EXPLORING THE BOUNDARY OF SPARSE ATTENTION: RETRIEVABILITY, HEAD STABILITY, AND ANN APPROXIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse attention is a promising way to reduce the cost of long-context inference, but it remains unclear when top-k attention is accurate and whether practical approximate nearest neighbor (ANN) search can replace exact top-k retrieval. We study these two questions separately—whether attention is retrievable by exact top-k, and whether ANN indexes can find the keys it needs—across Llama-3.1-8B and Qwen3-8B on RULER, LongBench, and LongProc. Exact top-k attention often approximates full attention well: across the 13 RULER tasks, the median number of keys needed to cover 90% of the attention mass grows far more slowly than the context, up to a model-specific length frontier. Its effectiveness nonetheless depends strongly on model, task, context length, and attention head. Retrieval-style tasks need significantly fewer keys than aggregation-style tasks, and fixed-budget top-k attention degrades sharply at very long contexts, especially beyond a model's native context window. Head retrievability is structured and stable across inputs and decoding steps (held-out rank correlation ≈ 0.97), and a head-aware budget calibrated once improves task scores by up to 7.3 points over a uniform budget of the same size. However, off-the-shelf ANN indexes such as IVF, HNSW, and PQ do not reliably recover enough attention mass under aggressive search budgets, and their set recall is not a reliable proxy for the mass they retain. These results suggest that sparse attention is useful but not universal: reliable deployment requires model- and task-aware strategies, adaptive budgets, and ANN methods tuned for attention mass rather than recall.

1 INTRODUCTION

During decoding, every attention head compares the current query with every cached key. The cost of this linear scan grows with the context and comes to dominate inference at long context lengths. A natural remedy is to reduce the computation space: retrieve the k keys with the largest attention scores and attend only to them. Top-k attention has been studied as an approximation to dense attention (Gupta et al., 2021; Synk et al., 2025; Haris, 2025), and several systems now implement it with page-level selection (Tang et al., 2024), locality-sensitive hashing (Chen et al., 2025), product quantization (Zhang et al., 2025), or graph indexes (Liu et al., 2025); trainable sparse attention builds such selection into the model itself (Yuan et al., 2025; Lu et al., 2025).

Whether this approach works depends on two conditions that end-to-end evaluations do not separate. First, attention must be **retrievable**: a small set of keys must carry most of the softmax mass, and the size of that set must be predictable enough to budget for. Second, attention must be **searchable**: a sublinear index must actually find that set. When a retrieval-based method loses accuracy, it is therefore unclear which condition failed—whether attention was not concentrated to begin with, or whether the index missed the keys that mattered. This paper maps the boundary of sparse attention along both conditions and measures them separately: exact top-k attention serves as an oracle for the first, and approximate indexes are compared against this oracle for the second.

We study Llama-3.1-8B-Instruct and Qwen3-8B on all 13 RULER tasks at five context lengths from 4K to 128K tokens, on LongBench, and on LongProc, and organize the study around three research questions (Figure 1):

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

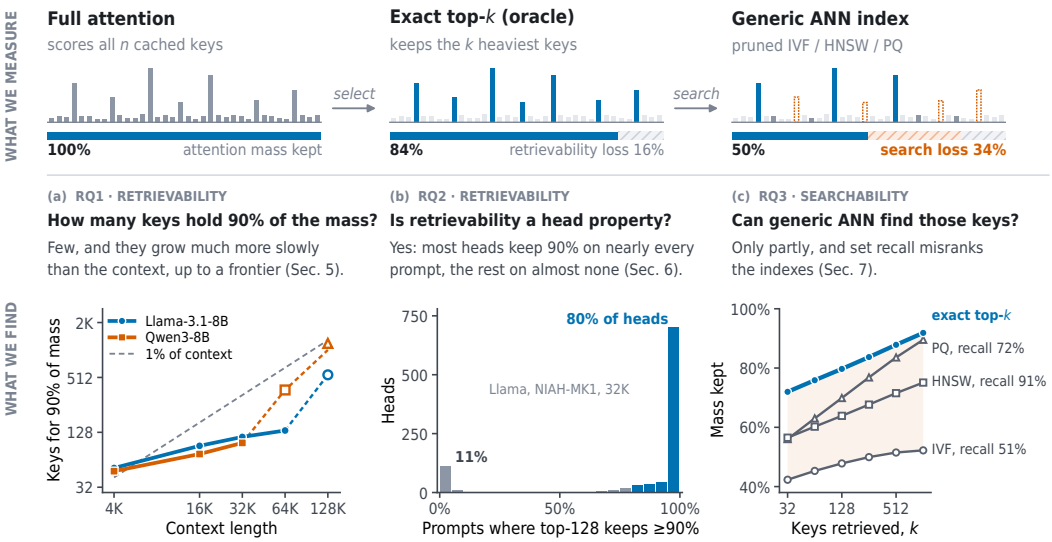


Figure 1: **What we measure and what we find.** *Top:* retrieval-based attention loses mass when it keeps only k keys (retrievability) and when an index must find them (searchability); we measure both against exact top-k (strips schematic; mass bars measured on LongBench, Llama-3.1-8B, k=256). *Bottom:* our three questions, answered by measurements: (a) median number of keys covering 90% of the mass over 13 RULER tasks (hollow: beyond the length frontier; △: over 1024); (b) Llama heads on NIAH-MK1 at 32K, by the share of 500 prompts on which top-128 keeps 90%; (c) exact top-k vs. three off-the-shelf indexes on the same queries, with set recall at k=256.

RQ1 How many keys does attention need, and how does this scale with context length? (Section 5)

RQ2 Is retrievability a stable property of attention heads, so that a budget can be set before the attention weights are seen? (Section 6)

RQ3 Can generic ANN indexes find the keys that attention needs? (Section 7)

Our answers to these questions are the main contributions of this paper.

A1: Attention needs few keys, and their number grows much more slowly than the context. Across RULER tasks, a small set of keys carries most of the attention mass, and enlarging the context sixteenfold, from 4K to 64K tokens, less than triples the median number of keys needed to cover 90% of it, This trend holds up to a *length frontier* near the longest context the model was trained on or extended to; beyond it, attention becomes diffuse and the required set grows sharply (Section 5).

A2: Retrievability is a stable property of individual heads. Whether a head concentrates its attention depends far more on the head than on the input: most heads concentrate on nearly every prompt and the others on almost none, a head's coverage barely changes during generation, and a ranking of heads measured on the first decoding steps of a few prompts carries over to later steps on new prompts (Section 6). Head-level budgets can therefore be calibrated once, before generation.

A3: Generic ANN indexes recover only part of the attention mass, and recall does not reveal how much. Under pruned search, off-the-shelf IVF, HNSW and PQ indexes keep roughly half of the attention mass, well below exact top-k at the same k, and recovering most of the rest requires searching a large part of the cache. Set recall, the metric such indexes are usually tuned for, can rank them in the wrong order, because it counts keys while attention weights them (Section 7). This finding does not rule ANN indexes out from sparse attention, instead, it measures the gap that new attention-specific indexes can be optimized for.

We do not propose a new sparse-attention method. Instead, we measure both conditions with a single quantity, the attention mass that a set of keys retains, using exact top-k as the oracle (Section 3), and translate the answers into design guidelines that we partly test with generation experiments (Section 8).

2 RELATED WORK

Sparse and retrieval-based attention. Fixed sparsity patterns (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020) and content-based routing (Kitaev et al., 2020; Roy et al., 2021) reduce attention cost at training time. For inference, KV-cache eviction keeps heavy or recent tokens (Zhang et al., 2023; Xiao et al., 2024; Li et al., 2024), while query-aware methods retrieve keys per step (Tang et al., 2024; Singhania et al., 2024), using LSH sampling (Chen et al., 2025), product quantization (Zhang et al., 2025), or attention-aware graph indexes that address query–key distribution mismatch (Liu et al., 2025). Adaptive budgets replace fixed k with top-p or distribution-based thresholds (Lin et al., 2025; Zhu et al., 2026), sampling complements top-k with error guarantees (Desai et al., 2026), and range search guarantees that no key above a threshold is missed (Dehghankar & Asudeh, 2026). Our measurements help explain why such attention-specific designs are needed: the required set is small but not constant (Section 5), and set recall is an unreliable proxy for retained mass (Section 7).

Heads and budgets. Retrieval heads (Wu et al., 2025) and the split into retrieval and streaming heads (Xiao et al., 2025; Tang et al., 2025) motivate head-specific policies; adaptive budget allocation across heads (Feng et al., 2024) and offline pattern assignment (Jiang et al., 2024) assume that head behavior can be calibrated in advance. We test this assumption directly—across inputs, decoding steps, held-out prompts and context lengths—and identify where it fails (Section 6).

Analyses of sparse attention. Nawrot et al. (2026) benchmark training-free sparse-attention methods end to end and find that longer sequences tolerate higher sparsity. Rather than evaluating methods, we characterize attention itself: we separate whether the needed keys exist (retrievability, via exact top-k) from whether an index can find them (searchability), which explains their length trend through the sublinear growth of the required budget and locates where it breaks.

3 MEASURING RETRIEVABILITY AND SEARCHABILITY

Both conditions can be measured with a single quantity: the attention mass that a set of keys retains.

Top-k attention as retrieval. At a decoding step, a head with query q attends over n cached keys and values with weights $\alpha_j = \mathrm{softmax}_j(q^\top k_j/\sqrt{d})$ and output $o = \sum_j \alpha_j v_j$.A sparse method selects a set S of $|S| = k$ keys and computes $o_S = \sum_{j\in S} \alpha_j v_j / \sum_{j\in S} \alpha_j$. We define the *coverage* of S as the attention mass it retains, $\mathrm{C}(S) = \sum_{j\in S} \alpha_j \in [0, 1]$, and report it as a percentage throughout, so that 90% coverage means $\mathrm{C}(S) = 0.9$. Exact top-k selection $S_k^\star$ maximizes coverage among all sets of size k, so $\mathrm{C}(S_k^\star)$ upper-bounds what any method selecting k keys can retain. Coverage also controls the output error of any selection, exact or approximate (proof in Appendix A):

Proposition 1. *For any S with $\mathrm{C}(S) > 0$,* $\|o - o_S\| \le 2\,\big(1 - \mathrm{C}(S)\big)\,\max_j \|v_j\|$.

In practice, coverage also tracks the actual attention output across selection mechanisms: over 3,456 index configurations on LongBench, including exact top-k, it predicts the relative output error far better than set recall does.[1] Our evidence for task quality is narrower: on Qwen3-8B CWE at 16K and 32K, raising exact top-k coverage from 83–85% to 91–92% raises the task score by 6–7 points, to the level of dense attention, but the coverage needed for dense quality differs across settings (Section 8). We therefore use coverage as the single metric for both conditions.

Retrievability. Let $\mathrm{C}_{p,\ell,h,t}(k)$ be the exact top-k coverage of head h in layer ℓ at decoding step t of prompt p. We average it over steps and then uniformly over prompts, layers and heads to obtain the coverage curve $\bar{\mathrm{C}}(k)$, and summarize how many keys attention needs by $k^\star_{90} = \min\{k : \bar{\mathrm{C}}(k) \ge 0.9\}$, interpolated log-linearly between measured budgets. Because $k^\star_{90}$ thresholds the *average* curve, it is the budget at which heads keep 90% of their mass on average, not an average of per-head budgets, which are heavy-tailed (Section 5). We call a head *stably retrievable* if its step-averaged top-128 coverage reaches 90% on at least 80% of prompts, and measure temporal stability by the coefficient of variation (CV) of its coverage across steps. Retrievability measures only how concentrated a head's attention is; unlike the retrieval heads of Wu et al. (2025), retrievability in this paper says nothing about what the head attends to.

[1] Spearman $\rho = -0.98$ vs. -0.56; Figure 17 in Appendix F.

Searchability. An index returns a set S_I of k keys. We report its coverage $\mathrm{C}(S_I)$ and its set recall $|S_I \cap S_k^\star|/k$. Recall weights all top-k keys equally, whereas coverage weights them by attention; the two diverge when an index misses a few heavy keys or many light ones.

4 EXPERIMENTAL SETUP

Models and benchmarks. We use Llama-3.1-8B-Instruct (Grattafiori et al., 2024) and Qwen3-8B (Yang et al., 2025). Both have 32 query and 8 KV heads of dimension 128; Llama has 32 layers and supports a 128K-token context window; Qwen has 36 layers and a 32K native window. We run both models with their released configurations at every length, so Qwen's 64K and 128K runs lie beyond its native window. RULER (Hsieh et al., 2024) is our main benchmark because it controls both context length and task type. We use all 13 tasks—eight needle-in-a-haystack variants (NIAH), variable tracking, common- and frequent-word extraction (CWE, FWE), and two QA tasks (Appendix B)—at 4K, 16K, 32K, 64K and 128K tokens, with 500 prompts per task up to 32K and 50 per task at 64K and 128K. LongBench (Bai et al., 2024) adds natural documents: 18 tasks spanning QA, summarization, few-shot learning, synthetic tasks and code, with 20 prompts per task and inputs truncated to 64K tokens for Llama and 40K for Qwen. LongProc (Ye et al., 2025) adds long procedural outputs: on its Countdown task the models generate 800–900 tokens, which lets us test whether head-level behavior persists over long generations. The generation experiments in Section 8 run five RULER tasks (Qwen at 16K and 32K, Llama at 64K and 128K), each with 20 calibration and 100 held-out test prompts (Appendix B).

Queries and budgets. For every prompt we decode greedily for up to 64 tokens and analyze the queries of the generated tokens (the full generation for LongProc), since 64 tokens are enough for most queries to allow a complete generation without truncating. At each decoding step, layer and query head, we recompute the attention distribution in fp32 from the post-RoPE queries and keys (after Qwen3's per-head QK normalization); under grouped-query attention each query head uses the keys and values of its KV group. We evaluate exact top-k selection at $k \in \{32, 64, 128, 256, 512, 1024\}$. Coverage is aggregated as defined in Section 3. We treat the first four positions as attention sinks.

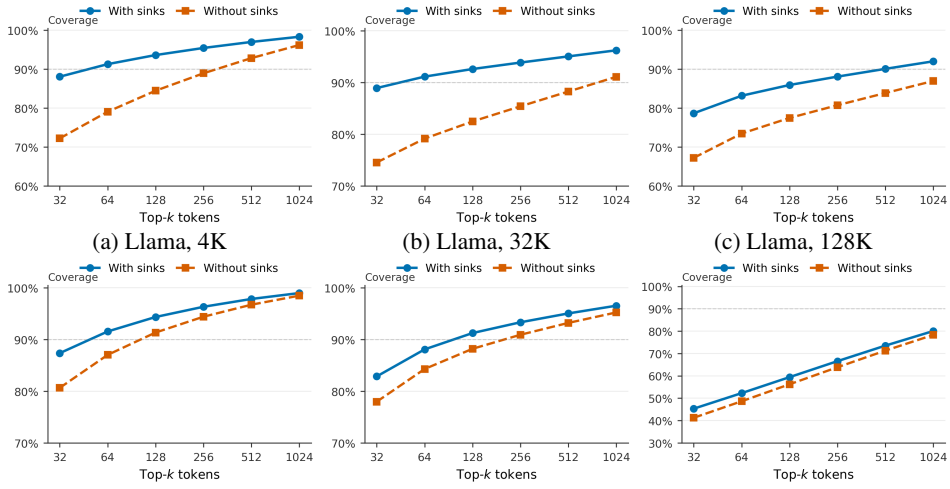
Head-level statistics. Per-head statistics use exact top-128 coverage (Section 3). For the layer-wise breakdown in Appendix D, we call a (head, prompt) pair *concentrated* if its top-128 coverage is at least 90%, *diffuse* if even its top-512 coverage stays below 50%, and *mixed* otherwise; these categories describe concentration only, not the functional retrieval and streaming heads of prior work. For the held-out test, we estimate per-head rankings on decoding steps 0–15 of calibration prompts and evaluate them on steps 16–63 of disjoint prompts, for 18 LongBench tasks and both models (36 conditions).

Approximate search. All indexes use FAISS (Douze et al., 2024) with the inner-product metric over the post-RoPE keys visible to each query head, and coverage is measured against the full attention of the same queries. On RULER NIAH-S1, which we use to vary context length and model, we fix each family at the lowest-effort setting of its grid—the regime in which retrieval would save the most computation: IVF probing 4 of 64 lists, HNSW with M=16 and ef_{search}=1024, and flat PQ with m=16. On LongBench (Llama, 18 tasks, layers 0/16/31, 10 prompts) we sweep a grid per index family and use exhaustive IVF on the same queries as the exact reference (Appendix B).

5 RQ1: HOW MANY KEYS DOES ATTENTION NEED?

We answer RQ1 with exact top-k selection, which retains the largest attention mass that any k keys can hold. Attention turns out to need few keys, and their number grows much more slowly than the context—but only up to a length frontier, beyond which attention becomes diffuse.

A few dozen keys carry most of the mass. At 4K tokens, exact top-32 attention covers a median of 87% of the attention mass across the 13 RULER tasks for both models, and top-128 covers 94–95% (Figure 2; Appendix C). A large part of this mass sits on the attention sinks (Xiao et al., 2024): on LongBench, the first four tokens receive a median of 55% (Llama) and 30% (Qwen) of the decode-time attention (Appendix D). Sinks are cheap to keep because their positions are fixed,



(a) Llama, 4K (b) Llama, 32K (c) Llama, 128K

(d) Qwen, 4K (e) Qwen, 32K (f) Qwen, 128K

Figure 2: **Exact top-k coverage on RULER NIAH-MK1.** Solid: coverage of the full attention distribution; dashed: coverage within the non-sink part, i.e., after removing the first four tokens and renormalizing the remaining weights (a conditional quantity, so the gap between the curves is not the sink mass). Coverage at a fixed k barely moves from 4K to 32K for Llama but drops at 128K; for Qwen it already declines at 32K and collapses at 128K. Other tasks and lengths are in Appendix C.

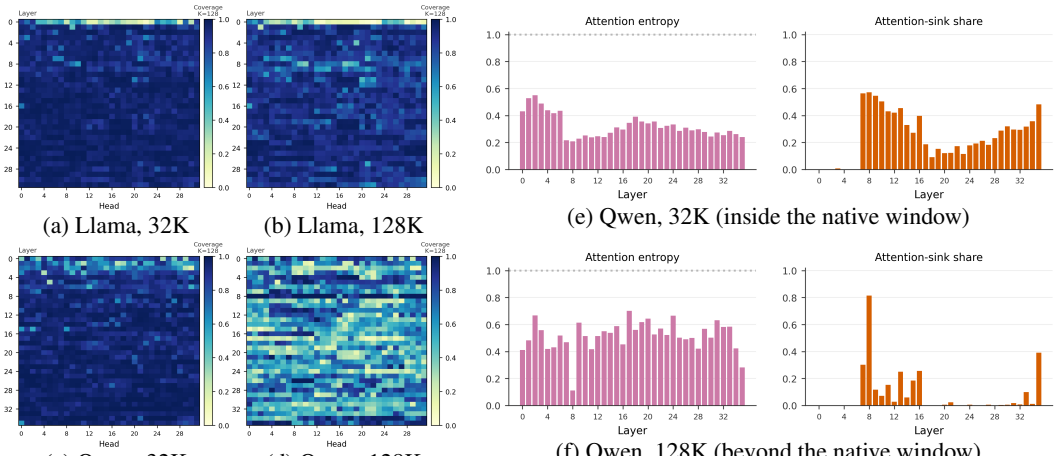
Table 1: **Keys needed to cover 90% of the attention mass** ($k^{\star}_{90}$, exact top-k) on RULER, for three representative tasks and the median over all 13 tasks. Llama-3.1 at 128K and Qwen3 beyond 32K lie past the model's length frontier; >1024 means that 90% is not reached with 1024 keys. Appendix C lists every task.

	Llama-3.1					Qwen3				
Task	4K	16K	32K	64K	128K	4K	16K	32K	64K	128K
NIAH-MK1	46	43	43	56	465	48	71	94	334	>1024
CWE	109	163	97	60	260	298	592	844	>1024	>1024
QA-1	41	91	123	126	562	44	75	171	463	>1024
Median	52	91	114	134	547	48	74	98	373	>1024

so what matters for retrieval is the rest of the mass. It is also concentrated, though less so: after removing the sinks and renormalizing, top-128 still covers a median of 76% (Llama) and 86% (Qwen) of the non-sink mass on LongBench, and on RULER the non-sink curves approach the full ones as k grows (Figure 2, dashed). Content tokens thus need larger budgets than the full-distribution coverage suggests.

The required set grows much more slowly than the context. Table 1 tracks $k^{\star}_{90}$ as the context grows. For Llama, the median over tasks rises from 52 keys at 4K to 134 keys at 64K. Fitting a power law $k^{\star}_{90} \propto n^{\beta}$ to the medians between 4K and 64K gives $\beta \approx 0.35$, far below the linear growth ($\beta = 1$) implied by a budget proportional to the context; the share of the cache that attention needs accordingly falls from 1.3% at 4K to 0.2% at 64K. Qwen follows the same trend inside its native window (48 to 98 keys from 4K to 32K). Neither common budget rule matches this behavior: a fixed k gradually loses mass as the context grows, whereas a fixed share of the context overspends—at 64K, a 1% budget retrieves 655 keys where 134 suffice. The median hides substantial heterogeneity. Individual tasks follow different trajectories: for Llama, NIAH-S1 stays at 40–46 keys from 4K to 64K, and CWE even falls from 163 to 60 keys between 16K and 64K. Aggregation-style CWE needs 7–9 times more keys than NIAH-S1 for Qwen inside its native window, but only 1.5–3.5 times more for Llama,[2] consistent with the task sensitivity reported for end-to-end sparse attention (Nawrot et al., 2026). Heads vary as well: on LongBench, the per-head budget for 90% coverage has a median of 47 keys for Llama but a 90th percentile of about 630, and 6% of heads do not reach 90% within 1024 keys (Appendix D). A budget of $k^{\star}_{90}$ therefore serves the typical head, not every head.

[2]Per-task values for both models and all lengths: Table 2 in Appendix C.



(a) Llama, 32K (b) Llama, 128K (e) Qwen, 32K (inside the native window)

(c) Qwen, 32K (d) Qwen, 128K (f) Qwen, 128K (beyond the native window)

Figure 3: **Beyond the length frontier, attention becomes diffuse.** (a–d) Per-head top-128 coverage on NIAH-MK1 (rows: layers; columns: heads). Low-coverage heads cluster in a few layers; beyond the frontier, Llama loses concentration in some heads and Qwen in most layers. (e, f) Per-layer normalized attention entropy (left; 1 = uniformly distributed, smaller entropy means concentrated) and attention share of the first four tokens (right) for Qwen3-8B. From 32K to 128K, entropy rises in most layers and the sink share vanishes outside a few layers: the mass spreads out rather than moving to other keys. Llama shows the same trend, more weakly (Appendix D).

Retrievability breaks at the length frontier. The favorable scaling holds only within the lengths a model handles well. Llama's median $k^\star_{90}$ quadruples from 64K to 128K, the edge of its supported window. Qwen degrades as soon as it runs beyond its 32K native window: at 64K the median $k^\star_{90}$ is 373, and at 128K no task reaches 90% coverage with 1024 keys (median coverage at k=1024 is 81%). Figure 3a–d shows where the mass goes. At 32K, almost every head of both models is concentrated; at 128K, Llama loses concentration in a subset of heads, while Qwen, four times beyond its native window, loses it across most layers. In Qwen at 128K the normalized attention entropy rises in most layers and the attention-sink share collapses (Figure 3e,f); Llama shows the same trend, more weakly, at the edge of its supported window (Appendix D). In both models attention becomes diffuse rather than shifting to other keys. This is consistent with softmax attention flattening as the number of keys grows (Nakanishi, 2025); we treat the frontier as an empirical boundary, not a causal claim.

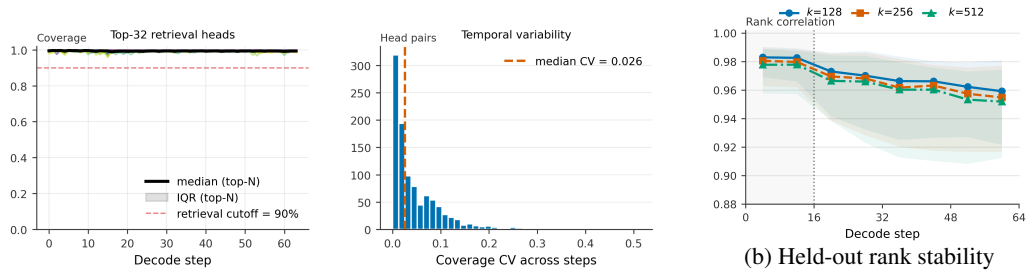
6 RQ2: Is retrievability a stable property of heads?

A retrieval system must decide, per head, how many keys to fetch before it sees the attention weights. This is only possible if retrievability is predictable from information available ahead of time. We test predictability across inputs, across decoding steps, and on held-out prompts, and find that, within the lengths a model handles well, retrievability behaves as a property of the head rather than of the input.

Across inputs. Retrievability is close to binary at the head level. On NIAH-MK1, 83% of Llama's heads at 4K and 80% at 32K are stably retrievable (top-128 coverage $\geq 90\%$ on at least 80% of 500 prompts); the per-head distribution is bimodal, with the remaining heads failing on almost every prompt rather than on some (Appendix D). Qwen shows the same structure inside its window (84% at 4K, 72% at 32K). The fraction falls at the frontier—to 38% for Llama at 128K and to 51% and 9% for Qwen at 64K and 128K—mirroring Table 1.

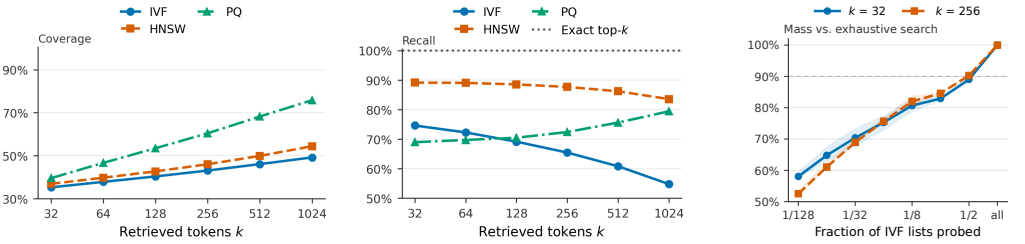
Across decoding steps. Within a prompt, a head's coverage barely changes as generation proceeds (Figure 4a). The median CV of per-head coverage across decoding steps is 0.026 for Llama on CWE at 32K, and 0.024–0.030 on NIAH-MK1 for both models at 32K and for Llama at 128K; it doubles to 0.060 only for Qwen at 128K. The set of retrievable heads also persists, with a Jaccard overlap with the first step of mostly 0.7–0.8,[3] but random sets of the same size would already overlap by about 0.55, so we rely on the stricter rank-based test below.

[3]Figure 13 in Appendix D.



(a) Within-prompt stability (Llama, CWE, 32K) (b) Held-out rank stability (LongBench)

Figure 4: **Retrievability persists across decoding steps.** (a) Left: coverage of the 32 most retrievable heads over 64 decoding steps; right: coefficient of variation of each head's coverage across steps (all heads). (b) Spearman correlation between per-head retrievability rankings estimated on steps 0–15 of calibration prompts and measured on later steps of disjoint prompts; lines: median over 36 model–task conditions; bands: 10th–90th percentiles.



(a) Coverage (Llama, 32K) (b) Recall (Llama, 32K) (c) Mass vs. search effort (IVF)

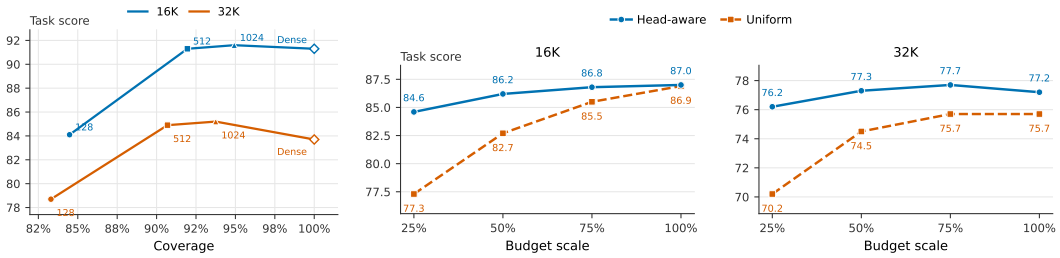
Figure 5: **Generic ANN indexes recover only part of the attention mass.** (a, b) IVF (4 of 64 lists), HNSW (M=16, ef=1024) and PQ (m=16) on RULER NIAH-S1: recall ranks HNSW first, coverage ranks PQ first. (c) Attention mass retained by IVF relative to exhaustive search on the same queries, as a function of the fraction of lists probed (best n_{list} per fraction; 18 LongBench tasks, Llama; bands: interquartile range over tasks).

On held-out prompts. The strongest test is whether a ranking estimated once transfers to new prompts and later steps. We rank heads by retrievability on steps 0–15 of calibration prompts and evaluate on steps 16–63 of disjoint prompts across 18 LongBench tasks and both models. The median Spearman correlation is 0.971–0.974 for $k \in \{128, 256, 512\}$ and remains about 0.95 in the final step bin (Figure 4b). On LongProc, the most retrievable heads keep near-full coverage for 800–900 generated tokens, but Qwen's overall top-1024 coverage stays near 60% (Appendix E): stability is not sufficiency.

7 RQ3: Can generic ANN indexes find the keys?

We now replace exact top-k selection with approximate search over three index families on which retrieval-based attention systems build: IVF, HNSW (Malkov & Yashunin, 2020) and product quantization (Jégou et al., 2011). We use them off the shelf, without the attention-specific adaptations such systems add—e.g., query-aware graph construction (Liu et al., 2025), sampling-based estimation (Chen et al., 2025) or adaptive budgets (Lin et al., 2025)—so as to measure how far generic search is from exact top-k, i.e., how much these adaptations must recover. On RULER NIAH-S1 we compare the indexes at their lowest-effort settings across context lengths and models; on LongBench we sweep each family's grid and compare every setting against exhaustive search over the same queries (Section 4). We find that they recover only part of the mass, and that recall does not reveal how much.

Under pruning, generic indexes retain about half of the mass. At k=256, the three indexes retain between 36% and 81% of the attention mass across models and lengths, and most settings sit near 50% (Figure 5a; Appendix F). IVF is the weakest: probing 4 lists caps the candidate pool, so its coverage grows only slowly with k. When the probed lists hold fewer than k keys (about 256 keys at 4K with 4 of 64 lists), IVF returns fewer than k keys; we count the missing keys as lost mass and as misses for recall (Appendix B). On LongBench, where the exact reference uses the same queries and



(a) Coverage vs. task score (b) Head-aware vs. uniform allocation at matched budgets

Figure 6: **Retained mass translates into quality, and head structure can be exploited.** Qwen3-8B, RULER CWE, 100 held-out prompts per setting. (a) Exact top-k (labels give k) against dense attention (open diamonds); in this setting, top-k reaches the dense score above 90% coverage. (b) A per-head budget calibrated once on separate data and frozen, versus a uniform split of the same layer-wise, step-wise total budget; budget scales rescale the calibrated policy. The two panels use different test prompts (Appendix B).

layers, exact top-256 keeps 84% of the mass, against 50% for IVF, 68% for HNSW and 77% for PQ with the same configurations (Figure 1c).

Recall can misrank indexes. The metric that ANN indexes are usually tuned for does not reliably predict the metric attention cares about. On Llama at 32K and k=256, HNSW recovers 88% of the exact top-k keys but retains 47% of the mass, whereas PQ recovers only 73% of the keys yet retains 61% (Figure 5a,b). On Qwen, HNSW's recall exceeds PQ's by 13–25 points at every length, while their coverage stays within two points. Across all 3,456 configurations of the LongBench grid, coverage tracks the relative attention-output error almost monotonically ($\rho = -0.98$), and at matched coverage IVF, HNSW and PQ have nearly the same error; recall is a much weaker predictor ($\rho = -0.56$).[4] Recall counts keys, whereas attention weights them: an index can miss the few heavy keys at high recall, or only the light tail at low recall.

Keeping most of the mass requires broad search. To measure the cost of recovering the mass, we compare each IVF configuration with exhaustive search over the same queries, samples and layers (Figure 5c). Probing half of the lists retains only 89–90% of the exhaustive-search mass; probing one eighth retains 81–82%; probing 1/128 retains 52–58%. In our grid, every configuration that retains at least 95% of the mass scores every key: exhaustive IVF, or flat PQ with 64 sub-quantizers, which scores every compressed key. The best graph configuration (HNSW, ef=4096) retains about 94% at ten times the CPU latency of the exhaustive IVF scan. Attention needs a fraction of a percent of the cache (Section 5), yet here a generic index must touch much of the cache to find it.

Scope. We compare indexes at the same number of requested keys, not at matched search cost: IVF and HNSW prune the search, whereas flat PQ scores every compressed key. Our ANN results cover one RULER task, three layers and ten prompts per length, plus the LongBench grid for Llama; latencies are single-query CPU times. We therefore claim a gap in retained mass for generic indexes at a given number of returned keys—not a limitation of retrieval-based attention systems in general, and not an end-to-end efficiency comparison. We also do not test hybrids that always keep the sink tokens and search only the rest.

8 Implications for sparse-attention design

The answers to RQ1–RQ3 suggest applying top-k attention selectively. We derive five guidelines and test G2–G4 on generation quality, using exact top-k under a logical token budget and making no claim about speed:

- **G1: Do not transfer a fixed sparse strategy across models and tasks.** Tasks differ in both the level and the trend of their budgets (Appendix C), and the two models differ in which heads concentrate and where their length frontier lies (Figure 3); a strategy tuned for one model or task should be re-checked before reuse.

[4] Both relations are plotted in Figure 17 in Appendix F.

- **G2: Make the budget length-aware.** A fixed k loses mass as the context grows, and the loss accelerates near and beyond the model's native or training length, where a percentage-based or mass-targeted budget is safer. Below the frontier, $k^{\star}_{90}$ grows sublinearly, so a fixed percentage overspends; calibrating the budget per length, or retrieving up to a mass target, avoids both.
- **G3: Allocate the budget by head.** Heads concentrated on nearly every prompt can use small budgets. Diffuse heads (top-512 coverage below 50%) need much larger budgets, with dense attention as a conservative fallback; mixed heads need larger or adaptive budgets (Appendix D).
- **G4: Calibrate once.** Head retrievability is stable across inputs, decoding steps and held-out prompts (Section 6), so a head-level policy can be calibrated on a few prompts and early steps and reused throughout generation.
- **G5: Build and evaluate indexes for mass, not recall.** Indexes should retrieve up to a mass target rather than a fixed k, as threshold- and range-based methods do (Lin et al., 2025; Dehghankar & Asudeh, 2026), and be tuned on retained mass rather than recall (Section 7); handling query–key distribution mismatch (Liu et al., 2025), which we also observe (Appendix G), can help recover heavy keys.

Coverage tracks generation quality (G2). Qwen3-8B decodes RULER CWE at 16K and 32K with exact top-k, $k \in \{128, 512, 1024\}$, on 100 held-out prompts per setting (Figure 6a). Raising k from 128 to 512 increases coverage from 84.5% to 92.0% at 16K and from 83.3% to 90.7% at 32K, and task scores recover to the dense level, from 84.1 to 91.3 and from 78.7 to 84.9. Over all seven fixed-k settings we ran, a larger k raises coverage and lowers the mean loss in six; in the seventh (Llama CWE, 64K) even dense attention scores 4.4. The coverage needed to match dense attention varies, however: Llama FWE at 64K still loses 7.7 points at 90% coverage (Appendix H). These coverages are measured on each policy's own sparse trajectory (Appendix B) and are not directly comparable with the profiling runs of Section 5.[5] A fixed k thus lands at different coverage and quality at different lengths (G2).

Calibrate once, allocate by head (G3, G4). The head-aware policy is calibrated on 20 separate prompts with dense decoding: each head receives the smallest $k \in \{64, 128, \ldots, 4096\}$ whose step-averaged coverage reaches 90% on at least 80% of these prompts, or all visible keys if no k does. The frozen budgets select about 5% of the visible keys on Qwen CWE (roughly 800 keys per head at 16K, 1,800 at 32K). Scaled policies multiply every budget by 25–75% and round up; a uniform policy splits the same layer-wise, step-wise total evenly across heads (Appendix B). At the 25% scale, head-aware allocation improves the CWE score from 77.3 to 84.6 at 16K (+7.3, 95% CI [4.8, 9.8]) and from 70.2 to 76.2 at 32K (+6.0 [3.2, 8.9]); at 50%, by +3.5 [1.3, 5.8] and +2.8 [0.5, 5.4] (Figure 6b); at 75% and 100% the intervals include zero. The gain is largest when budgets are tight: beyond Qwen CWE, it also appears for Llama QA-2 at 128K (+4.0 [1.0, 8.0] at the 25% scale), and on the other tasks, several of which are near their score ceiling, head-aware allocation performs on par with uniform in most settings, with a few exceptions. The calibrated assignment itself carries the benefit: permuting the calibrated budgets among a layer's heads lowers the score significantly in most settings we tested, by 7.6–7.8 points on Qwen CWE and 15.6–16.6 on Llama FWE (Appendix H). Because the policy is frozen, this tests G3 and G4 together, though not whether each head's budget still meets its target on new prompts.

9 Limitations and conclusion

Limitations. We study two 8B models, which cannot disentangle the causes of the length frontier. Samples at 64K and 128K are 50 prompts per task.

Conclusion. Attention is retrievable: a few dozen to a few hundred keys carry 90% of the mass, a number that grows far more slowly than the context until the length frontier (RQ1), and retrievability is a stable property of each head (RQ2). Finding these keys is harder: off-the-shelf indexes keep only part of the mass under pruned search, and recall can misrank them (RQ3). Progress therefore depends on length- and head-aware budgets and on indexes built to find attention mass.

[5] Tables 2 and 3 in Appendix C; Appendix B compares the two protocols.

AI USE STATEMENT

Large language model assistants were used for LaTeX template. All experiments, analyses and scientific claims were designed, run and verified by the authors.

REPRODUCIBILITY STATEMENT

Sections 3–4 and Appendix B specify models, benchmarks, sample counts, decoding protocol, budgets, head-level thresholds and index configurations, including the full ANN search grid. We will release the analysis code, per-condition summaries and plotting scripts.

REFERENCES

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. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 3119–3137, 2024.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. MagicPIG: LSH sampling for efficient LLM generation. In *International Conference on Learning Representations*, 2025.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Mohsen Dehghankar and Abolfazl Asudeh. Sparse attention as a range searching problem: Towards an inference-efficient index for KV cache. *arXiv preprint arXiv:2605.06763*, 2026.

Aditya Desai et al. vAttention: Verified sparse attention via sampling. In *International Conference on Learning Representations*, 2026.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The Faiss library. *arXiv preprint arXiv:2401.08281*, 2024.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-KV: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference. *arXiv preprint arXiv:2407.11550*, 2024.

Aaron Grattafiori et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Ankit Gupta, Guy Dar, Shaya Goodman, David Ciprut, and Jonathan Berant. Memory-efficient transformers via top-k attention. In *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*, pp. 39–52, 2021.

Themistoklis Haris. $k$nn attention demystified: A theoretical exploration for scalable transformers. In *International Conference on Learning Representations*, 2025.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011. doi: 10.1109/TPAMI.2010.57.

Huiqiang Jiang et al. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. In *Advances in Neural Information Processing Systems*, 2024.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

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*, 2024.

Chaofan Lin et al. Twilight: Adaptive attention sparsity with hierarchical top-p pruning. In *Advances in Neural Information Processing Systems*, 2025.

Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, Chen Chen, Fan Yang, Yuqing Yang, and Lili Qiu. RetrievalAttention: Accelerating long-context LLM inference via vector retrieval. In *Advances in Neural Information Processing Systems*, 2025.

Enzhe Lu et al. MoBA: Mixture of block attention for long-context LLMs. *arXiv preprint arXiv:2502.13189*, 2025.

Yu. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2020. doi: 10.1109/TPAMI.2018.2889473.

Ken M. Nakanishi. Scalable-softmax is superior for attention. *arXiv preprint arXiv:2501.19399*, 2025.

Piotr Nawrot, Robert Li, Renjie Huang, Sebastian Ruder, Kelly Marchisio, and Edoardo M. Ponti. The sparse frontier: Sparse attention trade-offs in transformer LLMs. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

Prajwal Singhania, Siddharth Singh, Shwai He, Soheil Feizi, and Abhinav Bhatele. Loki: Low-rank keys for efficient sparse attention. In *Advances in Neural Information Processing Systems*, 2024.

Ryan Synk, Monte Hoover, John Kirchenbauer, Neel Jain, Alex Stein, Manli Shu, Josue Melendez Sanchez, Ramani Duraiswami, and Tom Goldstein. Exploiting sparsity for long context inference: Million token contexts on commodity GPUs. *arXiv preprint arXiv:2502.06766*, 2025.

Hanlin Tang et al. RazorAttention: Efficient KV cache compression through retrieval heads. In *International Conference on Learning Representations*, 2025.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context LLM inference. In *International Conference on Machine Learning*, 2024.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. Retrieval head mechanistically explains long-context factuality. In *International Conference on Learning Representations*, 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.

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

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Xi Ye, Fangcong Yin, Yinghui He, Joie Zhang, Howard Yen, Tianyu Gao, Greg Durrett, and Danqi Chen. LongProc: Benchmarking long-context language models on long procedural generation. In *Conference on Language Modeling*, 2025.

Jingyang Yuan et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big Bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33, pp. 17283–17297, 2020.

Hailin Zhang et al. PQCache: Product quantization-based KVCache for long context LLM inference. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2025.

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. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, 2023.

Kan Zhu et al. Tactic: Adaptive sparse attention with clustering and distribution fitting for long-context LLMs. In *International Conference on Learning Representations*, 2026.

A Proof of Proposition 1

Let $C = \mathrm{C}(S)$ and $M = \max_j \|v_j\|$. Since $o_S = C^{-1} \sum_{j \in S} \alpha_j v_j$,

$$o - o_S = \sum_{j \notin S} \alpha_j v_j - \frac{1 - C}{C} \sum_{j \in S} \alpha_j v_j.$$

By the triangle inequality, the first term has norm at most $M \sum_{j \notin S} \alpha_j = (1 - C)M$, and the second at most $\frac{1-C}{C} M \sum_{j \in S} \alpha_j = (1 - C)M$. Hence $\|o - o_S\| \leq 2(1 - C)M$. The first term alone is the error of the unnormalized estimate $\sum_{j \in S} \alpha_j v_j$, which therefore satisfies the tighter bound $(1 - C)M$. Both bounds hold for any selection rule, so it applies equally to exact top-k and to keys returned by an approximate index. □

B Experimental details

Section 4 describes models, benchmarks, decoding protocol and head-level thresholds. We add the remaining details here.

Tasks. RULER: NIAH single S1–S3, multi-key MK1–MK3, multi-value MV and multi-query MQ; variable tracking (VT); common- and frequent-word extraction (CWE, FWE); QA-1 and QA-2.

Generation experiments. The experiments of Section 8 decode greedily in bfloat16 (Qwen3 with thinking disabled). Each prompt is the RULER chat prompt followed by the task's answer prefix, and CWE allows up to 512 new tokens. Every setting has 20 calibration and 100 test prompts, disjoint from each other and from a few development prompts. The fixed-k and shuffled experiments share one test set; the budget-scaling experiments use a new one generated with a different RULER seed, so dense scores differ between Figure 6a and b (e.g., 91.3 vs. 86.8 on Qwen CWE at 16K). The prompt is prefilled with dense attention, and sparse attention starts at the query that predicts the first answer token. From then on, each layer and query head attends only to the keys chosen by exact top-k over all visible keys: the prompt, including the first four positions, and the tokens generated so far; no position is reserved. The budget counts selected positions per query head; query heads that share a KV group may select different keys, all query–key scores are still computed, and the full cache is kept, so budgets do not translate into memory or bandwidth savings.

Coverage in generation. Coverage in Figure 6 and Appendix H is measured during these runs: for every generated token, layer and head, it is the mass of the selected keys under the full attention distribution of that query, averaged over layers, heads, generated tokens and test prompts. Each policy therefore follows its own trajectory. These values differ from the coverages of Section 5, which follow the dense trajectory over the first 64 generated tokens of other prompts (for Qwen CWE at k=128: 84.5% and 83.3% at 16K and 32K in the generation runs, against 81.6% and 79.7% in Table 3). Scores use RULER's metric, the percentage of reference items found in the output, and all intervals are paired bootstrap 95% confidence intervals over the 100 test prompts (2,000 resamples).

Head-aware budgets. Calibration decodes the 20 calibration prompts with dense attention and computes, for every layer and query head, top-k coverage at $k \in \{64, 128, 256, 512, 1024, 2048, 4096\}$, averaged over the generated tokens of each prompt. A head's budget k_h is the smallest k at which at least 80% of the calibration prompts reach 90% coverage; a head that misses this target at k=4096 is marked *full* and reads all visible keys. The per-layer, per-head table is frozen before any test prompt is scored and is calibrated separately for every model, task and length. At scale $\lambda \in \{0.25, 0.5, 0.75\}$, head h receives $\max(1, \lceil \lambda k_h \rceil)$ keys at every layer and step, where k_h of a full head is the current number of visible keys. The uniform policy at the same scale divides the head-aware total of each layer and step evenly across the 32 query heads, up to an integer remainder, and the implementation checks that the totals match at every layer and step. The shuffled control permutes the calibrated budgets among the heads of each layer with three seeds, fixed along the trajectory; its scores are averaged over seeds within each prompt before bootstrapping. Averaged over layers, heads and steps, the unscaled policy selects 4.9–5.7% of the visible keys on Qwen CWE and up to 15.2% on Llama FWE at 128K (Table 7).

Implementation. Models are loaded with Hugging Face Transformers in bfloat16; attention distributions are recomputed in fp32. Both models use their released configurations at every length. For Qwen3-8B this is a 40,960-position window without RoPE scaling, so its 64K and 128K runs extend beyond the positions it is configured for; LongBench inputs for Qwen are truncated to 40K tokens. For Llama NIAH-S1 at 64K we also ran all 500 prompts; the result matches the 50-prompt run.

Sink ablation. For the dashed curves in Figure 2 we drop the first four positions, renormalize the remaining weights to sum to one, and take top-k over the rest. The resulting quantity is the coverage of the non-sink attention distribution; the non-sink coverage in the original distribution equals it times $1 - s$, where s is the sink mass of the same query.

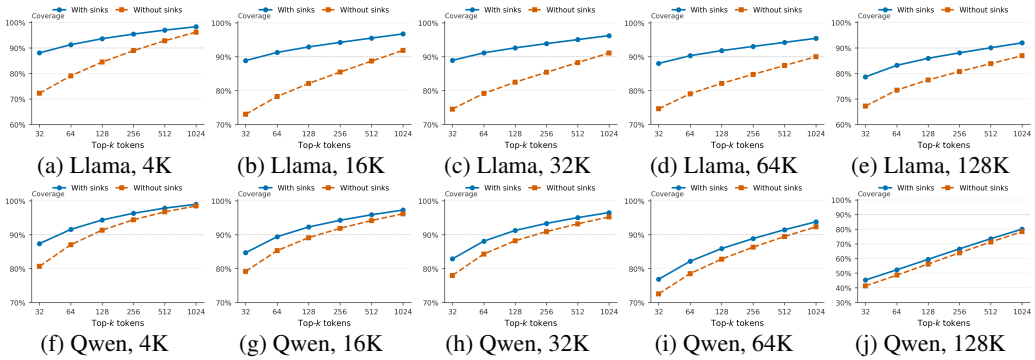
ANN outputs. Each index is rebuilt for every query position over the keys visible to that query. FAISS pads its output with -1 when fewer than k candidates are found; we drop these entries and score only the valid keys, so an index that returns fewer than k keys loses the corresponding mass. Recall uses k as the denominator. The number of valid keys per query was not logged. On RULER, each prompt contributes 8 evenly spaced decoding queries.

ANN indexes. HNSW uses $ef_{\text{construction}}$=200 throughout. On LongBench (Llama, 18 tasks, layers 0/16/31, 10 prompts) we sweep IVF $n_{\text{list}} \in \{64, 128, 256, 512\}$ and $n_{\text{probe}} \in \{4, 8, 16, 32, 64\}$; HNSW $M \in \{16, 32, 48\}$ and $ef_{\text{search}} \in \{1024, 2048, 4096\}$; and PQ $m \in \{16, 32, 64\}$. Latencies are single-query CPU times per head and exclude index construction.

C Full RULER results

Table 2: $k^{\star}_{90}$ (keys needed for 90% attention mass, exact top-k) for every RULER task, model and length. Values are interpolated from the measured budgets.

	Llama-3.1					Qwen3				
Task	4K	16K	32K	64K	128K	4K	16K	32K	64K	128K
NIAH-S1	42	46	45	40	>1024	44	67	98	236	>1024
NIAH-S2	41	40	55	62	492	46	66	92	373	>1024
NIAH-S3	37	45	51	62	349	43	63	85	358	>1024
NIAH-MK1	46	43	43	56	465	48	71	94	334	>1024
NIAH-MK2	68	123	114	217	784	258	700	>1024	>1024	>1024
NIAH-MK3	52	94	167	210	547	55	337	475	866	>1024
NIAH-MV	62	107	130	184	590	48	71	95	350	>1024
NIAH-MQ	61	93	110	134	480	52	74	96	378	>1024
VT	52	56	123	184	655	58	79	99	220	>1024
CWE	109	163	97	60	260	298	592	844	>1024	>1024
FWE	55	88	126	140	488	36	42	58	192	>1024
QA-1	41	91	123	126	562	44	75	171	463	>1024
QA-2	57	105	123	178	670	55	125	189	481	>1024



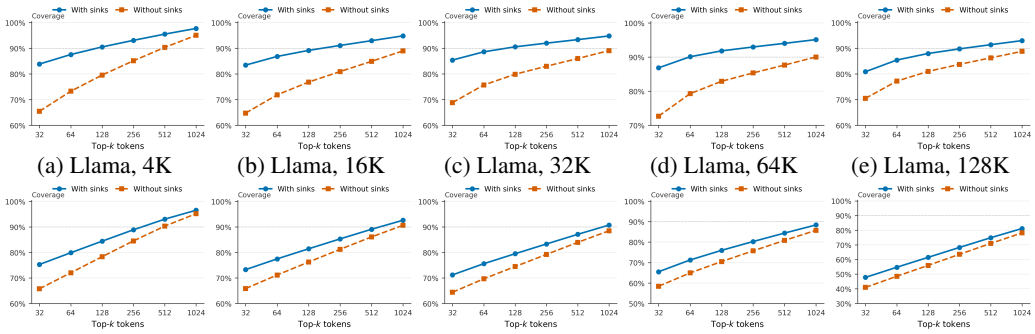
(a) Llama, 4K (b) Llama, 16K (c) Llama, 32K (d) Llama, 64K (e) Llama, 128K

(f) Qwen, 4K (g) Qwen, 16K (h) Qwen, 32K (i) Qwen, 64K (j) Qwen, 128K

Figure 7: Exact top-k coverage on NIAH-MK1 across all context lengths (solid: with sinks; dashed: sinks masked).

Table 3: Exact top-128 coverage (%) for every RULER task, model and length, measured on the dense decoding trajectory over the first 64 generated tokens. The generation experiments of Section 8 measure coverage differently, on each policy's own sparse trajectory and on other prompts (Appendix B), so their values at the same k differ from this table.

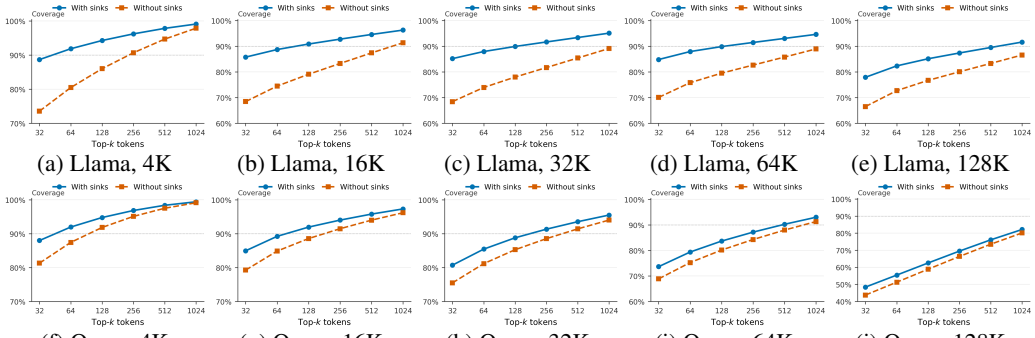
	Llama-3.1					Qwen3				
Task	4K	16K	32K	64K	128K	4K	16K	32K	64K	128K
NIAH-S1	94.5	93.0	92.8	92.6	82.0	94.9	92.4	91.0	88.0	67.2
NIAH-S2	94.0	93.0	92.0	91.5	85.9	94.7	92.6	91.5	85.6	60.3
NIAH-S3	94.4	92.8	92.1	91.6	87.2	95.0	92.9	91.8	85.6	59.9
NIAH-MK1	93.8	93.0	92.7	92.0	86.1	94.5	92.4	91.4	86.1	59.9
NIAH-MK2	92.3	90.1	90.3	88.8	84.3	85.2	79.2	76.8	72.8	60.5
NIAH-MK3	93.6	91.0	89.2	88.7	85.5	94.3	85.2	83.5	79.4	60.7
NIAH-MV	93.0	90.6	90.0	89.1	84.8	94.6	92.4	91.4	85.7	60.0
NIAH-MQ	93.2	91.2	90.5	89.9	85.6	94.2	92.3	91.4	85.1	59.9
VT	94.3	93.2	90.1	89.1	84.1	93.8	92.0	91.1	88.0	68.6
CWE	90.7	89.3	90.8	92.0	88.1	84.6	81.6	79.7	76.2	61.8
FWE	93.3	91.2	90.0	89.8	86.4	95.8	94.9	93.2	88.6	69.0
QA-1	94.4	91.1	90.1	90.0	85.3	94.9	92.1	88.9	83.9	62.9
QA-2	93.2	90.6	90.1	89.2	84.8	93.9	90.1	88.6	83.7	63.5



(a) Llama, 4K (b) Llama, 16K (c) Llama, 32K (d) Llama, 64K (e) Llama, 128K

(f) Qwen, 4K (g) Qwen, 16K (h) Qwen, 32K (i) Qwen, 64K (j) Qwen, 128K

Figure 8: Exact top-k coverage on CWE, an aggregation task, across all context lengths.



(a) Llama, 4K (b) Llama, 16K (c) Llama, 32K (d) Llama, 64K (e) Llama, 128K

(f) Qwen, 4K (g) Qwen, 16K (h) Qwen, 32K (i) Qwen, 64K (j) Qwen, 128K

Figure 9: Exact top-k coverage on QA-1 across all context lengths.

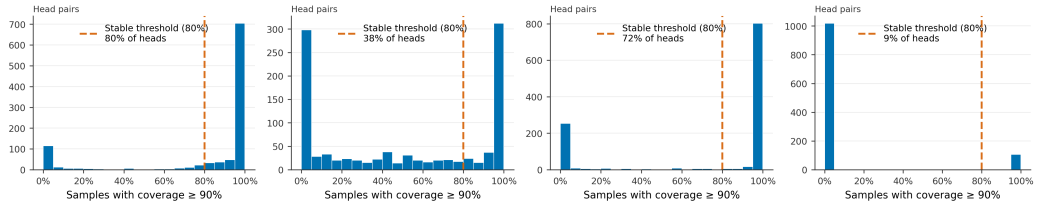
D HEAD-LEVEL ANALYSES

Figure 10 shows the per-head stability histograms behind the stable-head fractions in Section 6: within the lengths a model handles well the distribution is bimodal, whereas for Qwen at 128K it collapses to heads that almost never reach 90% coverage. Figure 11 breaks (head, prompt) pairs into concentrated, mixed and diffuse types by layer. Figure 12 shows normalized attention entropy and the share of attention on the first four tokens per layer; for Qwen at 128K, entropy rises in almost every layer and the sink share disappears outside a few layers. Figure 13 tracks the set of retrievable heads over 64 decoding steps and shows the coverage curve at individual steps; the dips in the Jaccard overlap coincide with a shrinking set, i.e., heads leave the set rather than being replaced by new ones.

Table 4 summarizes the attention sinks and the per-head budgets on LongBench, computed from the per-condition summaries of the LongBench runs (18 tasks, 20 prompts each, all layers and heads). Sinks receive about half of Llama's decode-time attention and about a third of Qwen's. The per-head budget for 90% coverage is heavy-tailed: most heads need fewer keys than the mean-curve $k^{\star}_{90}$, but a quarter need more than 200 keys and about 6% do not reach 90% within 1024 keys.

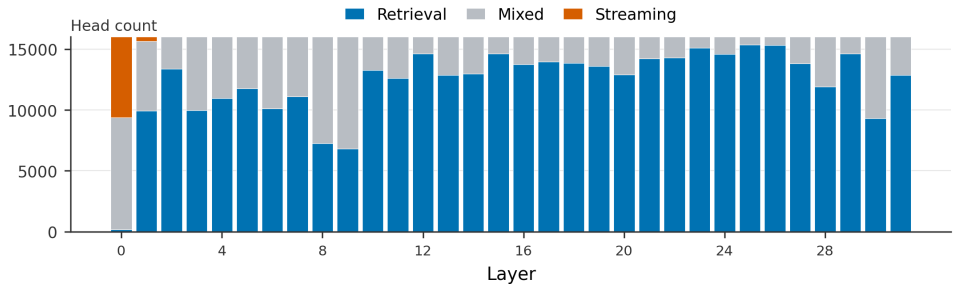
Table 4: Attention sinks and per-head budgets on LongBench (decode stage). Sink mass: attention on the first four tokens (median over tasks). Non-sink coverage: top-128 coverage of the renormalized non-sink distribution. Per-head $k^{\star}_{90}$: the 90% budget of each head's prompt-averaged coverage curve, pooled over tasks, layers and heads.

	Sink	Top-128 coverage		Mean-curve	Per-head $k^{\star}_{90}$			Heads
Model	mass	all	non-sink	$k^{\star}_{90}$	P50	P75	P90	>1024
Llama-3.1	55.0%	90.2%	75.8%	121	47	217	627	5.8%
Qwen3	30.3%	89.7%	86.0%	136	61	225	676	6.0%

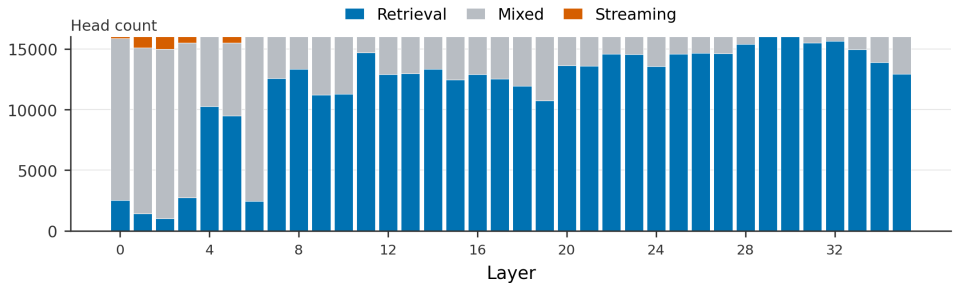


(a) Llama, 32K (b) Llama, 128K (c) Qwen, 32K (d) Qwen, 128K

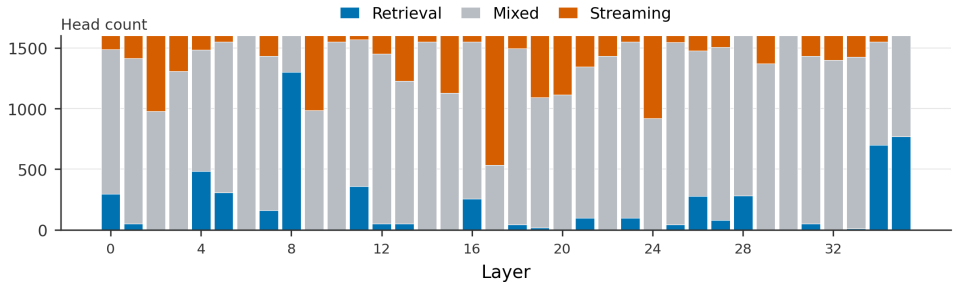
Figure 10: Cross-prompt stability of per-head retrievability (NIAH-MK1): number of (layer, head) pairs by the fraction of prompts on which top-128 coverage reaches 90%.



(a) Llama, 32K

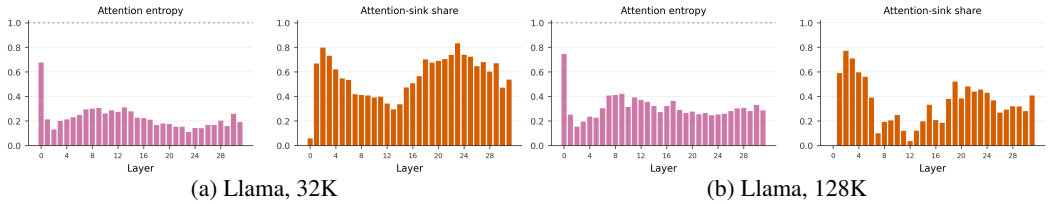


(b) Qwen, 32K



(c) Qwen, 128K

Figure 11: Head types by layer on NIAH-MK1, counted over (head, prompt) pairs. The plot legends use the original labels: "retrieval" denotes concentrated pairs and "streaming" denotes diffuse pairs (Section 4).



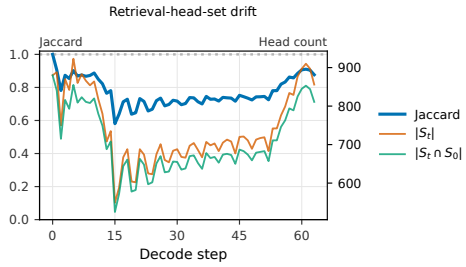
(a) Llama, 32K

(b) Llama, 128K

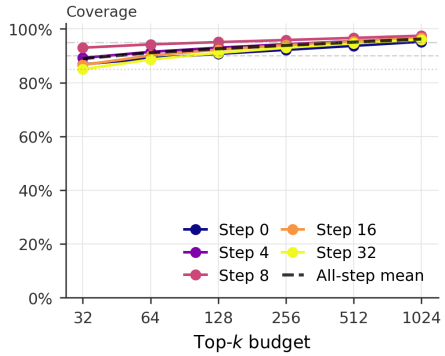
(c) Qwen, 32K

(d) Qwen, 128K

Figure 12: Normalized attention entropy (left of each pair) and attention share on the first four tokens (right) per layer, NIAH-MK1.



(a) Retrievable-head set over 64 steps (Llama, CWE, 32K)



(b) Coverage at individual steps (Llama, NIAH-MK1, 32K)

Figure 13: (a) Jaccard overlap between the retrievable-head set at each step and at step 0, with set sizes. Because most heads are retrievable, random sets of the same size would overlap by about 0.55. (b) Exact top-k coverage at selected decoding steps.

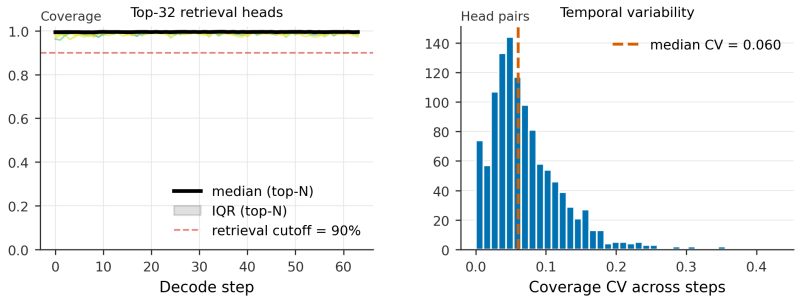
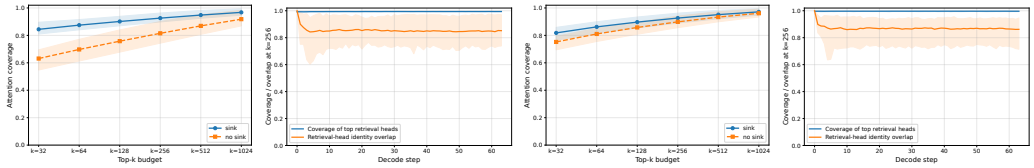


Figure 14: Temporal stability for Qwen at 128K (NIAH-MK1): coverage of the most retrievable heads and CV across steps (median 0.060), about twice the value inside the native window.

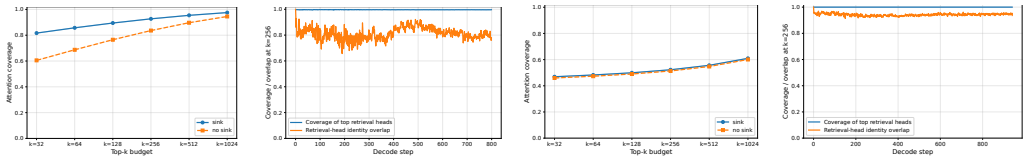
E LONGBENCH AND LONGPROC

On LongBench (18 tasks, contexts truncated at 64K for Llama and 40K for Qwen), exact top-k coverage follows the same pattern as on RULER inside the models' windows, and the most retrievable heads keep near-full coverage across 64 decoding steps (Figure 15). On LongProc Countdown, where outputs reach 800–900 tokens, the most retrievable heads of both models stay near full coverage for the whole generation, while Qwen's overall coverage remains near 60% even at k=1024 (Figure 16).



(a) Llama coverage (b) Llama decoding (c) Qwen coverage (d) Qwen decoding

Figure 15: LongBench, averaged over tasks. Coverage panels: exact top-k coverage with and without sinks (bands: range over tasks). Decoding panels: coverage of the most retrievable heads (blue) and overlap of the retrievable-head set with step 0 (orange) over 64 steps.



(a) Llama coverage (b) Llama decoding (c) Qwen coverage (d) Qwen decoding

Figure 16: LongProc Countdown (long outputs). Panels as in Figure 15, over 800–900 generated tokens.

F APPROXIMATE SEARCH: FULL RESULTS

Table 5: LongBench ANN grid (Llama-3.1, 18 tasks, layers 0/16/31). "Mass" is attention mass retained relative to exhaustive IVF on the same queries (%); "Rec." is set recall against exact top-k (%); latency is ms per head and query at k=32 (CPU). For IVF we list the best n_{list} at each probed fraction.

			$k = 32$		$k = 256$		
Index	Configuration	Probed	Mass	Rec.	Mass	Rec.	ms
IVF	n_{list}=64, n_{probe}=64	1	100.0	99.9	100.0	99.9	0.48
IVF	n_{list}=128, n_{probe}=64	0.5	89.0	97.9	90.3	95.0	0.28
IVF	n_{list}=256, n_{probe}=64	0.25	82.9	95.8	84.6	87.6	0.19
IVF	n_{list}=512, n_{probe}=64	0.125	80.7	94.2	82.0	78.6	0.15
IVF	n_{list}=512, n_{probe}=32	0.0625	75.3	90.1	75.7	63.7	0.11
IVF	n_{list}=512, n_{probe}=16	0.0312	70.4	84.0	68.9	47.9	0.09
IVF	n_{list}=512, n_{probe}=8	0.0156	64.8	75.1	61.0	33.0	0.08
IVF	n_{list}=512, n_{probe}=4	0.00781	58.0	63.0	52.5	21.0	0.07
HNSW	M=48, ef=4096		93.7	98.4	94.4	97.5	5.10
HNSW	M=32, ef=4096		93.6	98.1	94.3	97.2	5.03
HNSW	M=16, ef=4096		92.5	96.8	93.1	95.8	4.88
HNSW	M=48, ef=2048		87.2	97.3	88.7	95.4	1.91
HNSW	M=32, ef=2048		87.0	97.1	88.5	95.2	1.88
HNSW	M=16, ef=2048		85.9	95.9	87.3	94.0	1.80
HNSW	M=48, ef=1024		79.8	95.9	82.0	92.2	0.80
HNSW	M=32, ef=1024		79.4	95.7	81.7	92.0	0.78
HNSW	M=16, ef=1024		78.3	94.6	80.6	91.0	0.74
PQ	m=64, 8 bits		99.7	93.3	99.9	94.5	0.37
PQ	m=32, 8 bits		92.4	81.3	98.4	83.2	0.19
PQ	m=16, 8 bits		77.5	69.6	91.6	72.4	0.12

Table 5 makes the recall–mass mismatch explicit: at k=256, PQ with m=32 recovers only 83% of the exact top-k keys yet retains 98% of the mass, whereas IVF probing half of its lists recovers 95%

of the keys but retains 90% of the mass. Figure 17 relates both metrics to the actual attention output over all 3,456 configurations of the grid (18 tasks, six budgets, all index configurations): coverage predicts the relative output error almost monotonically, and at matched coverage the three index families have nearly the same error, whereas recall is a weak predictor. Figures 18 and 19 report coverage and recall of the fixed-budget configurations on RULER NIAH-S1 at every length.

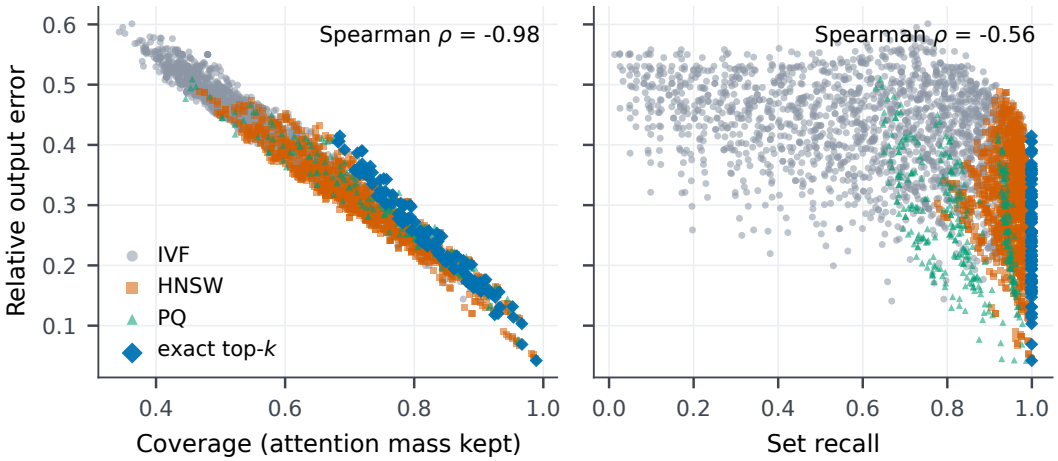
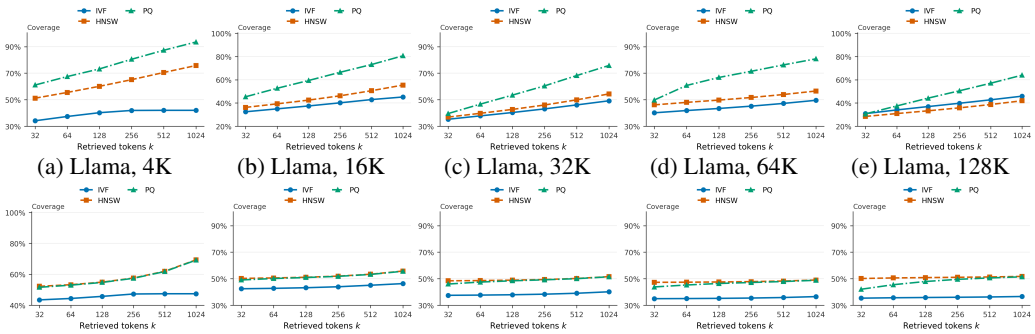


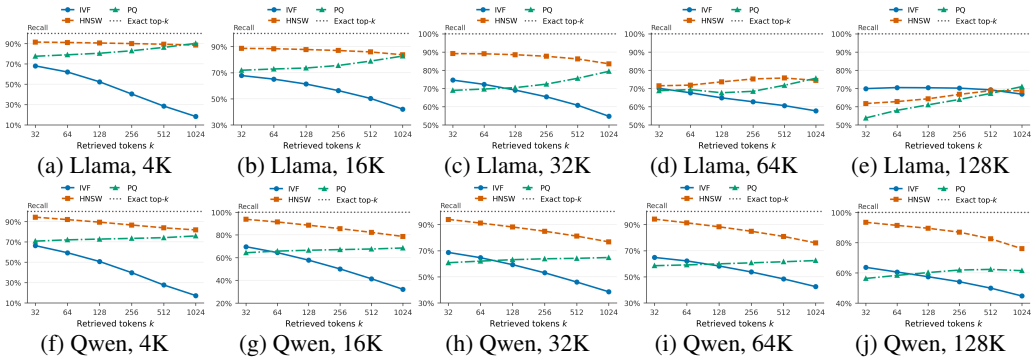
Figure 17: Coverage, not set recall, predicts the attention-output error. Each point is one (task, k, index configuration) of the LongBench grid (Llama-3.1, layers 0/16/31, same queries for all points); y is the mean relative error $\| \sum_{j \in S} \alpha_j v_j - o \| / \| o \|$ of the returned set S. Exact top-k is exhaustive IVF search.



(a) Llama, 4K (b) Llama, 16K (c) Llama, 32K (d) Llama, 64K (e) Llama, 128K

(f) Qwen, 4K (g) Qwen, 16K (h) Qwen, 32K (i) Qwen, 64K (j) Qwen, 128K

Figure 18: Coverage of IVF, HNSW and PQ at fixed pruning configurations on RULER NIAH-S1.



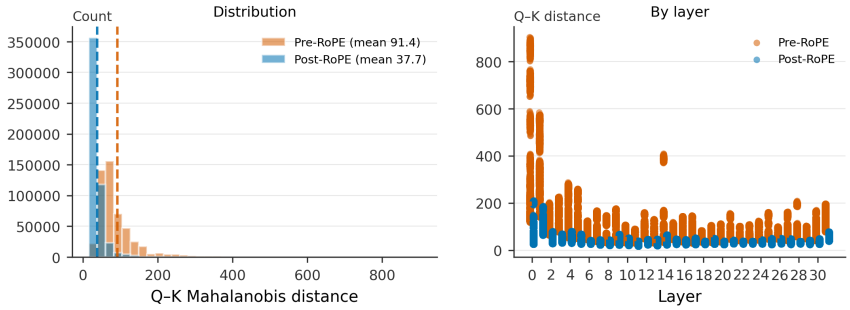
(a) Llama, 4K (b) Llama, 16K (c) Llama, 32K (d) Llama, 64K (e) Llama, 128K

(f) Qwen, 4K (g) Qwen, 16K (h) Qwen, 32K (i) Qwen, 64K (j) Qwen, 128K

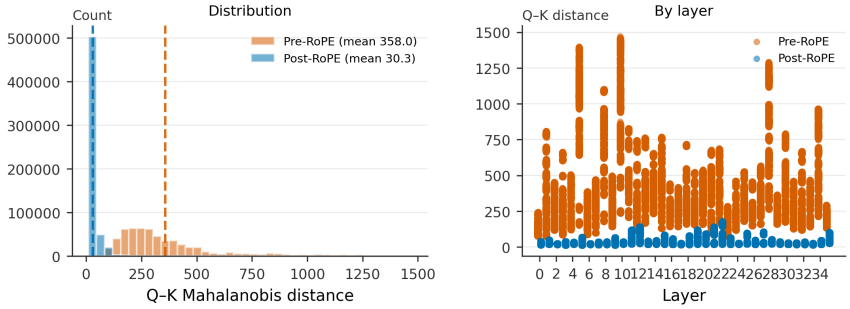
Figure 19: Set recall against exact top-k for the same configurations.

G QUERY–KEY DISTRIBUTION MISMATCH

Retrieval-based attention searches an index built on keys with a query vector, although queries and keys follow different distributions (Liu et al., 2025). Figure 20 shows the Mahalanobis distance of queries to the key distribution before and after RoPE. RoPE reduces the measured distance, especially in early layers. Because RoPE also widens the key distribution by rotating keys at different positions, part of this reduction reflects a larger key covariance rather than better alignment; in either case attention scores are defined in the post-RoPE space, which is where all our indexes operate.



(a) Llama, NIAH-MK1, 32K



(b) Qwen, NIAH-MK1, 32K

Figure 20: Query-to-key Mahalanobis distance before and after RoPE: overall distribution (left) and by layer (right).

H GENERATION EXPERIMENTS: ADDITIONAL RESULTS

Tables 6 and 7 report the fixed-k and full-budget allocation experiments of Section 8. Quality loss is the dense score minus the sparse score in points, so negative values mean that the sparse run scored higher. Bold intervals exclude zero. Protocols are given in Appendix B.

Coverage and quality. Within each fixed-k setting, a larger k raises coverage and, except for Llama CWE at 64K, where dense attention itself scores 4.4, lowers the mean loss (Table 6). The coverage needed to match dense attention depends on the setting: Qwen stays within 1.3 points of dense once coverage reaches about 90%, whereas Llama FWE still loses 7.7 points at 90.1% coverage (64K) and 13.3 points at 86.1% (128K).

Allocation policies. At the full calibrated budget, permuting the calibrated budgets among the heads of each layer lowers the score significantly in five of seven settings, whereas head-aware and uniform allocation differ by at most 2.8 points (Table 7). Under budget scaling, the head-aware gain is largest on Qwen CWE (Figure 6b) and task-dependent elsewhere. Mean coverage does not rank these policies: on Qwen CWE at the 25% scale, uniform allocation retains more mean coverage than head-aware allocation (86.5% vs. 83.3% at 16K; 89.3% vs. 83.2% at 32K) yet scores 6–7 points lower. A mean over heads weights every head equally, so it compares budgets within one allocation policy better than it compares policies.

Table 6: Exact top-k in generation: dense score, and coverage (%) and quality loss (points) at each k. 100 test prompts per setting.

		k=128		k=512		k=1024	
Setting	Dense	Cov.	Loss	Cov.	Loss	Cov.	Loss
Qwen CWE 16K	91.3	84.5	7.2	92.0	0.0	94.9	−0.3
Qwen CWE 32K	83.7	83.3	5.0	90.7	−1.2	93.8	−1.5
Qwen FWE 16K	96.7	85.4	2.3	92.3	1.3	95.0	0.3
Qwen FWE 32K	90.3	82.4	3.0	89.7	1.3	92.7	0.0
Llama CWE 64K	4.4	79.1	0.0	87.5	0.3	91.0	1.9
Llama FWE 64K	80.3	78.5	26.0	86.7	14.7	90.1	7.7
Llama FWE 128K	81.0	68.9	25.7	81.2	19.0	86.1	13.3

Table 7: Allocation at the full calibrated budget, on the test set of Table 6. Sel.: mean share of visible keys selected (%). Coverage (%) of the head-aware (HA), uniform (U) and shuffled (S, mean of three seeds) policies, and score differences in points with 95% CIs.

		Coverage				
Setting	Sel.	HA	U	S	HA − U	HA − S
Qwen CWE 16K	4.9	93.7	93.5	88.0	+0.6 [−0.8, 2.0]	**+7.6 [5.8, 9.3]**
Qwen CWE 32K	5.6	94.0	95.1	87.8	+0.6 [−1.3, 2.5]	**+7.8 [5.7, 9.9]**
Qwen FWE 16K	5.5	94.3	93.9	88.4	+0.3 [−1.3, 2.3]	**+1.6 [0.4, 2.8]**
Qwen FWE 32K	8.2	94.4	95.6	87.2	**−1.3 [−2.7, −0.3]**	+0.2 [−1.4, 1.9]
Llama CWE 64K	6.4	92.4	94.0	84.1	**−2.8 [−4.4, −1.4]**	**−2.1 [−3.6, −0.6]**
Llama FWE 64K	10.6	93.4	96.2	86.0	−0.3 [−1.7, 1.0]	**+15.6 [12.4, 19.1]**
Llama FWE 128K	15.2	92.9	97.6	82.3	−0.3 [−3.3, 2.3]	**+16.6 [12.0, 21.0]**