NOT ALL QUERIES ARE EQUAL: IDENTIFYING INFORMATIVE QUERIES FOR KV CACHE COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Long prompts make the KV cache the dominant memory cost, so a common remedy is to keep only part of it once prefill ends. The standard methods share one form: they sum the attention each key receives from prefill queries and keep the keys with the largest sums. They differ only in which queries they include. We show that queries differ sharply in how useful they are for this sum. A few informative queries point to the keys that generation will read, while most are noisy and do not, and the choice between them largely determines accuracy. We propose Q-Trust, which identifies informative queries from the attention of the user's request and selects the keys they point to through 2-hop attention. Without any training, Q-Trust improves accuracy over SnapKV by 1.72–2.56 points on Llama-3.1-8B, Qwen2.5-7B, and Mistral-7B-v0.3, and by 0.78–1.58 points over SnapKV with pooling. A Triton kernel that reuses the log-sum-exp from FlashAttention limits the prefill overhead to 3.5% for 16K-token prompts.

1 INTRODUCTION

In long-context inference, the KV cache grows linearly with context length, limiting batch size and increasing latency. A widely used remedy scores every key immediately after prefill and keeps only the top B.

The standard methods share a single scoring form. Each key is scored by the attention it receives from prefill queries: H2O (Zhang et al., 2023) sums over all queries, SnapKV (Li et al., 2024) over an observation window at the end of the prompt, and TOVA (Oren et al., 2024) over the last query alone. The methods differ only in the weight w given to each query, and to date w has been a *function of position*, blind to the query's content. A sentence that the request asks about may lie deep in the document and carry exactly what generation needs, yet a window method gives its queries no weight, and an accumulation method weights them no more than those of any other token.

We argue that w should reflect how informative each query is rather than its position. We first show that prefill queries differ widely in usefulness. A few informative queries point to the keys that generation will read, beyond what their position alone explains, whereas most queries are noisy and do not. The median informativeness of document queries is -0.004, indistinguishable from zero, while the top 1% reach $+0.229$. This gap carries over to accuracy. When we split document queries into quartiles by informativeness and let only one quartile enter the sum, the sampled LongBench score falls monotonically from 36.32 for the top quartile to 23.93 for the bottom, below a random subset of the same size.

It remains to identify informative queries. We propose **Q-Trust**, which decides which queries to trust by estimating their informativeness from the user's request. The tokens that the request attends to have queries that in turn attend to the context needed for the answer, so we weight each query by the attention its token receives from the request and add the attention of the weighted queries to the request's own attention. Because attention is followed twice, we call this 2-hop attention. Unlike prior methods, which select queries by position or generate pseudo-responses to obtain them, Q-Trust finds informative prefill queries from the request itself.

This score visits every attention row, so a naive implementation slows prefill considerably. We implement Triton (Tillet et al., 2019) kernels that reuse the log-sum-exp already computed by FlashAttention-2 (Dao, 2024) and skip the queries whose weight falls below a threshold τ.

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.

Our contributions are as follows.

- We cast attention-based KV cache compression as a weighted sum over queries and show that the choice of which queries to trust determines accuracy. Selecting more informative queries substantially improves accuracy under compression, whereas noisy queries fall below a random subset.
- Treating each prefill query as a predictor of future attention, we show that the attention of the user's request preferentially weights the predictive ones. Building on this observation, Q-Trust improves accuracy by 1.72–2.56 points over SnapKV and by 0.78–1.58 points over SnapKV with pooling on three models without any training.
- A kernel that reuses the log-sum-exp from FlashAttention, combined with selective computation, limits the prefill overhead to 3.5% for 16K-token prompts.

2 Query Informativeness

This section examines whether some queries are worth more in the sum than others, whether existing methods use them, and how to find them.

2.1 KV Compression as Query Weighting

Let N be the prompt length and $A(q, i)$ the prefill attention probability from query q to key i ($i \le q$). Ideally, compression preserves the future attention, the attention that the queries q_{N+t} of generated tokens will pay to prompt keys:

$$u(i) = \sum_{t=1}^{T} A(q_{N+t},\, i), \qquad \mathcal{I}^* = \text{top-}B\ u. \tag{1}$$

Because u cannot be observed during prefill, existing methods approximate it with a weighted sum of prefill attention:

$$\hat{u}(i) = \sum_{q} w(q)\, A(q, i). \tag{2}$$

H2O assigns uniform w to all queries, SnapKV to the last W queries, and TOVA to the last query alone. The model fixes A and the budget fixes B, so w is the only design choice.

Compression usually keeps the first few tokens (attention sinks) and the most recent tokens. We treat the first 4 tokens as sinks and the last 16 as recent tokens; the tokens in between are document tokens, and their queries are document queries. Document queries constitute most of the prompt, and on them the methods' choices of w diverge most.

2.2 Not All Queries Are Equally Informative

We define the informativeness of a query q as how well its attention row predicts u. A direct comparison, however, is dominated by structure shared by all queries, such as attention sinks and the locality induced by RoPE. Even the *mean attention over all queries*, which carries no query-specific content, reaches a cosine similarity of 0.321 with u, close to the 0.330 of the top 1% of individual queries.

We therefore remove this shared component before the comparison:

$$\iota(q) = \cos\big(\tilde{A}_q - \mathrm{proj}_{m_{b(q)}} \tilde{A}_q,\ \ \tilde{u} - \mathrm{proj}_{m_{b(q)}} \tilde{u}\big), \tag{3}$$

$$m_b = \frac{1}{|\mathcal{D}_b|} \sum_{q' \in \mathcal{D}_b} \tilde{A}_{q'}, \qquad \mathrm{proj}_m x = \frac{x^\top m}{\|m\|^2}\, m, \qquad b(q) = \big\lfloor \log_2(N - 1 - q) \big\rfloor.$$

Here $\tilde{A}_q$ and $\tilde{u}$ are $A(q, \cdot)$ and u restricted to document keys and normalized to sum to one. The index $b(q)$ is the bin of q's distance from the end of the prompt, $\mathcal{D}_b$ the set of document queries in bin b, and m_b their mean row, the component shared by queries at a similar distance. The operator $\mathrm{proj}_m x$ is the projection of x onto m, so both vectors lose their shared component before the cosine is taken.

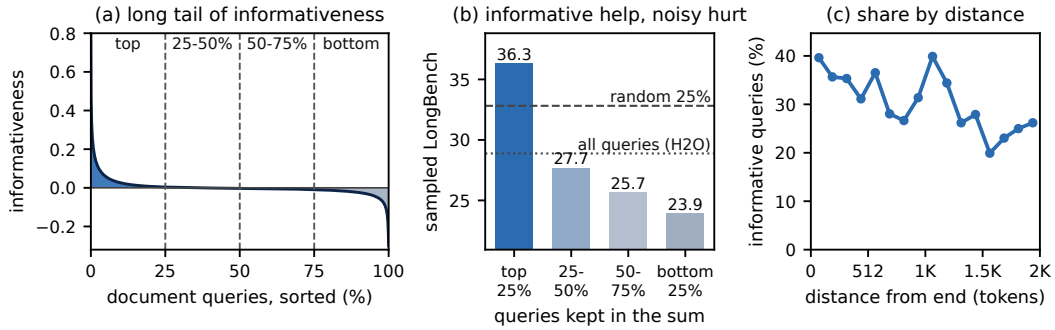

Figure 1: (a) Informativeness of document queries in sorted order; dashed lines mark quartile boundaries. (b) Sampled LongBench accuracy when only one quartile of document queries enters the sum; colors as in (a). (c) Share of informative queries (top quartile) among document queries in each 128-token bin of distance from the end of the prompt, up to 2K tokens. The overall share is 25%.

$\iota(q)$ measures *whether this query points to the keys read during generation better than a typical query at the same distance*. A positive value means the query attends, more than its position predicts, to keys that generation later reads; a negative value means it attends to keys that generation ignores. We compute ι from the model's own full-cache output. Measurements use sampled LongBench: up to the first 20 prompts of at least 1K tokens from each of the 16 LongBench tasks, truncated to 8K tokens (315 prompts for the accuracy experiments, whose accuracies therefore differ from the 32K setting of §4). They cover all 32 layers and 8 KV heads of Llama-3.1-8B.

Informativeness is long-tailed (Figure 1a). It stays flat near zero between the 25th and 75th percentiles, so these queries add nothing beyond their position, and it rises steeply in the top quartile. Queries in the lower tail have negative values and point to keys that generation ignores. We call queries that carry information beyond their position informative and the rest noisy. The same shape appears across task groups and layers.

The gap carries over to accuracy. We split document queries into quartiles by ι, compress using only one quartile with equal weights, and score the generated outputs. Every condition also includes the sinks and the last 16 queries, and a random subset of the same size serves as the control. The top quartile scores 36.32 (Figure 1b), $+3.50$ points above random (32.82; 95% confidence interval (CI) $[+1.56,\ +5.45]$). The remaining quartiles score 27.67, 25.66, and 23.93; the bottom quartile falls 8.89 points below random ($[-11.48,\ -6.55]$). The gap between the top and bottom quartiles, $+12.39$ points $[+9.73,\ +15.08]$, is positive in all six task groups, and the ordering holds when measured by similarity to the full-cache output. Only the few informative queries contribute to accuracy; the rest dilute the signal.

2.3 Position-Based Weights Miss Informative Queries

We measure the average informativeness of the queries a method trusts as $E_w[\iota] = \sum_q w(q)\iota(q)$.

Window methods ignore queries outside the observation window. As Figure 1c shows, 41% of queries within 64 tokens of the end are informative, and even beyond 2K tokens the share is 23%, close to the overall 25%. Since 69% of document queries lie beyond 2K tokens, 63% of informative queries do too, and 98.8% lie more than 64 tokens from the end. A 64-token window reaches only 1.2% of them.

Accumulation methods use every document query but cannot distinguish them. Under uniform weights, the noisy majority takes most of the weight, and the $E_w[\iota]$ of H2O is only $+0.004$. Weighting the same queries in proportion to ι raises it to $+0.126$: what matters is not which queries are used but which are trusted.

Table 1: Weight placed on informative queries by query weightings available at prefill (sampled LongBench, all layers and KV heads). Top 1%: weight on the top 1% of queries relative to uniform. Top 25%: share of the weight on the top quartile.

	Llama-3.1-8B		Qwen2.5-7B		Mistral-7B-v0.3	
Document-query weight	Top 1%	Top 25%	Top 1%	Top 25%	Top 1%	Top 25%
Uniform (H2O)	1.00×	25.0%	1.00×	25.0%	1.00×	25.0%
Accumulated attention of all queries	1.29×	26.3%	1.28×	27.0%	2.77×	28.1%
Distance ($\propto 1/d$)	6.92×	34.6%	6.41×	33.9%	7.35×	35.3%
s	**22.33×**	**58.0%**	**22.45×**	**59.2%**	**11.50×**	**42.5%**

2.4 Request Attention Identifies Informative Queries

Let $s(j)$ be the attention that the last $W = 16$ queries (the observation window) pay to token j:

$$s(j) = \sum_{t \in \text{last } W} A(t, j). \tag{4}$$

The observation window holds the user's request, so $s(j)$ measures how much the request refers to token j.

Each prefill query row $A(z, \cdot)$ is a candidate predictor of the future attention u, and §2.2 shows that only a few of these predictors are informative. What makes a predictor good is its relevance to generation: for a question about an entity, the queries at the mentions of that entity attend to the sentences describing it, which is what generation needs. This motivates estimating u as a mixture of the predictors, each weighted by its relevance $p(z)$ to generation:

$$\hat{u}(i) = \sum_{z} p(z)\, A(z, i). \tag{5}$$

The relevance $p(z)$ is unknown during prefill, but the request states what generation needs, so the request's reference distribution, the normalized s, serves as our estimate of $p(z)$. The query of a token with larger $s(j)$ should then be more informative.

Table 1 confirms this. On Llama-3.1-8B, weights proportional to s place 58.0% of the document-query weight on the informative queries (top quartile) and 22.3 times the uniform weight on the top 1%, and s ranks first on all three models. In terms of the average informativeness of trusted queries (§2.3), s reaches $E_w[\iota] = +0.078$, $+0.071$, and $+0.055$ on the three models, while uniform weights stay below $+0.02$. Using the accumulated attention of all queries as the query weight instead of s is no better than uniform, and weighting by distance from the end, as existing methods do, puts only 34.6% on the top quartile. When queries are sorted by s into 20 equal bins, the share of informative queries rises monotonically from 7.0% in the lowest bin to 60.8% in the highest. Whether a query is informative depends on its relation to the request, not on its position.

The result does not depend on the definition of ι: with no shared component removed, or with only the document-wide mean removed, 69.2% and 70.5% of the weight, respectively, fall on the top quartile, and 64.0% does when keys within 32 tokens of each query are excluded. The weighting also translates into accuracy. In the setting of §2.2, weighting document queries by s scores 36.43, on par with using the informative queries (36.32) and above SnapKV (34.27). Restricting this weighted sum to the top 10% of document queries by ι raises the score to 39.69 ($+3.26$ points $[+1.46,\ +5.19]$ over s alone), while a random 10% (34.37) and the bottom 10% (30.18) fall below it. Finally, weighting document queries in proportion to ι, which requires the future attention, reaches 41.08. The choice of query weight thus leaves large headroom, and s recovers part of it from the request alone.

3 Q-Trust

3.1 Request-Guided Scoring

Q-Trust turns Eq. (5) into a score (Figure 2). The request's own queries are the closest proxy for generation, so their attention enters the score directly, and every other query is weighted by the

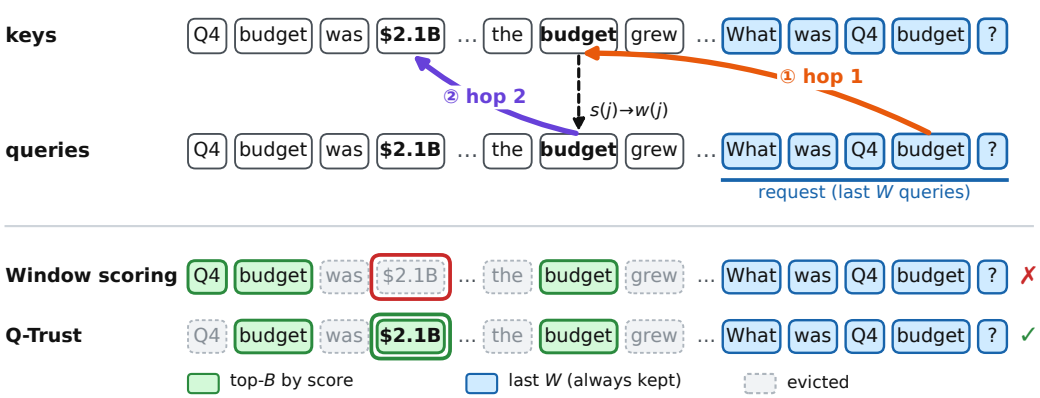

Figure 2: An illustrative example of Q-Trust, with the prompt drawn as keys (top) and queries (bottom). ① The request's query refers to the key of a mention of "budget" in the document (s in Eq. (4)), and this score becomes the weight w of the same token's query (dashed). ② That query refers to the key of the distant value "$2.1B" (Eq. (7)). Scoring with the request's attention alone evicts "$2.1B", whereas Q-Trust keeps it. The last W tokens are always kept.

estimate $p(z) \propto s(z)$:

$$w(q) \propto \begin{cases} 1, & q \in \text{last } W, \\ s(q), & \text{otherwise}, \end{cases} \tag{6}$$

normalized to sum to one. The score accumulates prefill attention with this weight and splits into two terms:

$$\text{score}(i) = \sum_{q \geq i} w(q)\, A(q,i) \;\propto\; s(i) + \sum_{q \notin \text{last } W} s(q)\, A(q,i). \tag{7}$$

The first term is the request's direct attention, which finds the tokens the request refers to (first hop); the second collects the keys that the queries of those tokens refer to (second hop). In Figure 2, the request attends to a mention of "budget" in the document, and the query of that mention attends to the distant value "$2.1B"; scoring by the request's attention alone would evict this value, which the second hop keeps. The first term must be retained because the second hop is causal: a query points only to keys before it, so a key that follows the tokens the request refers to, such as the value in "the budget was $2.1B", is reached only through the request's direct attention. The first hop is the same window pass that observation-window methods already compute, so Q-Trust adds only the second. Per KV head, the first hop costs $O(WN)$ and the second $O(|\mathcal{Q}|N)$, where $\mathcal{Q}$ is the set of query rows computed (§3.2).

The last W tokens are always kept, and the remaining $B - W$ are chosen by score. Each layer and KV head is handled independently. Q-Trust requires no training and has two hyperparameters, the window size W and the threshold τ of §3.2. We use $W = 16$ and $\tau = 1$ for all models and budgets and analyze both in §4.7.

3.2 Selective Computation

Document queries with small weight contribute little to the score, so we drop those with $w(q) < \tau/N$; the rows of the observation window are always computed. Since the uniform weight is $1/N$, τ expresses the threshold as a multiple of the uniform weight. On 8K-token prompts, 37.1% of rows are computed at $\tau = 0.25$ and 12.8% at $\tau = 1$.

3.3 Kernel

Eq. (7) visits every query row, so its cost is of the same order as one attention pass. A straightforward PyTorch implementation, however, recomputes the softmax and slows 16K-token prefill by 34% even at $\tau = 1$. We exploit two properties. First, FlashAttention-2 (Dao, 2024) computes the log-sum-exp LSE_q of each query while producing the output, so the attention probability can be

recovered without a softmax as $A(q,i) = \exp(\mathbf{q}_q^\top \mathbf{k}_i/\sqrt{d} - \mathrm{LSE}_q)$. Second, the score accumulates along the key axis, so parallelizing over key blocks avoids atomic operations. We implement Eqs. (4) and (7) as two Triton (Tillet et al., 2019) kernels, and Q-Trust runs on any model in the Transformers library without model-specific code.

4 Experiments

4.1 Setup

Benchmark and models. We evaluate on the 16 English tasks of LongBench (Bai et al., 2024), each scored with its original metric, and report the average. The metrics are F1 for QA, ROUGE-L for summarization, accuracy for classification, retrieval, and counting, and edit similarity for code. Tasks are grouped into Single-doc QA, Multi-doc QA, Summarization, Few-shot, Synthetic, and Code. We use Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen2.5-7B-Instruct (Yang et al., 2024), and Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), all with greedy decoding. The main text reports results on Llama-3.1-8B, and results on Qwen2.5-7B and Mistral-7B-v0.3 are in Appendix A. Prompts longer than 32K tokens are truncated in the middle, and generation lengths follow the LongBench settings for each task. All runs use PyTorch 2.8 and Transformers 5.10 on RTX 3090 and RTX 4090 GPUs.

Baselines. We compare against the attention-based H2O (Zhang et al., 2023), TOVA (Oren et al., 2024), and SnapKV (Li et al., 2024); the attention-free StreamingLLM (Xiao et al., 2024), L2-norm (Devoto et al., 2024), KeyDiff (Park et al., 2025), and TriAttention (Mao et al., 2026); and the reconstruction-based KVzip (Kim et al., 2025). All baselines are reimplemented in the same codebase. Like Q-Trust, H2O, TOVA, SnapKV, and KVzip always keep the last 16 tokens, and SnapKV uses an observation window of 16. We report SnapKV both with and without the max pooling (kernel size 7) of the original method, which also keeps the neighbors of each selected key (SnapKV + pooling).

Q-Trust. Unless stated otherwise, we use $W = 16$, $\tau = 1$, and a KV budget of 128.

4.2 Overall Results

Figure 3 shows the LongBench average for Llama-3.1-8B for KV budgets from 128 to 1024. At budget 128, Q-Trust outperforms SnapKV by $+1.91$ points and SnapKV + pooling by $+1.58$ points (95% CIs $[+1.38, +2.48]$ and $[+1.01, +2.17]$; paired bootstrap within tasks, 4,000 resamples). Pooling helps more as the budget grows, and SnapKV + pooling matches Q-Trust from budget 512 onward; at small budgets, the second hop, which selects keys the request never attends to, matters most. Against SnapKV without pooling, Q-Trust is higher at every budget, by $+1.21$ points even at 1024. Task-group scores at budgets 256–1024 are in Appendix B.

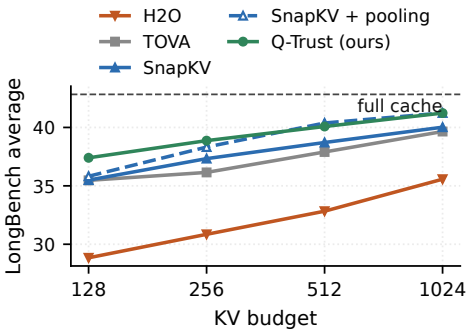

Figure 3: LongBench average for Llama-3.1-8B versus KV budget.

4.3 Results by Task Group

Table 2 reports scores by task group for Llama-3.1-8B at budget 128. Q-Trust scores highest in Few-shot and Code, and the largest gain is in Few-shot: $+6.87$ points over SnapKV and $+5.17$ points over SnapKV + pooling. SnapKV + pooling is best in Single-doc QA and Multi-doc QA, and H2O in Summarization, where uniform accumulation over the whole document suits the task; a flatter query weight brings Q-Trust to the same level (§4.7.3). In few-shot prompts, the examples needed for the answer often lie far from the end of the prompt, consistent with the finding in §2.3 that informative queries are spread across distant positions. The effect is strongest on TREC, whose prompts spread labeled examples of 50 question types over the whole context: Q-Trust scores 64.5 against 43.5 for SnapKV and 51.0 for SnapKV + pooling.

Table 2: LongBench scores by task group for Llama-3.1-8B-Instruct at KV budget 128. Rows marked "+ Ada-KV", below the thick rule, combine each score with the head-wise budget allocation of Ada-KV. SQA: Single-doc QA, MQA: Multi-doc QA, Sum.: Summarization, FS: Few-shot, Syn.: Synthetic.

Method	SQA	MQA	Sum.	FS	Syn.	Code	Avg.
Full cache	31.59	24.74	29.25	69.33	50.50	59.76	42.83
L2-norm	12.50	15.23	20.39	51.65	3.25	40.11	24.13
TriAttention	3.05	4.34	4.27	24.08	3.25	19.99	9.61
KeyDiff	15.18	19.52	20.54	55.27	29.25	40.65	29.46
H2O	19.10	19.90	**23.87**	58.72	3.50	44.87	28.84
StreamingLLM	13.36	16.79	18.91	54.56	**50.75**	51.96	32.27
KVzip	15.32	15.07	22.49	57.54	46.50	47.09	32.40
TOVA	19.66	20.42	21.30	62.68	50.00	47.64	35.47
SnapKV	20.67	19.50	21.58	58.85	49.75	53.25	35.49
SnapKV + pooling	**22.31**	**20.98**	21.20	60.55	45.25	53.77	35.82
Q-Trust	21.76	20.70	22.26	**65.72**	49.25	**54.30**	**37.40**
H2O + Ada-KV	18.77	19.22	**24.20**	60.31	3.75	45.50	29.13
TOVA + Ada-KV	19.07	20.94	21.34	63.49	**50.50**	48.34	35.76
SnapKV + Ada-KV	21.13	20.35	21.76	59.66	**50.50**	53.94	36.10
Q-Trust + Ada-KV	**22.09**	**21.03**	22.28	**65.36**	**50.50**	**54.69**	**37.67**

4.4 Needle-in-a-Haystack

We evaluate retrieval of a single sentence from a long context with Needle-in-a-Haystack (NIAH). A 7-digit number is hidden in one sentence of a context built from documents that do not overlap with LongBench, and the question at the end of the prompt asks for it. Each of the 44 (length, depth) pairs, with lengths from 4K to 32K, has 5 prompts. At budget 128 (Figure 4), Q-Trust scores 95.0%, above SnapKV (93.8%) and TOVA (88.8%) but below SnapKV + pooling (97.1%). NIAH asks about a single sentence, so keeping a few more tokens around the selected keys through pooling is enough to raise accuracy. LongBench requires understanding the context, which widening the kept neighborhood alone cannot provide; there Q-Trust is stronger. H2O reaches only 34.0%.

4.5 Combination with Budget Allocation

Ada-KV (Feng et al., 2025) divides the budget among the heads of a layer after the scores are computed, so it applies to any score. As the "+ Ada-KV" rows of Table 2 show, Q-Trust remains best with Ada-KV at 37.67, $+1.57$ points above SnapKV + Ada-KV (36.10; 95% CI $[+1.07,\ +2.09]$). The largest gain is again in Few-shot ($+5.70$ points over SnapKV + Ada-KV), and Q-Trust is also best in Single-doc QA, Multi-doc QA, and Code. Ada-KV raises every method by 0.27–0.61 points and leaves the ranking unchanged.

4.6 Cost

Q-Trust changes only the scores, so its KV cache size and decoding cost match those of the baselines; the prefill scoring alone differs. As Table 3 shows, computing the score in PyTorch adds $+21.5\%$ at 8K and $+34.4\%$ at 16K, which the kernel reduces to $+2.4\%$ and $+3.5\%$; at 32K, the kernel adds $+4.6\%$ against $+54.6\%$ for PyTorch. The kernel matches an fp32 reference within 10^{-6} (bf16 PyTorch: 10^{-3}), and the two implementations select the same keys 99.7–99.8% of the time.

Table 3: Uncompressed prefill time and the time Q-Trust adds (Llama-3.1-8B, 5 prompts, one RTX 3090; †two GPUs).

Length	Prefill	PyTorch	Kernel
4K	0.97 s	$+13.7\%$	$+2.1\%$
8K	2.06 s	$+21.5\%$	$+2.4\%$
16K	4.63 s	$+34.4\%$	$+3.5\%$
32K†	11.54 s	$+54.6\%$	$+4.6\%$

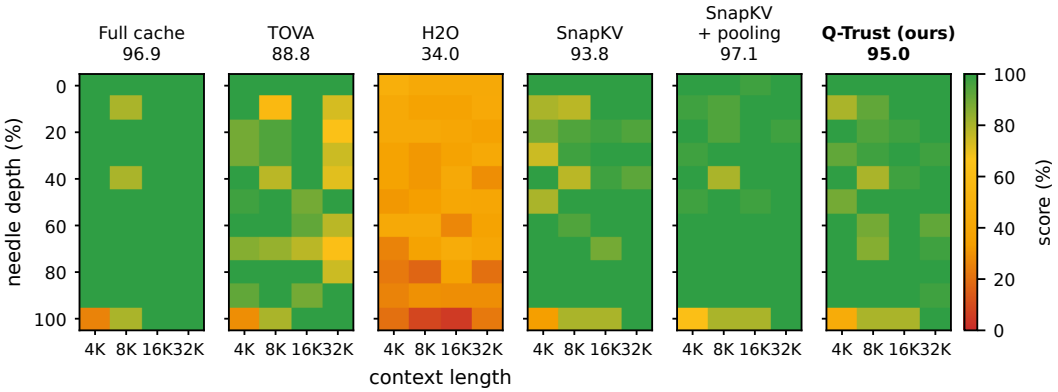

Figure 4: Needle-in-a-Haystack results for Llama-3.1-8B (KV budget 128). Each cell averages 5 prompts; the number under each title is the overall average.

Table 4: LongBench average versus the observation window (Llama-3.1-8B, KV budget 128). SnapKV scores 35.49 and TOVA 35.47.

	Size W							Request span	
Window	4	8	16	32	64	128	256	Last sentence	Full request
LongBench	36.41	37.14	**37.40**	36.53	34.87	34.38	34.09	36.87	35.22

Table 5: Effect of the threshold τ (Llama-3.1-8B, KV budget 128): LongBench average, share of query rows computed (mean over 48 prompts of 8K tokens), and added time relative to uncompressed prefill at 16K tokens (one RTX 3090).

τ	0 (dense)	0.1	0.25	0.5	1	2	4
LongBench	37.76	37.70	37.59	37.57	37.40	37.07	36.82
Rows computed	100%	60.5%	37.1%	22.6%	12.8%	6.8%	3.6%
Added time (16K)	+11.4%	+9.6%	+7.1%	+5.0%	+3.5%	+2.0%	+1.2%

Table 6: LongBench scores by task group versus the temperature T of the query weight (Llama-3.1-8B, KV budget 128, $\tau = 0$). The last row uses the best T for each task.

T	SQA	MQA	Sum.	FS	Syn.	Code	Avg.
1 (Q-Trust)	**23.21**	20.50	22.54	66.21	48.50	**54.88**	37.76
2	23.12	**20.69**	23.48	**66.30**	**49.00**	53.67	**37.88**
4	22.14	20.60	**24.15**	64.45	44.00	50.42	36.43
8	20.91	20.57	24.07	62.11	15.25	48.46	31.90
Best per task	24.07	20.82	24.24	66.43	49.00	54.88	38.40

4.7 Ablation Study

4.7.1 Observation Window Size

The observation window determines which queries in Eq. (4) count as the request (Table 4). For $W = 4$–16, accuracy stays within 36.41–37.40, above every baseline, while for $W \geq 64$ the window extends into the end of the document and accuracy falls to 34.09 at $W = 256$. Among request spans, the last sentence (the question) gives 36.87, close to $W = 16$, while the full request with its instruction gives 35.22; $W = 16$ captures the end of the request without locating its boundary.

4.7.2 THRESHOLD τ

Table 5 reports accuracy and cost as a function of τ. At the default $\tau = 1$, Q-Trust skips 87% of query rows while staying within 0.36 points of dense computation, and the overhead falls from $+11.4\%$ to $+3.5\%$. At $\tau = 4$ it skips 96% of rows with $+1.2\%$ overhead and still exceeds every baseline.

4.7.3 TEMPERATURE OF THE QUERY WEIGHT

Adding a temperature T to the document-query weight of Eq. (6) gives $w(q) \propto s(q)^{1/T}$; $T = 1$ corresponds to Q-Trust, and as $T \to \infty$ the weights approach the uniform accumulation of H2O. As Table 6 shows, the best T differs by task. Summarization, which covers the whole document, favors flat weights (24.15 at $T = 4$), while Single-doc QA, Synthetic, and Code peak at $T = 1$–2. Choosing the best T for each task raises the average to 38.40, 0.64 points above $T = 1$ (37.76), so a lightweight selector that predicts T from the prompt could improve accuracy further.

5 RELATED WORK

Accumulated attention. H2O (Zhang et al., 2023) observes that a few heavy hitters dominate generation and keeps the keys with the largest accumulated attention together with recent tokens. Keyformer (Adnan et al., 2024) accumulates a Gumbel-noised softmax to offset the uneven redistribution of attention after eviction, and RoCo (Ren & Zhu, 2024) averages over the attending queries to remove the bias toward early keys.

Recent queries. Scissorhands (Liu et al., 2023) evicts keys that often receive below-uniform attention from the most recent queries, TOVA (Oren et al., 2024) evicts the token least attended by the current query, and SnapKV (Li et al., 2024), observing that attention in an observation window at the end of the input resembles attention during generation, keeps the keys most attended by that window. NACL (Chen et al., 2024) scores with the question tokens and adds random eviction, and Lookahead Q-Cache (Wang et al., 2025) rescores keys with the queries of a few response tokens generated in advance. Both families fix in advance which queries to trust; Q-Trust decides this for each prompt from its content.

Other scores. Attention-free methods keep attention sinks and a recent window (StreamingLLM (Xiao et al., 2024)), low-norm keys (L2-norm (Devoto et al., 2024)), keys distinct from the mean key (KeyDiff (Park et al., 2025)), or keys predicted important from calibration statistics (TriAttention (Mao et al., 2026)). OBCache (Gu et al., 2026) estimates the change in attention output caused by eviction, including value states, and KVzip (Kim et al., 2025) scores KV pairs by context reconstruction, enabling query-agnostic reuse. Q-Trust instead reweights queries for the given request, using only the attention of prefill.

Budget allocation. Ada-KV (Feng et al., 2025) allocates the budget across the heads of a layer once scores are computed and therefore composes with Q-Trust (§4.5).

Following attention more than once. Attention rollout (Abnar & Zuidema, 2020) multiplies attention across layers to trace information flow, TokenRank (Erel et al., 2025) defines global token importance by the stationary distribution of attention viewed as a Markov chain, and Nexus Sampling (Duong et al., 2026) weights recent queries by their overlap with the accumulated attention in streaming eviction. Our 2-hop attention stays within one layer and follows the attention of document queries weighted by the request.

6 CONCLUSION

We cast attention-based KV cache compression as a weighted sum over queries and showed that only a few informative prefill queries, which position-based weights cannot single out, point to the keys read during generation. Q-Trust identifies them from the attention of the user's request and uses that attention as the query weight of 2-hop attention. Without training, it outperforms existing methods on three models with 3.5% prefill overhead, establishing the query weight as a design axis for KV cache compression.

AI USE STATEMENT

We used generative AI tools for three purposes. First, for the survey of related work, generative AI compiled lists of related papers, and the authors reviewed the papers. Second, for writing, the authors wrote the drafts and generative AI polished the wording. Third, for experiments, the authors wrote the main implementation of the method, and generative AI wrote the experiment scripts that run it. The authors take full responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

Q-Trust requires no training. The scoring rule and its selective computation are defined in §3, and the benchmark, models, baselines, and hyperparameters are given in §4.1. All experiments use public models and datasets with greedy decoding, and Appendices A and B report the results on other models and at other budgets. Upon acceptance, we will release the code, including the Triton kernels and the evaluation scripts.

REFERENCES

Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4190–4197, 2020.

Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant J. Nair, Ilya Soloveychik, and Purushotham Kamath. Keyformer: KV cache reduction through key tokens selection for efficient generative inference. In *Proceedings of Machine Learning and Systems*, 2024.

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.

Yilong Chen, Guoxia Wang, Junyuan Shang, Shiyao Cui, Zhenyu Zhang, Tingwen Liu, Shuohuan Wang, Yu Sun, Dianhai Yu, and Hua Wu. 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)*, pp. 7913–7926, 2024.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, 2024.

Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. A simple and effective l_2 norm-based strategy for KV cache compression. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18476–18499, 2024.

Duc Duong, Hoang Anh Duy Le, Jianwen Xie, Anshumali Shrivastava, and Zhaozhuo Xu. Forget without compromise: Nexus sampling for streaming KV-cache eviction under fixed budgets. *arXiv preprint arXiv:2606.23961*, 2026.

Yotam Erel, Olaf Dünkel, Rishabh Dabral, Vladislav Golyanik, Christian Theobalt, and Amit H. Bermano. Attention (as discrete-time Markov) chains. In *Advances in Neural Information Processing Systems*, 2025.

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. In *Advances in 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.

Yuzhe Gu, Xiyu Liang, Jiaojiao Zhao, and Enmao Diao. OBCache: Optimal brain KV cache pruning for efficient long-context LLM inference. In *International Conference on Machine Learning*, 2026.

Albert Q. 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, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W. Lee, Sangdoo Yun, and Hyun Oh Song. KVzip: Query-agnostic KV cache compression with context reconstruction. In *Advances in Neural Information Processing Systems*, 2025.

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. In *Advances in Neural Information Processing Systems*, volume 37, pp. 22947–22970, 2024.

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. In *Advances in Neural Information Processing Systems*, volume 36, pp. 52342–52364, 2023.

Weian Mao, Xi Lin, Wei Huang, Yuxin Xie, Tianfu Fu, Bohan Zhuang, Song Han, and Yukang Chen. TriAttention: Efficient long reasoning with trigonometric KV compression. *arXiv preprint arXiv:2604.04921*, 2026.

Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. Transformers are multi-state RNNs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18724–18741, 2024.

Junyoung Park, Dalton Jones, Matthew J. Morse, Raghavv Goel, Mingu Lee, and Chris Lott. KeyDiff: Key similarity-based KV cache eviction for long-context LLM inference in resource-constrained environments. In *Advances in Neural Information Processing Systems*, 2025.

Siyu Ren and Kenny Q. Zhu. On the efficacy of eviction policy for key-value constrained generative language model inference. *arXiv preprint arXiv:2402.06262*, 2024.

Philippe Tillet, H. T. Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19, 2019.

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. 34158–34174, 2025.

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

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

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. In *Advances in Neural Information Processing Systems*, 2023.

A Results on Other Models

Figure A.1 shows the LongBench average for Qwen2.5-7B and Mistral-7B-v0.3 versus KV budget. Q-Trust stays above the plotted baselines at every budget on both models. At budget 128, it outperforms SnapKV by $+1.72$ and $+2.56$ points on Qwen2.5-7B and Mistral-7B-v0.3, respectively (95% CIs $[+1.17,\ +2.29]$ and $[+1.93,\ +3.18]$), and SnapKV + pooling by $+0.78$ and $+1.14$ points ($[+0.20,\ +1.39]$ and $[+0.45,\ +1.82]$); gains over SnapKV of $+0.92$ and $+0.88$ points remain at budget 1024. On Mistral-7B-v0.3, Q-Trust at budget 128 (41.62) exceeds the best baseline at budget 256 (41.00). Table A.1 reports scores by task group for the two models at budget 128. As on Llama-3.1-8B, SnapKV + pooling is best in Single-doc QA and Multi-doc QA, and Q-Trust in Few-shot and Code; the largest gain is again in Few-shot, $+4.33$ and $+9.00$ points over SnapKV. On TREC, Q-Trust scores 49.5 and 66.5 against 37.0 and 39.0 for SnapKV.

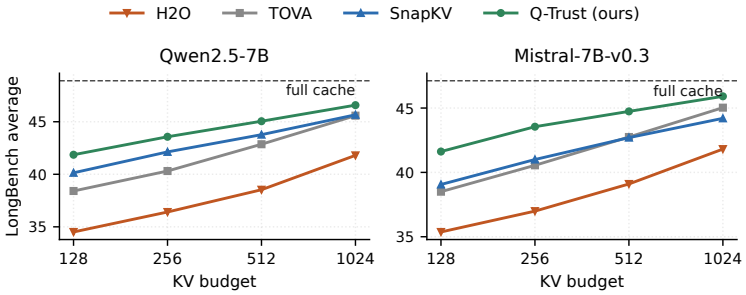

Figure A.1: LongBench average for Qwen2.5-7B and Mistral-7B-v0.3 versus KV budget. Dashed lines show the full cache.

Table A.1: LongBench scores by task group for Qwen2.5-7B-Instruct and Mistral-7B-Instruct-v0.3 at KV budget 128. Abbreviations follow Table 2.

Method	SQA	MQA	Sum.	FS	Syn.	Code	Avg.
Qwen2.5-7B-Instruct							
Full cache	41.90	45.18	26.41	69.00	54.25	63.22	48.90
L2-norm	18.54	28.14	17.12	44.77	14.25	35.34	26.56
TriAttention	18.95	26.60	17.49	42.37	23.00	35.14	27.03
KeyDiff	23.07	29.07	16.51	47.75	16.00	35.87	28.31
H2O	27.88	34.55	**22.06**	55.27	24.00	42.41	34.51
StreamingLLM	23.18	31.44	16.16	54.61	16.50	47.13	31.46
KVzip	26.20	32.31	21.51	51.48	14.25	47.52	32.38
TOVA	28.75	37.34	18.88	54.62	52.50	45.37	38.41
SnapKV	31.18	38.56	19.61	54.69	**53.25**	51.77	40.14
SnapKV + pooling	**34.98**	**40.23**	18.79	55.42	52.75	51.78	41.08
Q-Trust	34.18	40.06	19.79	**59.02**	52.75	**52.55**	**41.86**
Mistral-7B-Instruct-v0.3							
Full cache	38.63	38.66	28.75	70.69	52.00	59.83	47.12
L2-norm	19.96	26.89	19.61	46.04	27.62	36.78	29.14
TriAttention	20.33	25.64	19.49	45.69	16.50	33.90	27.14
KeyDiff	23.30	28.14	20.41	51.55	20.75	36.40	30.28
H2O	28.70	32.90	**23.97**	60.70	21.79	41.79	35.37
StreamingLLM	21.48	27.89	18.16	57.17	27.75	51.34	33.27
KVzip	25.09	30.80	22.79	60.46	20.75	47.32	34.59
TOVA	29.30	33.50	21.65	57.75	47.00	47.72	38.50
SnapKV	28.99	31.85	22.47	57.11	**49.75**	52.12	39.06
SnapKV + pooling	**32.70**	**34.15**	21.51	61.55	45.75	53.26	40.48
Q-Trust	31.23	33.25	22.55	**66.11**	**49.75**	**53.53**	**41.62**

B Results at Larger KV Budgets

Table B.1 reports LongBench scores by task group for the three models at KV budgets 256, 512, and 1024. Abbreviations follow Table 2.

Table B.1: LongBench scores by task group at KV budgets 256, 512, and 1024. The best compressed method in each column is in bold.

Budget	Method	SQA	MQA	Sum.	FS	Syn.	Code	Avg.
Llama-3.1-8B-Instruct								
256	H2O	20.98	21.03	**24.94**	61.87	5.08	48.47	30.85
	TOVA	19.40	19.79	22.12	64.15	**50.75**	50.24	36.15
	SnapKV	23.42	**21.16**	22.77	60.60	50.50	56.21	37.33
	Q-Trust	**24.55**	21.10	23.44	**66.67**	50.50	**56.81**	**38.87**
512	H2O	25.06	**21.99**	**25.92**	63.54	7.25	50.59	32.83
	TOVA	23.15	20.40	23.30	66.16	**50.75**	52.96	37.90
	SnapKV	26.52	21.24	24.25	62.22	50.50	**57.88**	38.71
	Q-Trust	**27.69**	21.76	24.96	**67.12**	50.50	57.86	**40.08**
1024	H2O	26.58	23.08	**27.29**	65.50	17.00	53.86	35.57
	TOVA	26.16	21.39	24.95	**67.91**	**50.75**	55.86	39.65
	SnapKV	29.26	21.90	25.59	63.97	50.50	58.57	40.02
	Q-Trust	**29.67**	**23.37**	26.44	67.46	50.50	**58.95**	**41.23**
Qwen2.5-7B-Instruct								
256	H2O	30.70	36.41	**23.24**	57.28	24.50	45.38	36.41
	TOVA	30.33	38.65	20.36	57.70	53.25	48.67	40.31
	SnapKV	34.23	40.07	21.29	55.75	**54.00**	56.03	42.13
	Q-Trust	**36.40**	**41.05**	21.60	**59.13**	53.75	**57.52**	**43.57**
512	H2O	33.53	37.43	24.17	60.06	25.50	49.93	38.53
	TOVA	33.03	40.20	21.87	**62.86**	53.25	52.66	42.86
	SnapKV	36.74	41.44	22.58	57.31	**54.00**	59.09	43.77
	Q-Trust	**38.96**	**41.99**	23.14	60.51	53.75	**59.75**	**45.05**
1024	H2O	36.92	40.48	24.87	63.73	32.00	53.42	41.80
	TOVA	36.23	42.50	23.58	**67.77**	**54.00**	55.55	45.58
	SnapKV	39.72	**42.88**	24.27	59.85	**54.00**	61.26	45.66
	Q-Trust	**40.97**	42.59	24.53	63.08	**54.00**	**61.84**	**46.58**
Mistral-7B-Instruct-v0.3								
256	H2O	31.16	33.28	**25.17**	62.59	23.14	44.45	36.99
	TOVA	31.30	35.14	22.76	59.51	**51.00**	50.30	40.55
	SnapKV	32.79	33.89	23.25	58.30	**51.00**	54.68	41.00
	Q-Trust	**34.02**	**35.53**	24.12	**67.57**	**51.00**	**55.52**	**43.55**
512	H2O	33.37	35.90	**26.07**	64.98	24.45	47.87	39.10
	TOVA	33.48	**36.46**	24.06	64.26	51.50	53.06	42.75
	SnapKV	34.34	35.47	24.46	60.18	**51.75**	58.16	42.70
	Q-Trust	**35.78**	36.18	25.28	**68.15**	51.50	**58.33**	**44.74**
1024	H2O	35.03	37.10	**27.10**	66.67	33.32	52.33	41.81
	TOVA	36.14	37.39	25.08	**69.95**	**51.75**	55.64	45.03
	SnapKV	35.85	36.35	25.74	63.73	**51.75**	59.37	44.20
	Q-Trust	**37.02**	**37.53**	26.78	69.42	**51.75**	**59.42**	**45.91**