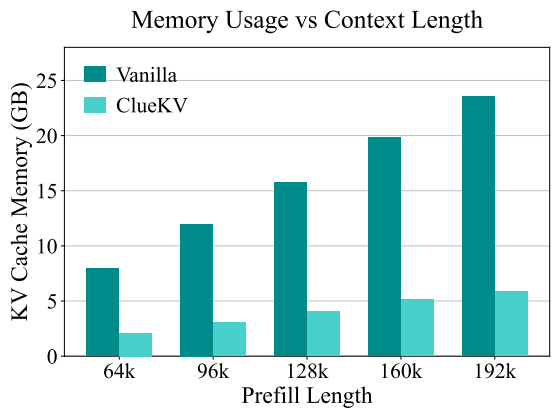

Figure 6: KV cache memory footprint in decoding phase for ClueKV (25% budget) vs. Vanilla at batch size 1. Our method achieves a 75% reduction in memory usage. The testing model uses Qwen3-8B

D ClueKV validity analysis

D.1 Analyzing reasoning-critical tokens selected via contrastive decode

We use LongReason to verify the effectiveness of the contrastive probability shift method in identifying reasoning-critical tokens. This dataset includes and expands upon the original dense clue short texts, which already contain complete reasoning elements and no redundant information. These original clue-dense texts after stopword filtering can more purely retain the core semantic units necessary for reasoning. These filtered tokens are used as ground truth reasoning-critical tokens.

Firstly, we conducted tests on LongReason to evaluate the model's accuracy for both the original essence short texts and various lengths. The specific results are as shown in tab 8.

Table 8: Model accuracy on original and various context lengths on LongReason.

Methods	Original	16k	32k	64k	128k
Qwen3-8B	82.12	71.91	68.14	67.38	62.72

The results show that using the original short text as the ground truth for reasoning-critical tokens is reasonable and there is a notable decline in the model's performance as context length increases due to the interference of redundant and irrelevant context.

Then, by analyzing and comparing the recall of the selected tokens by different methods for the ground-truth reasoning-critical tokens, we directly validated the reliability of contrastive decoding for identifying reasoning-critical tokens. We compared ClueKV with:

- **Top-Attn**: Selecting tokens with the top average attention score.
- **Most-Frequency**: Selecting the most frequently occurring token in the context.

The number of tokens selected for each method in the experiment was kept consistent with the number of GT tokens (thus recall = precision). The recall of the tokens selected by different methods for the ground-truth reasoning-critical tokens is shown in table 9.

The results demonstrate that the ClueKV method consistently outperforms the other base-

lines, achieving significantly higher overlap with ground-truth reasoning-critical tokens, validating the effectiveness of contrast decoding for identifying reasoning-critical tokens.

To provide an intuitive illustration, we present a case study from the benchmark, comparing the tokens selected by different methods and their impact on the final reasoning output.
Case Study:

- **Task**: Answer the question based on the novel *Enoch Soames*: "Why does Soames get upset with Beerbohm?"
- **With Contrastive Decoding**: ClueKV identified key semantic terms (e.g., betrayal, failure), enabling correct reasoning: *"Beerbohm portrayed Soames as a failed third-rate poet, leading to humiliation."*
- **With Most-Frequency**: Using most-frequent generic words (e.g., said, went) led to an incorrect answer: *"Soames is upset because Beerbohm did not appreciate his poetry."*
- **With Top-Attn**: Selecting tokens with the highest average attention scores (e.g., despise, Soame) led to an inadequate answer: *"Soames is upset because Beerbohm looked down upon him."*

This case clearly demonstrates that contrastive decoding method can effectively capture semantically meaningful tokens that are truly critical for reasoning, whereas frequency-based and attention-based selection strategies tend to retain generic or superficial tokens, ultimately leading to degraded answer quality.

D.2 A theoretical understanding of the effectiveness of probability shift comparison

We will first theoretical explanation for why contrastive probability shift provides a reasonable proxy for reasoning-relevant clues in this section. For a context S and a vocabulary token t, ClueKV computes:

$$\Delta P(t \mid S) = \frac{P_\theta\big(t \mid C(S)\big) - P_\theta\big(t \mid C(\emptyset)\big)}{P_\theta\big(t \mid C(\emptyset)\big) + \epsilon}.$$

Ignoring the small numerical stabilizer ϵ, this can be rewritten as

$$\Delta P(t \mid S) = \frac{P_\theta\big(t \mid C(S)\big)}{P_\theta\big(t \mid C(\emptyset)\big)} - 1.$$

Therefore,

$$\log\big(1 + \Delta P(t \mid S)\big) = \log \frac{P_\theta\big(t \mid C(S)\big)}{P_\theta\big(t \mid C(\emptyset)\big)}.$$

This quantity has the same form as a model-intrinsic pointwise mutual information proxy:

$$\mathrm{PMI}_\theta(S, t) = \log \frac{P_\theta\big(t \mid C(S)\big)}{P_\theta\big(t \mid C(\emptyset)\big)}.$$

Thus, CPS is a monotonic surrogate of a PMI-style likelihood ratio. Ranking tokens by $\Delta P(t \mid S)$ is therefore equivalent to ranking them by how strongly their predictive likelihood is increased by the given context relative to a context-free prior.

A token with high raw probability may simply be frequent under the language prior, while a token with high attention may only be lexically or positionally associated with the query. Neither criterion guarantees that the token plays a causal role in the deduction process. In contrast, a high PMI-style token is one whose predictive likelihood increases substantially only after conditioning on the reasoning clues in context. This shift in probability reflects a deeper change in the model's internal reasoning trajectory: when the full context S is provided, the model's predictive distribution is steered away from the generic prior toward a context-specific direction, and this deflection is concretely captured by the decoding probability shift. By measuring this shift, CPS surfaces the specific tokens that drive the model's reasoning in a particular direction, rather than those that are merely frequent or attention-salient. Such tokens tend to capture the entities, relations, dates, events, and intermediate variables that the model relies on to construct its reasoning chain. This is critical for long-context reasoning, where multi-step deduction requires connecting dispersed evidence through a sequence of intermediate semantic anchors. These anchors may not receive the highest attention scores or appear most frequently, yet they are indispensable for reconstructing the correct reasoning path.

Beyond this information-theoretic interpretation, two further considerations support the use of CPS at the first decoding step as a practical proxy for reasoning-relevant clues. First, by the time the model produces its first decoding token, it has already processed the full context through all transformer layers. The hidden states at this position therefore encode the complete information

Table 9: Recall of selected tokens for ground-truth reasoning-critical tokens.

Methods	16k	32k	64k	128k	Ave.
Ours	78.88%	71.90%	66.15%	55.14%	68.02%
Top-Attn	45.25%	47.13%	44.21%	37.89%	43.87%
Most-Frequency	24.35%	20.99%	12.15%	9.71%	16.81%

required for downstream deduction, and the first-token prediction reflects the model's initial reasoning direction, which typically governs the subsequent decoding trajectory over a non-trivial span of steps. Computing CPS at this step thus captures context-conditioned signals that are aligned with the model's upcoming reasoning path, without requiring access to future tokens. Second, the ideal contrastive baseline for isolating reasoning-critical information would be a variant S' of the context S from which all clue tokens have been removed, such that S' contains zero task-relevant information. However, constructing such an S' is impossible without already knowing which tokens are reasoning-critical, which would defeat the purpose. The empty chat template $C(\emptyset)$ provides a practical alternative: it also carries zero reasoning information about the current task, and it is universally available across all inputs without any manual construction. Using $C(\emptyset)$ as the contrastive prior therefore offers a generic, easily obtainable baseline that approximates the desired "reasoning-free" reference distribution.

Therefore, we use CPS as a self-guided clue signal for KV cache compression. Although CPS does not explicitly model the full reasoning structure directly, it approximates context-dependent informativeness at the token level. By preserving regions enriched with high-CPS tokens and prioritizing heads aligned with their distribution, ClueKV increases the chance that compressed KV caches retain the intermediate evidence required for multi-step reasoning.

D.3 Case study of the latent clue detection algorithm

In this section, we provide a qualitative analysis of the latent clue detection algorithm. Table 10 showcases several multi-hop inference queries and the corresponding keyword sets identified by ClueKV. These cases provide empirical evidence of how our contrastive decoding strategy effectively anticipates the information required to bridge the gap between a query and its supporting facts.

D.4 Validating JSD-Based Weighting for Reasoning-Critical Clue Preservation

To verify whether the proposed JSD-based head weighting can effectively identify heads that preserve reasoning-critical information, we conduct a controlled head-level analysis on a synthetic clue-reasoning dataset. The goal of this analysis is not to evaluate final task accuracy, but to directly test whether JSD-based weighting assigns larger importance to heads whose attention patterns better cover ground-truth reasoning clues.

Controlled clue-seeking dataset. Unlike standard long-context benchmarks, where the true reasoning structure is often entangled with natural-language noise, our synthetic setting explicitly controls two factors of reasoning complexity: reasoning depth and reasoning branch. Each instance is generated from one or more logical clue chains. A chain with depth d contains d sequentially connected clue sentences, where each clue points to the next intermediate evidence item.

For example, a depth-3 chain can be constructed as follows: "Alice's safe password is recorded in her notebook", "Alice's notebook says to look in her laptop", and "The back of Alice's laptop is marked with 9328493.". Given the query "What is Alice's safe password?", the model must follow the whole clue chain instead of relying on a single surface-matched sentence. To introduce multi-branch reasoning, we further create b parallel clue chains within the same instance. For instance, the context may contain Alice's, Bob's, and Carol's clue chains simultaneously, while both of their chains are mentioned in the final question.

After generating these clue chains, we randomly scatter all clue sentences into a long background context composed of semantically plausible but task-irrelevant distractor sentences. This prevents the model from solving the task using locality, adjacency, or simple positional heuristics. Since the dataset is procedurally generated, the token spans corresponding to all ground-truth reasoning clues are explicitly recorded, enabling direct head-level

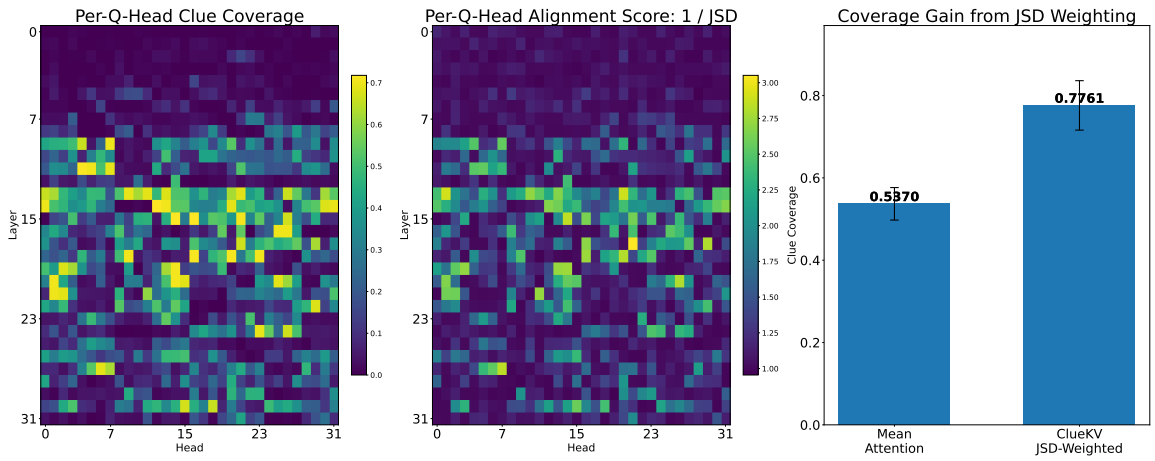

Figure 7: Validation of JSD-based weighting for preserving reasoning-critical clues on the controlled synthetic clue-reasoning dataset. Left: clue coverage of the top 25% attention-scored tokens for each query head, measured against ground-truth clue tokens. Middle: ClueKV alignment scores, computed as $1/\mathrm{JSD}$ between each head's attention distribution and the clue distribution. Right: average top-25% clue coverage comparison between uniform mean-attention aggregation and the proposed JSD-weighted aggregation. Error bars denote the standard deviation across transformer layers. JSD-based weighting substantially improves clue coverage, indicating that it can effectively identify and amplify heads aligned with reasoning-critical information. The testing model uses Llama3.1-8B-Instruction.

coverage measurement.

Coverage metric. For each input instance, we compute attention scores from the observation window near the query to all context tokens. For each query head h, we rank context tokens by their attention scores and select the top 25% tokens. We then measure how much of the ground-truth clue region is covered by these high-attention tokens:

$$\mathrm{Coverage}(h) = \frac{|\mathrm{Top}_{25\%}(A^{(h)}) \cap \mathcal{C}|}{|\mathcal{C}|},$$

where $\mathrm{Top}_{25\%}(A^{(h)})$ denotes the token set selected by the top 25% attention scores of head h, and $\mathcal{C}$ denotes the set of ground-truth clue tokens. A higher coverage score indicates that the head allocates more attention mass to reasoning-critical evidence and is therefore more clue-aligned.

JSD-based head weighting. We then compute a clue distribution over context segments based on the recorded clue regions. For each head, its token-level attention distribution is aggregated into the same segment granularity. We measure the Jensen–Shannon divergence between the head attention distribution and the clue distribution, and define the alignment score as:

$$s_h = \frac{1}{\mathrm{JSD}(A^{(h)}, D_{\mathrm{clue}}) + \epsilon},$$

where D_{clue} denotes the clue distribution and ϵ is a small constant for numerical stability, for more details, please refer to Alg. 1. A larger s_h indicates stronger distributional alignment between the head attention pattern and the reasoning-clue regions. These scores are then used to reweight heads before aggregating attention for token selection.

Results and analysis. We conducted the test on 360 data points across nine configurations (b from 1 to 3, d from 2 to 4) in our constructed 16k dataset and averaged the coverage results. Figure 7 visualizes the above results and the relationship between head-level clue coverage and JSD-based alignment. The left heatmap reports, for each query head, the coverage of ground-truth clue tokens among the top 25% attention-scored tokens. The middle heatmap shows the corresponding $1/\mathrm{JSD}$ alignment scores used by ClueKV. The two heatmaps exhibit similar structured patterns: heads with stronger clue coverage generally receive higher JSD-based alignment scores, while the relationship is not purely linear. This suggests that JSD captures distribution-level similarity between head attention and clue regions, rather than simply duplicating raw coverage.

The right panel compares two attention aggregation strategies. The conventional mean-attention baseline uniformly averages all query heads before selecting the top 25% tokens. In contrast, ClueKV uses the JSD-based alignment scores to upweight

heads whose attention distributions better match the clue distribution. As shown in the figure, JSD-weighted aggregation achieves substantially higher average clue coverage than mean attention. Error bars indicate the standard deviation across transformer layers.

Overall, this analysis validates the effectiveness of JSD-based weighting for preserving reasoning-critical clues. Since clue information is captured unevenly across heads, uniform averaging can dilute signals from clue-aligned heads. By contrast, JSD-based weighting identifies and amplifies heads whose attention patterns better match the clue distribution, thereby increasing the probability that KV compression retains tokens necessary for multi-step reasoning.

D.5 Analysis of Limitation Scenarios

Although ClueKV improves reasoning-oriented KV cache compression, we observe two scenarios where its advantages may become less pronounced.

Diffuse evidence in holistic summarization tasks. ClueKV relies on contrastive probability shifts to identify clue tokens that act as semantic anchors for downstream reasoning. This design is particularly effective when the answer depends on sparse but decisive evidence scattered across a long context. However, for holistic understanding tasks such as long-document summarization, style analysis, or broad thematic comparison, the useful information is often distributed across the entire document rather than concentrated around a few clue-bearing regions. In such cases, the contrastive clue distribution may become flatter and less discriminative, making head prioritization less effective. This does not necessarily harm performance severely, but it narrows the margin over strong attention-based compression baselines. For specific performance details, please refer to Table 13, where the performance of each method in summary-type tasks such as GovReport, QMSum, and MultiNews is presented.

A preliminary remedy is to combine clue-oriented compression with coverage-aware retention. For example, future variants could reserve a small portion of the cache budget for globally sampled tokens or segment-level representatives, while using the remaining budget for ClueKVs clue-guided selection. Another direction is to adapt the granularity of clue estimation: instead of relying only on token-level anchors, the model could estimate paragraph- or segment-level salience for tasks where evidence is semantically diffuse.

Static clue estimation during long-form generation. ClueKV estimates latent clues once during prefill and then reuses the resulting clue distribution throughout decoding. This choice keeps the method simple and efficient, since it avoids repeated contrastive computations during generation. However, the information need of the model may evolve over time, especially in long-form generation, multi-turn reasoning, or tasks where later decoding steps depend on intermediate conclusions. In these settings, clues that are important at the beginning may become less relevant, while new context regions may become important as the generated answer develops.

A possible extension is lightweight dynamic clue refresh. Instead of recomputing the clue distribution at every decoding step, the system could update it periodically or only when uncertainty signals indicate a distribution shift, such as high entropy, low confidence, or abrupt attention changes. Another practical strategy is to maintain a hybrid cache: a stable clue-preserving component obtained during prefill, together with a small adaptive component updated during decoding. This would preserve ClueKVs efficiency while allowing the cache to respond to evolving reasoning demands. Regarding this improvement, we are also planning to incorporate it as part of the upcoming work, as described in Section I.

Overall, these limitations suggest that ClueKV is most suitable for long-context tasks with sparse, reasoning-critical evidence. For more global or dynamically evolving tasks, future work may improve robustness by integrating coverage-aware allocation and lightweight clue refreshing.

E Ablation Study

In this section, we provide additional ablation studies to further validate the robustness of ClueKV and the rationale behind its core components.

E.1 Ablation Study of Clue Selection

To validate the design choices of ClueKV, we conduct a series of ablation studies on the Llama-3.1-8B-Instruction model, utilizing the NarrativeQA subset (from LongBench). For comparison, the black dotted line in the experimental result graph represents the outcome of the experiment without

Table 10: Case studies of keywords identified by ClueKV.

Query	Supporting Facts	Keywords Set of $\mathcal{T}_{\text{clue}}$
Where does the director of film *Man At Bath* work at?	(1) The film *Man at Bath* (2010) is directed by **Christophe Honoré**. (2) After moving to **Paris** in 1995, Christophe Honoré wrote articles in *Les Cahiers du Cinéma* (a **French** film magazine).	Christopher, film, Paris, French, ...
Who is the paternal grandmother of Marie Of Brabant, Queen Of France?	(1) Marie of Brabant, Queen of France was the daughter of **Henry III**, Duke of Brabant. (2) Henry III, Duke of Brabant was the son of Henry II of Brabant and **Marie of Hohenstaufen**.	Marie, margaret, mary, Henry, Hohenstaufen, ...
Which song came out first, *Joel The Lump Of Coal* or *Jugband Blues*?	(1) "**Jugband Blues**" is a song by **Pink Floyd** released on their second album *A Saucerful of Secrets* in **1968**. (2) "**Joel the Lump of Coal**" is a song by **The Killers** featuring **Jimmy** Kimmel, released on **December 1, 2014**.	Lump, Joel, Pink, Floyd, jimmy, ...
In which county was Larkin I. Smith born?	(1) Larkin I. Smith was born in Poplarville, Mississippi. (2) Poplarville is the county seat of **Pearl** River **County**, **Mississippi**.	Pearl, Mississippi, Smith, county, Marshall, ...
What city was the author of *What We Talk About When We Talk About Love* born in?	(1) The author of the collection *What We Talk About When We Talk About Love* is **Raymond** Carver. (2) Carver was born in Clatskanie, Oregon, a mill town on the **Columbia** River.	Pearl, Columbia, Raymond, Talk, Love, ...
The city that *Psycho Donuts* is located in is in which county?	(1) **Psycho Donuts** is a donut shop in **Campbell**, **California** opened in March 2009. (2) Campbell is a city in **Santa** Clara **County**, California, in the San Francisco Bay Area.	California, Campbell, Santa, Psycho, Donuts, ...

using latent clues, serving as the baseline. Specifically, we investigate the sensitivity of our model to the following critical components:

- **Impact of Chunk Size on Compression Fidelity:** We evaluate ClueKV across a spectrum of chunk sizes ($C \in \{32, 64, 128, 256\}$) on the LongBench subset. The results of the sensitivity experiment are shown in Figure 8. Under identical experimental settings, we find that sizes of 128 and 64 provide the most effective trade-off. Performance drops slightly at the chunk sizes of 256 and 32, indicating that moderate chunk sizes are required to maximize the efficacy of our method.

- **Robustness of Keyword Selection Strategies:** We employed two perturbation methods to verify the Robustness of our approach. One method involved randomly adding 3 to 5 interfering tokens to the latent clue set, while the other involved randomly discarding 3 to 5 tokens from the latent clue set for a comparative experiment. The experimental results are shown in Figure 9.

Due to space limitations, additional analysis of ablation experiments can be found in Appendix E

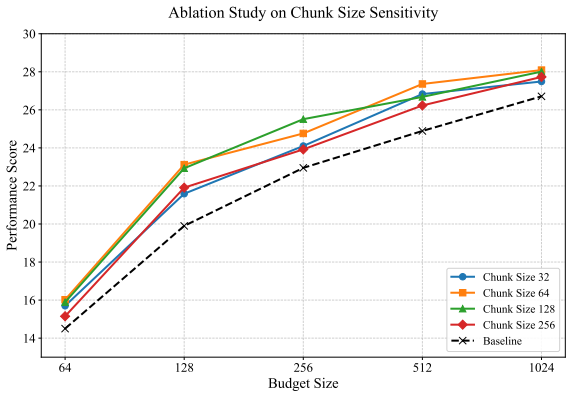

Figure 8: Sensitivity analysis regarding the choice of chunk size.

E.2 Sensitivity Analysis of Temperature τ

As formulated in Section 4.2, the temperature parameter τ plays a pivotal role in controlling the sharpness of the head importance distribution $\sigma^{(h)}$. To investigate its impact on reasoning performance, we conducted a sensitivity analysis on the LongRea-

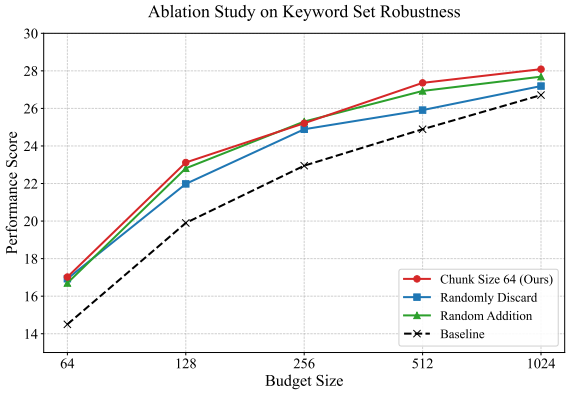

Figure 9: Performance stability is verified across varying keyword disturbance.

son (16k) and LongBench (Qasper) subsets using the Llama-3.1-8B-Instruction model with a 25% KV cache budget. We varied τ across a range of values $\{0.01, 0.1, 1.0, 10.0\}$. The results are summarized in Table 11.

Sharp vs. Smooth Prioritization. When τ is set to a low value (e.g., 0.01), the Softmax function approaches a argmax-like behavior, assigning nearly all weight to the single head with the highest alignment to the latent clue distribution. While this effectively captures the primary semantic pivots, we observe a slight performance degradation. This suggests that complex multi-hop reasoning often benefits from a “collaborative" effort among multiple top-ranked heads rather than relying on a single one.

The Dilution Effect. Conversely, as τ increases (e.g., 10.0), the importance weights $\sigma^{(h)}$ tend towards a uniform distribution ($\sigma^{(h)} \approx 1/N_H$). In this regime, the performance drops significantly and converges toward baseline methods that treat all heads equally. This confirms the functional specialization of attention heads observed in Section 3: if the specific signals from “clue-aligned" heads are diluted by noise from irrelevant heads, the model’s ability to retrieve scattered supporting facts is compromised.

Optimal Setting. We find that $\tau = 1.0$ provides the appropriate trade-off, sufficiently amplifying the heads that capture the spatial distribution of clues while maintaining a necessary level of diversity among relevant heads. Consequently, $\tau = 1.0$ is adopted as the default setting for all experiments in this paper.

E.3 Sensitivity Analysis of Clue Set Size κ

As defined in Section 4.1, the parameter κ determines the number of top-scoring candidate tokens extracted via contrastive decoding to form the latent clue set $\mathcal{T}_{clue}$. This parameter governs the trade-off between semantic coverage and signal-to-noise ratio. To evaluate its impact on reasoning performance, we conducted a sensitivity analysis on the LongReason (16k) and LongBench (Qasper) datasets using the Llama-3.1-8B-Instruction model at a 25% KV cache budget. We varied κ across $\{16, 32, 64, 128\}$, and the results are summarized in Table 12.

The experimental results reveal that a moderate value of κ is essential for optimal performance.

- **Small κ:** A significant performance drop is observed, particularly in multi-hop tasks like LongReason. This is because a restricted clue set often fails to capture all necessary semantic pivots required for complex deduction, leading to an incomplete spatial distribution D that misguides the head prioritization.

- **Large κ:** Increasing κ beyond 128 leads to a gradual decline in accuracy. While coverage is high, the inclusion of "semantic noise"—tokens that exhibit high probability shifts due to local context but are irrelevant to the core reasoning path—dilutes the precision of the clue distribution. This dilution makes the alignment score $L_{align}^{(h)}$ less discriminative, causing the model to lose focus on the most critical attention heads.

We find that $\kappa = 32$ provides a robust balance, ensuring sufficient coverage of reasoning anchors while maintaining a high signal-to-noise ratio across diverse tasks.

F Detailed results on all LongBench subtasks

To further evaluate the generalization ability of ClueKV, we conduct additional experiments on the LongBench benchmark, covering code generation, few-shot question answering, and long-document summarization tasks. The implementation follows the same KV compression setting with fixed token budgets. The detailed results are reported in Table 13.

As shown in the table, ClueKV maintains strong performance on code generation and few-shot question answering subtasks, achieving competitive or

Table 11: Impact of temperature τ on performance. Experiments are conducted with Llama-3.1-8B at a 50% compression budget.

Dataset	$\tau = 0.01$	$\tau = 0.1$	$\tau = 1.0$	$\tau = 10.0$
LongReason (16k)	47.98	49.62	**50.38**	47.60
NarrativeQA	27.51	28.34	**28.57**	27.93

Table 12: Impact of clue set size κ on performance. Experiments are conducted with Llama-3.1-8B-Instruction at a 50% compression budget. Bold indicates the best performance.

Dataset	$\kappa = 16$	$\kappa = \mathbf{32}$	$\kappa = 64$
LongReason (16k)	49.87	**50.38**	50.12
NarrativeQA	27.82	**28.57**	28.21

superior results on QMSum, RB-P, and Trec compared to other KV compression methods. On long-document summarization benchmarks, including GovReport, MultiNews, and SAMSum, ClueKV remains competitive but does not consistently outperform the current state-of-the-art approaches.

Overall, these results demonstrate that ClueKV generalizes well across diverse task types, while leaving room for further improvement on long-document summarization tasks.

G More Results on RULER Benchmark

To further validate the robustness of ClueKV, we conduct comprehensive experiments on the RULER benchmark across multiple backbone models, including Phi3-14B, Qwen2.5-32B, and Llama3.1-8B-Instruct.

For Phi3-14B and Qwen2.5-32B, we report results on retrieval and multi-hop reasoning tasks in Table 14, and QA/verification tasks in Table 15. Under the same 30% KV cache constraint, ClueKV consistently achieves higher average performance than both Ada-SnapKV and HeadKV across different task categories. In particular, the improvements are evident in multi-hop reasoning and multi-variable query tasks, highlighting the effectiveness of ClueKV in preserving critical contextual information under strict cache budgets.

We further evaluate Llama3.1-8B-Instruct across different context lengths (32K, 64K, and 128K), and report the averaged results in Table 16 and Table 17. As shown in these tables, ClueKV again outperforms competing KV compression methods under the same 30% cache constraint. The performance gains remain stable across varying context lengths, demonstrating strong generalization capability across both model architectures and input scales.

Overall, the consistent improvements observed in Tables 14–17 confirm the effectiveness and robustness of ClueKV on long-context reasoning benchmarks.

H GQA Interaction Details Explanation

In the derivations within the main body of the paper, we used MHA for illustration. However, our solution is also applicable under the GQA architecture model. In this section, we will provide a concise explanation. Unlike in the MHA, in the scenario based on the Group Query Attention (GQA) architecture, we perform the JSD scoring operation for each query head. However, this does not require introducing additional *kv_repeat* operations for each query head, and there is always only one shared kv cache within a group. In GQA, multiple query heads share the same KV set; thus, we compute only the JSD score over its shared KV set for each query head. Then, we perform the remaining compression on the shared KV cache within the GQA group, based on a JSD-weighted attention metric $\mathbf{S}_{\text{ref}}$:

$$\mathbf{S}_{\text{ref}} = \sum_{g=1}^{G} \sum_{h=1}^{N_g} \sigma^{(h)} \cdot \bar{\mathbf{s}}^{(h)}. \tag{16}$$

where G denotes the number of GQA groups, N_g is the number of query heads within each group, $\sigma^{(h)}$ is the JSD-based importance weight for the h-th query head, and $\bar{\mathbf{s}}^{(h)}$ is the normalized attention score vector of that head over the shared KV cache.

After obtaining $\mathbf{S}_{\text{ref}}$, we select the top-B KV positions with the highest scores as the retained cache entries for each GQA group:

$$\begin{aligned} \mathcal{I}^* &= \underset{\mathcal{I} \subset \{1,\ldots,L\},\, |\mathcal{I}|=B}{\arg\max} \sum_{t \in \mathcal{I}} \mathbf{S}_{\text{ref}}[t], \\ \mathbf{K}' &= [\mathbf{k}_t]_{t \in \mathcal{I}^*}, \quad \mathbf{V}' = [\mathbf{v}_t]_{t \in \mathcal{I}^*}. \end{aligned} \tag{17}$$

where B is the target KV cache budget per group.

Table 13: Performance comparison on additional LongBench subtasks under different KV cache budgets. ClueKV maintains strong performance on code generation and few-shot QA tasks. In summarization, although competitive, it is difficult to surpass current SOTA methods.

Budget	Method	GovReport	QMSum	MultiNews	Lcc	RB-P	Trec	SAMSum
	Full KV	34.99	25.49	27.22	62.72	57.11	72.50	43.73
2048	ClueKV	30.28	**25.26**	26.54	63.19	**57.69**	**71.50**	42.32
	HeadKV	29.99	24.71	**26.92**	**63.23**	57.32	70.50	**42.41**
	Ada-SnapKV	**30.61**	24.83	26.64	61.95	56.63	71.00	42.04
1024	ClueKV	28.86	24.48	26.42	62.95	**57.48**	70.50	**42.26**
	HeadKV	28.55	**24.65**	26.18	**63.02**	57.15	70.50	42.08
	Ada-SnapKV	**28.89**	24.42	**26.24**	60.94	56.22	**71.00**	41.84

Table 14: Comparison with HeadKV and Ada-SnapKV on RULER128K: Part1.

Methods	Cache	N-S1	N-S2	N-S3	N-MK1	N-MK2	N-MK3	Avg
Phi3-14B								
Phi3-14B	100%	100.00	76.62	78.68	43.79	5.75	4.19	51.84
+Ada-SnapKV	30%	100.00	71.20	61.30	39.80	4.10	2.55	46.49
+HeadKV	30%	100.00	74.42	62.58	41.07	4.41	0.70	47.86
+ClueKV	30%	100.00	73.01	78.32	42.68	4.01	3.54	**50.26**
Qwen2.5-32B								
Qwen2.5-32B	100%	100.00	90.50	96.63	73.38	19.93	7.86	64.72
+HeadKV	30%	100.00	87.69	76.96	68.36	14.60	1.72	58.22
+ClueKV	30%	100.00	86.46	95.83	71.88	14.58	6.25	**62.50**

It is worth emphasizing that, throughout this process, we do **not** need to perform any *kv_repeat* operation. Since all query heads within a GQA group share the same KV cache, the JSD scoring and attention computation are performed directly on the shared KV pairs, and only the final aggregated metric This design ensures that our method introduces **negligible additional memory overhead** compared to standard GQA inference (see Appendix B.2 for a detailed complexity analysis), while still benefiting from the fine-grained, per-head sensitivity analysis provided by the JSD-based scoring mechanism.

In summary, our approach seamlessly generalizes from MHA to GQA by: (1) computing per-head JSD scores using the shared KV cache within each group, (2) aggregating these scores into a unified reference metric $\mathbf{S}_{\text{ref}}$ through JSD-weighted summation, and (3) performing KV cache eviction based on this aggregated metric. This ensures that the retained KV entries best serve across all query heads within each group, achieving an effective balance between compression efficiency and generation quality.

I Future works

While ClueKV significantly reduces the KV cache footprint during the decoding stage, several promising avenues remain for further optimizing the system's efficiency and scalability.

I.1 Reducing Peak Memory via Early-Layer Clue Detection

A notable frontier for further refinement lies in managing the **transient peak memory footprint** during the initial prefill phase. While many current high-fidelity selection mechanisms, including ClueKV, rely on a comprehensive contextual analysis to ensure reasoning accuracy, this can lead to a temporary memory peak prior to the execution of pruning.

In future work, we aim to explore **Early-Exit Clue Detection**. Preliminary observations suggest that semantic anchors and "latent clues" often emerge within the first few layers (e.g., layers 1~4) of modern LLMs. By calculating the contrastive probability shifts ΔP using only these initial layerstreating them as a shallow "prober"we can esti-

Table 15: Comparison with HeadKV and Ada-SnapKV on RULER128K: Part2.

Methods	Cache	FWE	N-MQ	N-MV	QA-1	QA-2	VT	Avg
Phi3-14B								
Phi3-14B	100%	43.83	58.05	43.03	42.36	33.70	26.75	41.29
+Ada-SnapKV	30%	41.40	48.70	39.60	38.40	30.20	24.30	37.10
+HeadKV	30%	42.63	49.95	40.82	40.59	31.41	23.47	38.14
+ClueKV	30%	43.37	53.24	44.26	39.93	35.58	26.59	**40.50**
Qwen2.5-32B								
Qwen2.5-32B	100%	90.99	87.76	83.39	52.14	40.10	86.61	73.50
+HeadKV	30%	89.09	75.04	78.83	50.12	37.12	84.71	69.82
+ClueKV	30%	89.58	80.88	86.28	50.00	43.75	86.67	**72.86**

Table 16: Average performance on RULER (32K, 64K, 128K) using Llama3.1-8B-Instruct: Part1.

Methods	Cache	N-S1	N-S2	N-S3	N-MK1	N-MK2	N-MK3	Avg
Llama3.1-8B-inst	100%	100.00	100.00	97.87	100.00	100.00	93.07	98.49
+Ada-SnapKV	30%	100.00	92.93	88.27	90.93	71.33	56.67	83.35
+HeadKV	30%	100.00	97.07	82.93	93.73	76.67	65.60	85.00
+ClueKV	30%	100.00	95.33	97.47	97.47	79.73	78.67	**91.45**

mate the clue distribution $\mathbf{D}$ with a fraction of the computational cost and VRAM. This would allow the system to prune the KV cache for deeper layers *on-the-fly* during the first forward pass, effectively slashing the peak memory requirement and further accelerating the TTFT.

I.2 Dynamic Clue Refinement for Long-Form Generation

While ClueKV currently performs clue identification only once after prefill, long-form generation tasks (e.g., long story generation) often involve a shifting focus where the "information need" evolves as the decoding progresses.

In future work, we plan to transition from a static clue set $\mathcal{T}_{clue}$ to a Stateful Dynamic Clue Maintenance mechanism during the decoding phase. As the model generates new tokens, the system could periodically re-evaluate the importance of cached KV pairs by monitoring the semantic trajectories and attention patterns of the newly produced output. By implementing a “sliding-window clue update" or “clue re-indexing" loop, the model can evict KV pairs that are no longer relevant to the current reasoning branch and potentially “re-activate" previously suppressed information that gains new significance. This dynamic adaptation would allow ClueKV to maintain a lean memory footprint while ensuring that the most contextually pertinent “clues" are always prioritized throughout the entire generation lifecycle.

I.3 Cross-Layer Budget Reallocation

Our current implementation of ClueKV focuses on intra-layer head prioritization. A natural extension is **Inter-Layer Budget Reallocation**. Since different layers contribute unequally to deductive reasoning (with middle layers often handling the bulk of semantic synthesis), we plan to develop a global budget controller. By leveraging the clue distribution, the controller could skip KV cache storage entirely for redundant layers while allocating "extra" capacity to reasoning-heavy layers, achieving a more granular and optimal trade-off between accuracy and efficiency.

Table 17: Average performance on RULER (32K, 64K, 128K) using Llama3.1-8B-Instruct: Part2.

Methods	Cache	FWE	CWE	N-MQ	N-MV	QA-1	QA-2	VT	Avg
Llama3.1-8B-inst	100%	88.93	16.93	98.40	98.67	67.73	44.40	92.67	72.53
+Ada-SnapKV	30%	84.00	8.13	82.53	90.80	61.33	40.80	84.13	64.53
+HeadKV	30%	86.40	16.53	84.67	93.60	64.93	42.67	81.33	67.16
+ClueKV	30%	88.00	14.80	90.27	96.53	63.87	43.47	82.80	**68.82**