CACHENOTES: Task-Aware Key-Value Cache Compression for Reasoning-Intensive Knowledge Tasks

Giulio Corallo[1,2] **Orion Weller**[3] **Fabio Petroni**[4] **Paolo Papotti**[2]
[1]SAP Labs, France [2]EURECOM, France
[3]Johns Hopkins University, USA [4]European Molecular Biology Laboratory, Italy
Correspondence: giulio.corallo@sap.com

Abstract

Integrating external knowledge into Large Language Models (LLMs) is crucial for many real-world applications, yet current methods like Retrieval-Augmented Generation (RAG) face limitations with broad, multi-source queries, while long-context models are computationally prohibitive.

We introduce CACHENOTES: *Task-Aware Key-Value Cache Compression*. Given a task description and a corpus, CACHENOTES first generates a sequence of Compression-Planning-Tokens (CPTs), an offline task-focused distillation pass that identifies and organizes key information from the corpus. These CPTs are then used to guide a one-time compression of the corpus into a compact, reusable KV cache, which is then used alone at inference time to efficiently answer diverse, reasoning-intensive queries, eliminating repeated retrieval or context expansion.

Experiments on LongBench show that, on Question-Answering tasks at a $20\times$ compression, CACHENOTES outperforms RAG by over 8 F1 points and reduces latency by over $4\times$. On RULER, it surpasses previous *query-agnostic* compression methods by 55 points, narrowing the gap to *query-aware* compression approaches. Additional results on real-world enterprise and synthetic datasets demonstrate its strong performance on multi-hop and broad-coverage queries.

1 Introduction

Incorporating external information into Large Language Models (LLMs) significantly enhances their utility across applications, enabling more informed and accurate outputs (Petroni et al., 2019). Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) effectively injects relevant evidence for narrow, well-scoped queries, but it struggles with questions that require synthesizing information spread across multiple documents or sections of text (Barnett et al., 2024). Such multi-hop or broad queries are hindered by retrieval's reliance on local similarity, which fails to capture global relationships or latent links within the corpus, making it difficult to surface all relevant context (Weller et al., 2025) and often introducing noise or redundancy (Yu et al., 2024).

Recent advances have extended the context length that LLMs can process (Reid et al., 2024; Li et al., 2025a), making it possible to feed the entire corpus as input and enabling more holistic reasoning. However, this approach comes with significant computational costs, as handling large inputs requires substantial memory resources, particularly on GPUs, creating a scalability bottleneck (Liu et al., 2023). Furthermore, as context length grows, models often struggle to identify and exploit the truly relevant information buried within extensive text (Liu et al., 2024; Laban et al., 2025). This leads to a fundamental challenge:

> *How can LLMs efficiently and accurately perform reasoning-intensive tasks that require broad access to large external corpora, without the prohibitive costs of full-context inference or the limitations of retrieval-based methods?*

We address this challenge by introducing CACHENOTES: *Task-Aware Key-Value Cache Compression*, a *query-agnostic* yet task-guided framework that converts an entire corpus into a reusable, compact Key-Value (KV) cache. Unlike prior *query-aware* compression, which recomputes a compressed KV cache for each query (Li et al., 2025b; Rehg, 2024; Corallo and Papotti, 2024; Xu et al., 2025), or generic, prompt-based compression approaches (Kim et al., 2024), our method performs a one-time, offline compression guided by a human-written task description and a model-generated sequence of Compression-Planning-Tokens (CPTs). These CPTs distill key

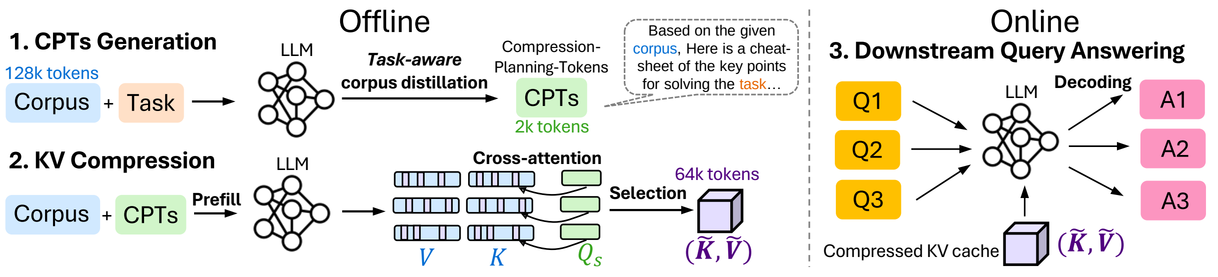


Figure 1: An illustration of our compression strategy. (1) A sequence of *Compression-Planning-Tokens* (CPTs) S is generated from the corpus, guided by a task description (e.g., factual QA). (2) The corpus KV cache (e.g., 128k tokens) is then compressed (e.g., to 64k tokens) *using* the CPTs Q_S as a proxy to preserve the most salient information. (3) At inference time, the LLM answers broad questions using only the compressed KV cache, as if it had access to the entire (uncompressed) corpus.

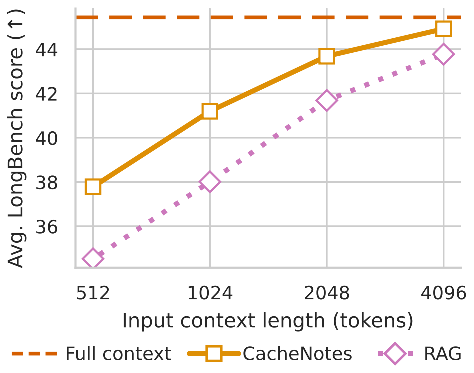


Figure 2: Aggregated results on 16 datasets of LongBench for LLAMA-3.1-8B-INSTRUCT. Our method CACHENOTES surpasses RAG when the input length is much smaller than the average corpus size of 10k tokens. RAG top-k: 2, 4, 8, 16.

information and act as a proxy for cross-attention-based token selection over the corpus. Figure 1 illustrates the workflow: two offline steps (CPTs generation and corpus KV compression) followed by one online step (downstream query answering). The resulting compressed KV cache can be rapidly leveraged at inference for any query within the task domain, eliminating retrieval, query-dependent re-compression, and fine-tuning, thereby overcoming context-length limitations while maintaining accuracy. Examples of CPTs and task descriptions are in Appendix B.2.

CACHENOTES establishes a new state of the art among *query-agnostic* compression methods and closely approaches more computationally intensive *query-aware* compression. On LongBench (Bai et al., 2024a), CACHENOTES outperforms RAG across high compression rates (e.g., compressing to 512–2048 tokens) despite being query-agnostic, and at a 2.5× compression rate it achieves 98.9% of full-context performance. Across LongBench and RULER, using LLAMA 3.1 (Dubey et al., 2024), QWEN 2.5 (Yang et al., 2024), and QWEN3, our task-aware compression consistently exceeds prior *query-agnostic* methods and retrieval-based approaches, while remaining competitive with *query-aware* compression. On real-world enterprise data and synthetic corpora requiring synthesis over widely distributed evidence, CACHENOTES markedly outperforms RAG, positioning compression as a key enabler for scaling LLM reasoning beyond retrieval-based methods.

2 Motivation and Problem Statement

The central challenge we address is enabling LLMs to perform *reasoning-intensive knowledge tasks* over large external corpora efficiently and effectively (Petroni et al., 2021; Su et al., 2024). As discussed in Section 1, current approaches force a trade-off: RAG is fast and cheap, but often misses dispersed evidence for multi-hop queries across many documents. Feeding the full corpus to a long-context model offers comprehensiveness, but is constrained by context length and cost. In particular, the *prefill* stage, where the model processes the input context, is **compute-bound**, as it requires a full forward pass over potentially millions of tokens. In contrast, the *decoding* stage, where tokens are generated one-by-one, is **memory-bound**, dominated by repeated reads from the KV cache.

Cache-Augmented Generation (CAG) (Chan et al., 2025) mitigates the cost of long context models by disaggregating the two stages: the expensive prefill can be done *offline*, producing a reusable KV cache so inference does not repeatedly reprocess the corpus. However, CAG alone leaves two issues: (i) the *memory bottleneck* from storing a large cache and (ii) the context-length limitation

Table 1: Comparison of KV cache strategies in terms of quality and efficiency for long-context LLM generation.

Method	Output Quality	Compute-Efficient	Memory-Efficient	Context Length
CAG	High	✓	✗	Model-limited
Query-aware KV comp.	High	✗	✓	Compression-set
Query-agnostic KV comp.	Low	✓	✓	Compression-set
Task-aware KV comp. (ours)	High	✓	✓	Compression-set

that caps how much content can be cached.

State of the Art. Methods differ in how they balance prefill and decoding (Table 1). *Full-context caching* (CAG) delivers high quality but requires memory that scales with corpus size. *Query-aware KV compression* (Corallo and Papotti, 2024; Li et al., 2025b; Corallo et al., 2025) shrinks the cache per query and preserves quality but reintroduces online cost by rerunning compression for every new query. *Query-agnostic compression* (Zhang et al., 2023; Devoto et al., 2024; Xiao et al., 2024a; Kim et al., 2025; Feng et al., 2025), in contrast, distills the corpus *once offline* into a compact KV cache reusable across different queries, but typically suffers notable quality drops relative to query-aware and full-context caching (Devoto et al., 2025).

Research Question. *Can we design a **query-agnostic** compression method that (i) preserves the efficiency benefits of offline prefill, (ii) alleviates the decoding-stage memory bottleneck, and (iii) achieves output quality on par with query-aware approaches?*

3 Methodology

We present our *task-aware, query-agnostic* compression strategy, motivated by the remarkable in-context learning capabilities of modern LLMs.

3.1 Intuition

Modern LLMs exhibit strong in-context learning capabilities (Brown et al., 2020; Dong et al., 2024): when provided with a sufficiently informative prefix, they can infer the target task and leverage relevant knowledge. Recent work has introduced models that explicitly generate intermediate reasoning traces before producing a final answer (Guo et al., 2025; Yang et al., 2025a). These models achieve substantially better performance on long-context reasoning tasks compared to their counterparts that skip this 'thinking' step (Bai et al., 2024b). We hypothesize that such reasoning traces function as an attention scaffold: during generation they progressively guide the model's attention toward the most relevant segments of the input corpus and at the end of the reasoning process, attention is concentrated on the key information needed to produce an accurate answer. In other words, they **denoise** the corpus by removing distractors and filtering out low-value content.

Inspired by this observation, we design a *query-agnostic* KV cache compression that denoises the corpus **offline**, before any query is posed. Given only a corpus and a brief task description, we prompt the model to generate a task-focused distillation pass, producing the **Compression-Planning Tokens (CPTs)**: intermediate tokens that highlight what the model considers essential for downstream query resolution. We then use CPTs as an attention scaffold: their query vectors score corpus keys via cross-attention (Eq. 3), and we keep the top-scoring tokens to form the compressed KV cache (rather than using CPTs as a textual prefix).

3.2 Task-Aware KV Compression

We propose a *task-aware* KV cache compression framework consisting of three stages (Figure 1):

(1) CPTs Generation (Offline). Given a corpus C and a task description T (e.g., *open-domain QA*), we prompt an instruction-following model $\mathrm{LLM}_{\mathrm{inst}}$ to generate a sequence of *Compression-Planning Tokens* (CPTs) S:

$$S = \mathrm{LLM}_{\mathrm{inst}}(C, T) \quad (1)$$

The prompt encourages the model to reason over the corpus in a task-oriented way: *Before I give you the question, read carefully the given document and memorize it, ensuring you preserve all critical details for solving the task. Create a cheat sheet covering the entire context.* Task descriptions can be domain-specific or augmented with few-shot exemplars to steer S toward the desired reasoning style. These CPTs distill what the model deems essential and serve as a proxy for corpus compression in the next stage.

(2) Layer-wise KV Compression (Offline). We concatenate corpus and CPTs,

$$\mathbf{I} = [\mathbf{C}, \mathbf{S}] \in \mathbb{R}^{n+m} \tag{2}$$

where n and m are the token lengths of C and S. Passing $\mathbf{I}$ through the base LLM, we extract for each transformer layer the query vectors of CPTs tokens Q_s and the Key vector of corpus tokens K_c.

$$\mathbf{W}^{(S,C)} = \mathrm{softmax}\left(\frac{Q_s K_c^\top}{\sqrt{d_k}}\right) \tag{3}$$

The principle of this operation is to 'probe' the corpus tokens that the CPTs highlight. To amplify the influence of more informative CPTs, we weight each row by the norm of the corresponding CPTs value vectors V_s (Devoto et al., 2025):

$$\mathbf{W}^{(S,C)}_{\mathrm{norm}} = \mathbf{W}^{(S,C)} \cdot \|V_s\| \tag{4}$$

Averaging across CPTs tokens yields a single importance score per corpus token; we select the top-k:

$$\mathrm{indices} = \mathrm{TopK}\big(\mathrm{avg}(\mathbf{W}^{(S,C)}_{\mathrm{norm}}), k\big) \tag{5}$$

We then gather the corresponding keys and values $\widetilde{K}, \widetilde{V} = (K_c, V_c)[indices]$. To preserve positional consistency after token pruning, we reapply rotary positional embeddings (Xiao et al., 2024a) by recalculating the appropriate sine-cosine offsets and re-rotating the compressed Key states $\widetilde{K}$.

(3) Downstream Query Answering (Online). When facing a new question $\mathbf{q}_{\mathrm{new}}$, at each attention layer, the $Keys$ and $Values$ of the question are appended to the precomputed cache:

$$\widetilde{\mathbf{K}} \leftarrow \begin{bmatrix} \widetilde{\mathbf{K}} \\ \mathbf{k}^{q_{\mathrm{new}}} \end{bmatrix}, \quad \widetilde{\mathbf{V}} \leftarrow \begin{bmatrix} \widetilde{\mathbf{V}} \\ \mathbf{v}^{q_{\mathrm{new}}} \end{bmatrix} \tag{6}$$

The LLM conditions on the updated $\widetilde{\mathbf{K}}$ and $\widetilde{\mathbf{V}}$ to answer. Inference cost now involves a single forward pass for the typically short question and scales linearly with the compressed cache $(\widetilde{\mathbf{K}}, \widetilde{\mathbf{V}})$.

3.3 Memory-Efficient Cross-Attention Scoring

Computing the cross-attention scores between the CPTs and a long corpus (Eq. 3) naively requires materializing a dense $m \times n$ attention matrix per layer, with memory complexity $\mathcal{O}(mnd)$. For very large corpora ($n \gg 10^5$) and non-trivial CPTs length ($m \sim 10^3$), this quickly exceeds GPU memory.

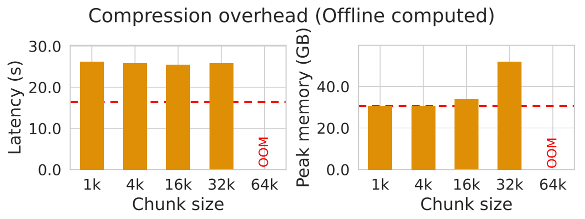


Figure 3: Compression's latency and GPU memory overhead (bars) compared to FlashAttention2 prefill (dotted red line).

Chunked Cross-Attention. To address this bottleneck, we adopt a mathematically equivalent *chunked* formulation that computes exact softmax scores without storing the entire matrix $m \times n$. Inspired by the *streaming softmax* trick used in FlashAttention (Dao et al., 2022), we partition the corpus keys into fixed-size chunks and process them sequentially, maintaining running maxima and partition functions to stabilize and normalize logits. This reduces memory from $O(mn)$ to $O(mc)$, where c is the chunk size, while remaining mathematically equivalent. Figure 3 reports **latency** (left) and **peak GPU memory** (right) of our chunked version, implemented in PyTorch (Paszke et al., 2019), on a $n = 128$k-token corpus with $m = 2$k CPTs. The dotted red line marks the cost of a vanilla prefill on the combined $n+m = 130$k tokens using FlashAttention2. Our chunked scoring keeps peak memory comparable to FlashAttention2 up to $c = 16$k token chunks and avoids out-of-memory failures that occur with the dense formulation at $c = 64$k tokens, with only a modest latency increase. Because compression in CACHENOTES is performed entirely *offline*, we prioritize memory scalability and correctness over runtime speed; future work could integrate specialized CUDA kernels or fused operations to further accelerate this step.

4 Experimental Setup

4.1 Models

We evaluate CACHENOTES on three open-weight LLMs: LLAMA-3.1-8B-INSTRUCT (Dubey et al., 2024), QWEN3-4B-INSTRUCT-2507 (Yang et al., 2025a), and QWEN2.5-14B-INSTRUCT (Yang et al., 2024), extended to 128k tokens with YARN (Peng et al., 2024). We additionally run QWEN3-32B on RULER to validate that CACHENOTES' trends hold at 30B+ scale (Appendix G).

4.2 Baselines

We compare against: **Query-aware:** RAG and FINCH (Corallo and Papotti, 2024). **Query-agnostic:** EXPECTED ATTENTION (Devoto et al., 2025), KVZIP (Kim et al., 2025), and **Full-Context** CAG (Chan et al., 2025) (upper bound). For compression-based methods, we maintain a uniform allocation per head and layer, varying only the scoring mechanism. We additionally report two ablations: (i) compression guided only by the task description (similar to the "catalyst prompt" of Kim et al. (2024)); (ii) using solely the generated CPTs as input (no corpus tokens compression).

Retrieval for RAG. We split the corpus into **256-token chunks** and use the dense retriever BGE-LARGE-EN-V1.5 (Xiao et al., 2024b) to select the top-k chunks for each query. For a target evidence budget $B \in \{512, 1024, 2048, 4096\}$ tokens we set $k = B/256$ (i.e., $k=\{2, 4, 8, 16\}$), ensuring the retrieved text fits the same token budgets as other methods. If the concatenated text would exceed B, we truncate by keeping the first $B/2$ tokens and the last $B/2$ tokens. The question/prompt tokens are not counted toward B. For completeness we also evaluated a *sparse* retriever (BM25) variant of RAG (Appendix C.1).

CPTs Generation. For each dataset sample (corpus and task description), we generate a one-time sequence of CPTs using vLLM (Kwon et al., 2023) for efficient offline inference. Crucially, we employ a *single, high-level task description per task family*, which is reused across all datapoints within that family. We provide task templates and compression prompt in Appendix B. While the quality of compression depends on the alignment between this description and the downstream task (see RQ3 for sensitivity analysis), this setup minimizes human supervision to a level comparable to defining a standard system prompt in RAG pipelines.

Decoding. All models use greedy decoding (Vijayakumar et al., 2016; Shao et al., 2017) for both CPTs and final answer generation.
Code: github.com/giulio98/cachenotes

4.3 Datasets and Metrics

We evaluated these methods on four datasets both with traditional evaluation metrics (such as F1 for QA) and LLM-based evaluation:

LongBench (Bai et al., 2024a) is a long-context benchmark spanning 16 datasets and six tasks, including single/multi-document QA, summarization, code completion, few-shot learning, and a synthetic reasoning task. Contexts average $\sim$10k tokens. Evaluation follows task-specific metrics: F1 for QA (Rajpurkar et al., 2016), ROUGE-L for summarization (Lin, 2004), and Edit Similarity for code completion (Svyatkovskiy et al., 2020).
🤗 **Data:** hf.co/datasets/giulio98/LongBench

RULER (Hsieh et al., 2024) is a synthetic benchmark that evaluates long-context reasoning across retrieval (extended NIAH), multi-hop tracing, aggregation, and QA. We follow common practice in other KV cache compression studies and use the 4k token subset (6,500 query-document pairs). RULER reports a String Match score, a case-insensitive substring matching metric, which is more lenient than strict exact match.
🤗 **Data:** hf.co/datasets/giulio98/ruler

Company Notes We evaluated CACHENOTES on *real-world* long documents, by curating 49 enterprise technical notes (16k–32k tokens; avg. 22k) from support and troubleshooting portals. Each document contains configuration guides, best-practice notes, and resolution procedures. Following Edge et al. (2024), we generate five *global queries* per document using GPT-4.1-MINI with the prompting strategy of Wei et al. (2025), yielding 245 query-context pairs.

Response quality is scored using Wei et al. (2025)'s automatic grader on three dimensions:

- **Helpful**: precision, contextual relevance, practical value;
- **Rich**: breadth and diversity of perspectives;
- **Insightful**: depth and originality.

We use GPT-4.1-MINI as the pairwise evaluator with the comparison prompt from Wei et al. (2025) and find a positive correlation with human judgments (see Appendix D).
🤗 **Data:** hf.co/datasets/giulio98/company-notes

Synthetic Dataset To assess the impact of inter-chunk connectivity on RAG and CACHENOTES, we design a controllable synthetic QA corpus. The corpus consists of 32k tokens organized into three structured chunk types: *People*, *Projects*, and *Memberships*. Each chunk captures a type of entity. The corpus structure is tuned through a connectivity parameter $k \in \{1, \ldots, 8\}$ that controls the

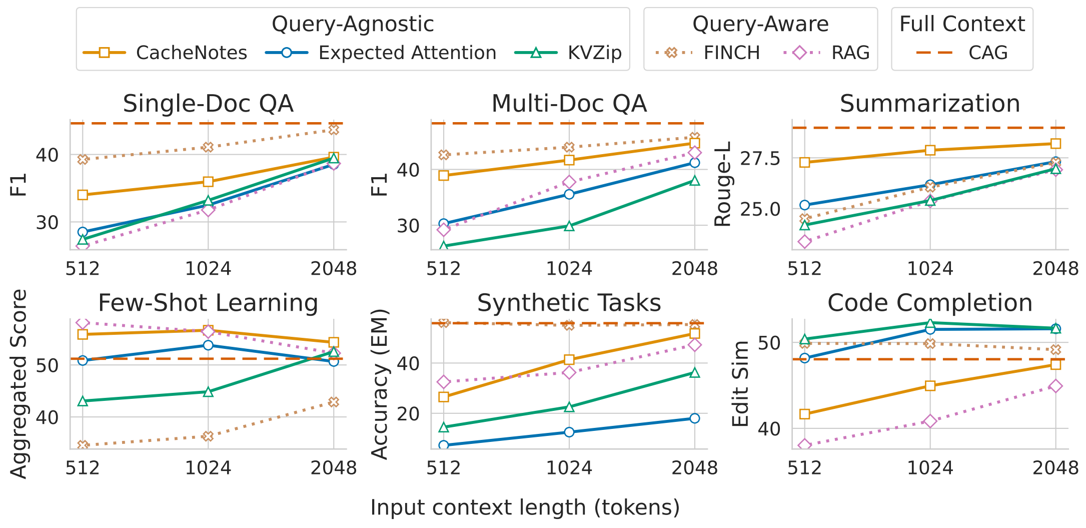


Figure 4: Performance results on Longbench for LLAMA-3.1-8B-INSTRUCT. RAG top-k: 2, 4, 8. Note that for *Code Completion*, minimal prompt enrichment (one-shot CPTs) recovers performance to match baselines (see Appendix C.2).

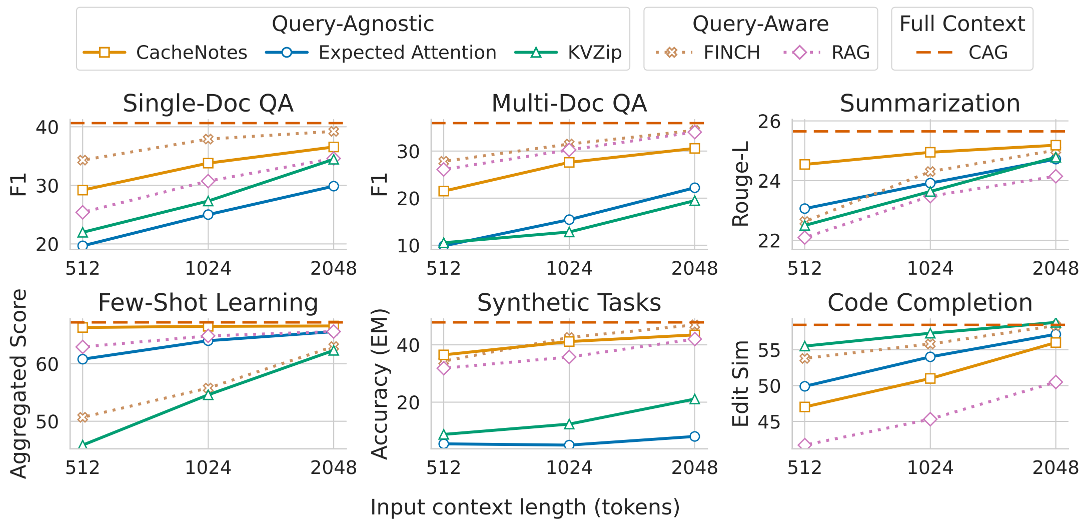


Figure 5: Performance results on Longbench for QWEN3-4B-INSTRUCT-2507. RAG top-k: 2, 4, 8.

number of projects each person is linked to. This design allows us to vary the degree of information spread across chunks, directly impacting the complexity of the queries. We generate two types of questions: *direct-retrieval* queries, which can be resolved from a single chunk, and *join-like* queries that require multi-hop reasoning across multiple chunks. For each connectivity level, we generate 50 queries (25 direct-retrieval and 25 join-like), resulting in a total of 400 queries. Ground truth answers are structured as lists of entities, allowing evaluation using the F1 score. Dataset construction details are provided in Appendix A. For the size of the answer output, we use 256 tokens.

🤗 **Data:** hf.co/datasets/giulio98/synthetic-dataset

5 Results and Discussions

We discuss five research questions.

RQ1: When Does CACHENOTES Surpass RAG? Across LongBench (Figures 4-5), CACHENOTES consistently matches or exceeds RAG. For instance, on *Single-Doc QA* it delivers **+3.8/+7.58 F1** over RAG at a 512-token budget for QWEN and LLAMA, respectively, and it remains competitive or superior on *Multi-Doc QA*.

The advantage widens on the *Company Notes* corpus (Fig. 6), where information is long and densely linked: CACHENOTES wins **30-40 more pairwise comparisons** in correctness and insightfulness at 1,024 tokens. LLM-as-judge preferences

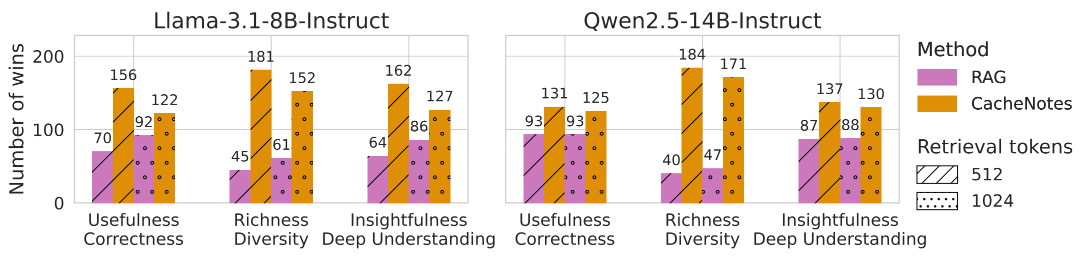


Figure 6: Performance results on Company Notes. RAG top-k: 2, 4.

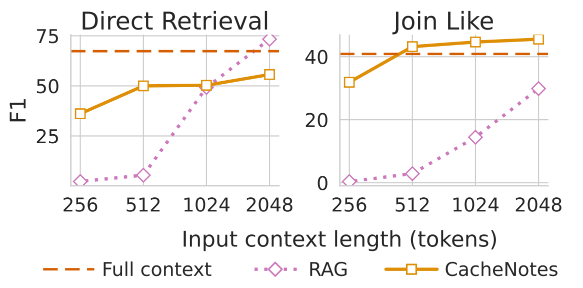


Figure 7: Performance results on Synthetic dataset for LLAMA-3.1-8B-INSTRUCT. RAG top-k: 1, 2, 4, 8.

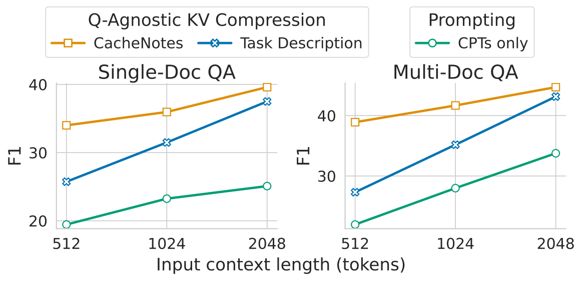


Figure 8: KV compression vs. using CPTs only. KVs selected by CPTs enhance downstream performance.

(62.7%) corroborate the LongBench results, confirming that the method better preserves dispersed evidence, though RAG can still win with concise, pragmatic answers. To isolate this effect, we benchmark the two methods on a controlled synthetic dataset (Figure 7) that contrasts *direct-retrieval* queries (answer lies in one chunk) with *join-like* queries (answer must be assembled across chunks). The results highlight a fundamental **precision-recall trade-off**. For *direct-retrieval*, RAG excels, acting as a high-precision filter that retrieves the exact localized evidence, while CACHENOTES, which applies global compression, inevitably downweights some fine-grained details. However, for *join-like* queries, recall becomes the bottleneck: RAG's performance drops by ∼15 points as connectivity increases, because missing even a single chunk in the evidence set leads to failure. In contrast, CACHENOTES preserves a global view of the corpus, allowing it to track or even outperform full-context performance.

RQ2: Why KV Compression When CPTs Already Exist? We ablated three variants on LongBench QA (Figure 8): (i) prompting directly from the CPTs (no corpus compression), (ii) CACHENOTES, and (iii) KV compression guided only by the task description.

CPTs-only input consistently underperforms both compression variants across 512/1,024/2,048-token budgets in Single-Doc and Multi-Doc QA. Even the weaker *task description only* compression beats CPTs-only, showing that **token-level, context-aware KV representations preserve cues that a plain textual prefix cannot**. Conversely, adding CPTs to guide compression clearly improves over task description only, indicating that high-level task hints alone are insufficient for strong downstream performance.

RQ3: Are *Compression-Planning Tokens* transferable across tasks? We test whether CPTs generated for one task (Summarization) can be reused for others by comparing CACHENOTES with its variant CACHENOTES-S, where CPTs are always derived from Summarization (Fig. 9). Results show clear *negative transfer*: performance drops on *Passage Count* (-3.8), *Passage Retrieval* (-12.5), and *TREC* (-8.0), with negligible change for *Code Completion* and *QA*. This indicates that summarization-driven CPTs capture broad, high-level signals but fail to encode the fine-grained evidence required by well-defined tasks, underscoring the need for task-specific CPTs generation.

RQ4: Can CACHENOTES Rival Query-Aware Compression? Figures 4-5 reveal that, despite being *query-agnostic*, CACHENOTES matches or outperforms the query-aware method FINCH on two LongBench tasks. On *Summarization*, it is consistently **+2-3 ROUGE-L** over FINCH across

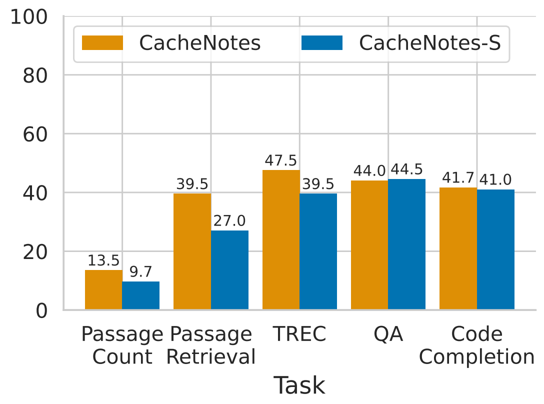


Figure 9: CACHENOTES vs. CACHENOTES-S (CPTs generated for Summarization) on LongBench with LLAMA-3.1-8B-INSTRUCT @512 tokens.

512–1,024 tokens; on *Few-Shot Learning*, it gains **+15-20 EM**, occasionally surpassing even the full-context upper bound. We attribute this to the CPTs providing a *globally coherent view* of the corpus, which benefits abstractive summarization and example-driven generalization.

For *Code Completion*, CACHENOTES trails the query-agnostic EXPECTED ATTENTION by **4-5 Edit-Sim** points, while FINCH remains competitive. This highlights a specific trade-off: while CPTs excel at capturing *global* semantic links for QA and Summarization, they can miss the *local* syntactic precision required for next-line code prediction, often summarizing the overall file logic rather than focusing on the immediate cursor context. However, this gap is not inherent to the compression mechanism: adding a single, lightweight *one-shot* code-completion example to the CPTs-generation prompt is sufficient to make CPTs emphasize the local variables/methods needed for next-line prediction, recovering performance across budgets (Appendix C.2, Table 4).

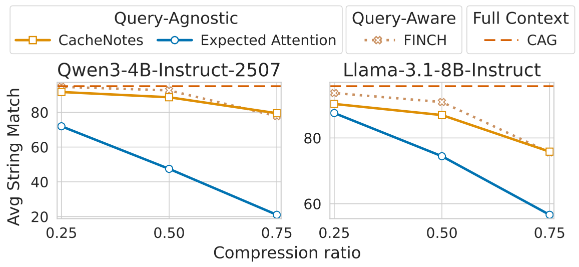


Figure 10: Results on RULER: CACHENOTES narrows the gap between query-aware and query-agnostic KV compression.

On RULER (Figure 10), using QWEN3-4B-INSTRUCT-2507, CACHENOTES remains robust under heavy compression: at 0.75 compression rate, it stays **+2 SM** above FINCH and **+55 SM** above EXPECTED ATTENTION.

Table 2: TTFT (s) and Peak GPU memory (GB) across corpus lengths. Each cell: *TTFT / Mem*. Lower is better.

Method	**16k**	**32k**	**64k**	**128k**
Full Prefill	1.03 / 17.92	2.31 / 19.88	5.22 / 23.78	15.88 / 31.6
Q-Aware	1.19 / **16.95**	2.38 / **16.95**	5.70 / **16.95**	17.33 / **16.95**
CAG	0.17 / 17.92	0.20 / 19.88	0.21 / 23.78	0.28 / 31.6
RAG	0.61 / **16.95**	0.61 / **16.95**	0.61 / **16.95**	0.61 / **16.95**
CacheNotes	**0.13 / 16.95**	**0.13 / 16.95**	**0.13 / 16.95**	**0.13 / 16.95**

RQ5: Impact on Practical Deployment. Table 2 reports online *time-to-first-token* (TTFT) and peak GPU memory as the corpus scales from 16k to 128k tokens (8k retrieval budget, 512-token prompt, single H100 GPU). Full-context prefill with FlashAttention2 slows sharply (1.0 s to **15.9 s**) and grows to 31.6 GB memory; *query-aware compression* similarly reaches **17.3 s**. RAG avoids full prefill but still incurs retrieval + chunk prefill (**0.61 s**). In contrast, CACHENOTES reuses an *offline-built* KV cache, keeping TTFT nearly flat at **0.13 s**, over **4× faster** than RAG and **122× faster** than full prefill at 128k, while capping memory at 16.9 GB. This efficiency comes from eliminating quadratic prefill: online cost depends only on the compact cache size, not corpus length.

Offline Computation. On an H100 GPU, processing a 100k-token corpus takes 13.5 s to generate 2k-token CPTs and 25.9 s to compress them, about 2.5× the cost of a vanilla FlashAttention2 prefill (15.9 s). Because this step is *one-time* and amortized across queries, our pipeline becomes more cost-efficient than full prefill after just ∼**2** downstream queries. In practice, CPTs generation and compression can run during maintenance or data-refresh windows, similar to RAG index rebuilding, so they do not impact online latency.

Dynamic Corpora. While our experiments assume a static corpus, the approach extends to evolving knowledge bases. When a document is added or modified, only that specific context requires CPTs generation and compression. This isolated update incurs the one-time offline cost (2.5× prefill), which is then amortized over all future queries accessing that context, similar to incremental indexing in RAG.

6 Related Work

Graph Retrieval-Augmented Generation. Traditional RAG retrieves text by semantic similarity but often misses cross-document links and relational evidence needed for complex reasoning. Graph Retrieval-Augmented Generation (GraphRAG) (Han et al., 2024; Edge et al., 2024; Mavromatis and Karypis, 2024) mitigates this by structuring the corpus into a graph, nodes as entities or passages, edges as relations, and retrieving paths, subgraphs, or communities to combine local content with global connectivity. However, GraphRAG requires *heavy engineering*: entity/relation extraction, schema design, graph maintenance, and community summarization: hard to automate and brittle on noisy, heterogeneous corpora. Our method avoids this engineering effort: it is fully unsupervised, simple to deploy, and lets models *see* the entire corpus via KV cache compression.

Reasoning-Intensive Retrieval. Dense retrievers struggle with multi-hop reasoning in knowledge-intensive tasks. Reasoning-aware retrieval (Shao et al., 2025; Wang et al., 2024; Trivedi et al., 2023) addresses this by coupling retrieval with model reasoning, but increases inference cost and requires per-query recomputation. Our approach differs by **decoupling reasoning from inference**, shifting this effort *offline* with CPTs generation and compressing the corpus into a reusable KV cache for any downstream query.

7 Conclusion

We introduced *task-aware KV compression*, enabling LLMs to efficiently process large corpora by pre-populating compact Key-Value states. While our work demonstrates significant promise, our experiments primarily use relatively small corpora; as context windows grow in open LLMs, the method becomes increasingly relevant for realistic large-scale settings. Future work should address other challenges before widespread deployment in production systems, particularly concerning the merging of compressed caches and the computational scalability of the compression process. Our results also reveal a complementary strength between KV compression (broad, multi-source queries) and RAG (narrow, focused queries), suggesting that a hybrid approach, offline corpus compression for global coverage with dynamic online retrieval for specificity, could further improve performance.

Limitations

Our method assumes that, at offline time, the base LLM can process at least the corpus within its context window and that sufficient GPU memory is available to run a full prefill for CPTs generation and cross-attention-based compression. When this assumption does not hold (e.g., the corpus exceeds the model's context or available memory), a natural workaround is to split it into smaller chunks, compress each independently, and later concatenate them at inference time such that the model can reason seamlessly across multiple sources. However, independently compressed caches lack cross-document attention: naively concatenating them neither restores inter-document dependencies nor preserves positional consistency. Recent works such as CacheBlend (Yao et al., 2025) address merging for uncompressed caches via selective recomputation, while methods like KVLink (Yang et al., 2025b) propose positional alignment and bridging tokens for improved reuse, but a robust, general solution for merging *compressed* caches remains open. Future research could explore principled strategies for alignment and fusion under compression, enabling scalable reasoning across large, fragmented corpora.

Acknowledgments

We thank the action editor and the reviewers for their comments, which helped us improve this work. We also thank Camilla Ferrario for designing the main figure and Riccardo Taiello for insightful research discussions.

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. 2024a. 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)*, pages 3119–3137, Bangkok, Thailand. Association for Computational Linguistics.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, and 1 others. 2024b. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*.

Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, and Mohamed Abdelrazek. 2024.

Seven failure points when engineering a retrieval augmented generation system. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*, pages 194–199.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Brian J Chan, Chao-Ting Chen, Jui-Hung Cheng, and Hen-Hsen Huang. 2025. Don't do rag: When cache-augmented generation is all you need for knowledge tasks. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 893–897.

Giulio Corallo, Elia Faure-Rolland, Miriam Lamari, and Paolo Papotti. 2025. TableKV: KV cache compression for in-context table processing. In *Proceedings of the 4th Table Representation Learning Workshop*, pages 166–171, Vienna, Austria. Association for Computational Linguistics.

Giulio Corallo and Paolo Papotti. 2024. Finch: Prompt-guided key-value cache compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1517–1532.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359.

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

Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. 2024. 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*, pages 18476–18499, Miami, Florida, USA. Association for Computational Linguistics.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128, Miami, Florida, USA. Association for Computational Linguistics.

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

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. 2025. Identify critical kv cache in llm inference from an output perturbation perspective. *arXiv preprint arXiv:2502.03805*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, and 1 others. 2024. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309*.

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

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W. Lee, Sangdoo Yun, and Hyun Oh Song. 2025. KVzip: Query-agnostic KV cache compression with context reconstruction. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*.

Minsoo Kim, Kyuhong Shim, Jungwook Choi, and Simyung Chang. 2024. InfiniPot: Infinite context processing on memory-constrained LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16046–16060, Miami, Florida, USA. Association for Computational Linguistics.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2025. Llms get lost in multi-turn conversation. *arXiv preprint arXiv:2505.06120*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, Cheng Zhu, Chunhao Zhang, Congchao Guo, Da Chen, Dong Li, and 1 others. 2025a. Minimax-01: Scaling foundation models with lightning attention. *arXiv preprint arXiv:2501.08313*.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2025b. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2023. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Costas Mavromatis and George Karypis. 2024. Gnn-rag: Graph neural retrieval for large language model reasoning. *arXiv preprint arXiv:2405.20139*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2024. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Isaac Rehg. 2024. Kv-compress: Paged kv-cache compression with variable compression rates per attention head. *arXiv preprint arXiv:2410.00161*.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy P. Lillicrap, Jean-Baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, Ioannis Antonoglou, Rohan Anil, Sebastian Borgeaud, Andrew M. Dai, Katie Millican, Ethan Dyer, Mia Glaese, Thibault Sottiaux, Benjamin Lee, and 34 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *CoRR*, abs/2403.05530.

Rulin Shao, Rui Qiao, Varsha Kishore, Niklas Muennighoff, Xi Victoria Lin, Daniela Rus, Bryan Kian Hsiang Low, Sewon Min, Wen-tau Yih, Pang Wei Koh, and 1 others. 2025. Reasonir: Training retrievers for reasoning tasks. *arXiv preprint arXiv:2504.20595*.

Yuanlong Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating High-Quality and Informative Conversation Responses with Sequence-to-Sequence Models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2210–2219, Copenhagen, Denmark. Association for Computational Linguistics.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han yu Wang, Haisu Liu, Quan Shi, Zachary S. Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan Ö. Arik, Danqi Chen, and Tao Yu. 2024. Bright: A realistic and challenging benchmark for reasoning-intensive retrieval. *CoRR*, abs/2407.12883.

Alexey Svyatkovskiy, Shao Kun Deng, Shengyu Fu, and Neel Sundaresan. 2020. Intellicode compose: Code generation using transformer. In *Proceedings of the 28th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering*, pages 1433–1443.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David

Crandall, and Dhruv Batra. 2016. Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models. *arXiv:1610.02424*.

Ziting Wang, Haitao Yuan, Wei Dong, Gao Cong, and Feifei Li. 2024. Corag: A cost-constrained retrieval optimization system for retrieval-augmented generation. *arXiv preprint arXiv:2411.00744*.

Jiale Wei, Shuchi Wu, Ruochen Liu, Xiang Ying, Jingbo Shang, and Fangbo Tao. 2025. Tuning llms by rag principles: Towards llm-native memory. *arXiv preprint arXiv:2503.16071*.

Orion Weller, Michael Boratko, Iftekhar Naim, and Jinhyuk Lee. 2025. On the theoretical limitations of embedding-based retrieval. *arXiv preprint arXiv:2508.21038*.

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

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024b. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 641–649, New York, NY, USA. Association for Computing Machinery.

Yuhui Xu, Zhanming Jie, Hanze Dong, Lei Wang, Xudong Lu, Aojun Zhou, Amrita Saha, Caiming Xiong, and Doyen Sahoo. 2025. Think: Thinner key cache by query-driven pruning. In *The Thirteenth International Conference on Learning Representations*.

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

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Jingbo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. 2025b. Kvlink: Accelerating large language models via efficient kv cache reuse. *arXiv preprint arXiv:2502.16002*.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. 2025. Cacheblend: Fast large language model serving for rag with cached knowledge fusion. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, page 94–109, New York, NY, USA. Association for Computing Machinery.

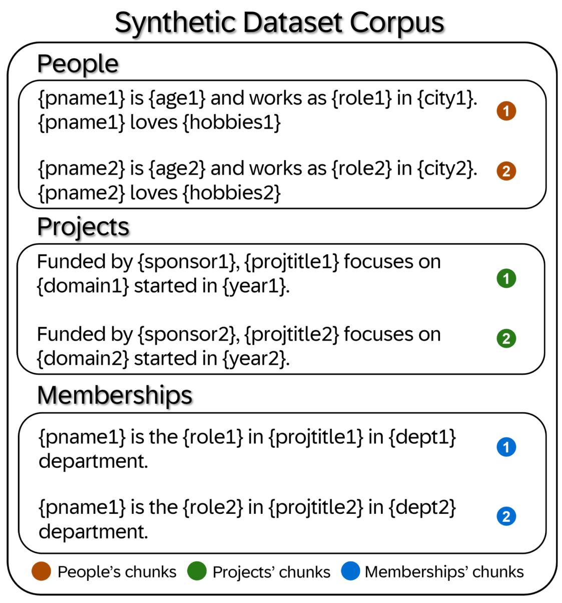

Synthetic Dataset Corpus
People
{pname1} is {age1} and works as {role1} in {city1}. {pname1} loves {hobbies1}
1
{pname2} is {age2} and works as {role2} in {city2}. {pname2} loves {hobbies2}
2
Projects
Funded by {sponsor1}, {projtitle1} focuses on {domain1} started in {year1}.
1
Funded by {sponsor2}, {projtitle2} focuses on {domain2} started in {year2}.
2
Memberships
{pname1} is the {role1} in {projtitle1} in {dept1} department.
1
{pname1} is the {role2} in {projtitle2} in {dept2} department.
2
People's chunks
Projects' chunks
Memberships' chunks

Figure 11: Overview of our synthetic dataset. In this example, the connectivity level is set to 2.

Shuo Yu, Mingyue Cheng, Jiqian Yang, and Jie Ouyang. 2024. A knowledge-centric benchmarking framework and empirical study for retrieval-augmented generation. *arXiv preprint arXiv:2409.13694*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, and 1 others. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.

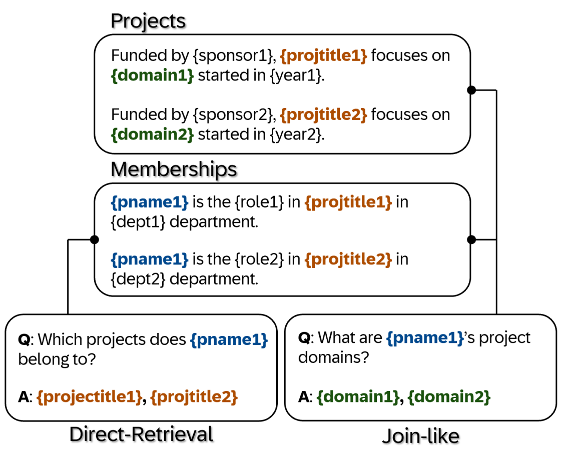
A Synthetic Dataset Construction

We construct a synthetic dataset designed to precisely control corpus complexity and the connectivity level between text chunks. By varying inter-chunk connectivity, we are able to thoroughly evaluate different methods, identifying the exact scenarios where each technique performs well or fails. Figure 11 illustrates the structured design of our corpus. Our dataset will be publicly released to support future research.

A.1 Structured Entities and Corpus Chunks

We define three entity types.

People. Each person is described through template-structured biographies containing attributes such as *name*, *age*, *occupation*, *city*, and *hobbies*. To maintain uniformity and facilitate controlled experiments, each biography text chunk is standardized to a length of 256 tokens using additional neutral filler text.



Projects
Funded by {sponsor1}, {projtitle1} focuses on {domain1} started in {year1}.
Funded by {sponsor2}, {projtitle2} focuses on {domain2} started in {year2}.
Memberships
{pname1} is the {role1} in {projtitle1} in {dept1} department.
{pname1} is the {role2} in {projtitle2} in {dept2} department.
Q: Which projects does {pname1} belong to?
A: {projectitle1}, {projtitle2}
Q: What are {pname1}'s project domains?
A: {domain1}, {domain2}
Direct-Retrieval
Join-like

Figure 12: Overview of our questions. In this example, the connectivity level is set to 2.

Projects. Each project has attributes including *title*, *domain*, *sponsor*, *year started*, and a descriptive summary. Like for people, each text chunk is standardized to 256 tokens.

Memberships. A membership represents the relationship between people and projects and specifies the *role* (e.g., *Engineer*, *Manager*) and *department* (e.g., *R&D*, *Marketing*) that a person holds in a project. These text chunks similarly include filler text to meet the fixed-length criterion.

We generate multiple corpus instances with varying *connectivity levels*, ranging from 1 to 8, where level k means each person links to exactly k projects. Higher connectivity increases dataset complexity by distributing relevant information about a person across multiple membership and project chunks, thus challenging the model's ability to synthesize scattered information. Each corpus at a given connectivity level comprises exactly 32k tokens, ensuring consistent corpus size across experiments.

A.2 Controlled Question Types for Evaluation

To rigorously evaluate the performance of both KV-cache compression and retrieval-based methods, we generate two primary question categories (Figure 12).

Direct Retrieval Questions. These questions require information localized within a single *Memberships* chunk. Example templates include:

- *Which projects does* `pname` *belong to?*
- *Which role does* `pname` *have in* `projtitle`*?*
- *Which department is* `pname` *part of?*

Answering these queries does not require cross-chunk synthesis. As shown in Figure 12, to solve these queries, the model has to (1) Locate the single *Memberships* chunk whose `name` attribute matches the query subject, (2) Read the target attribute `projects`, `role`, or `department` directly from that chunk, (3) Return the extracted value(s) without needing to consult any other chunks.

Join-like Questions. Answering these queries requires combining information across multiple *Memberships* and *Projects* chunks. For example:

- *What are* `pname`*'s project domains?*
- *In which years did* `pname`*'s projects begin?*
- *Who sponsors* `pname`*'s projects?*

Addressing these queries tests a model's capability for multi-hop reasoning and synthesis across distributed knowledge sources. As the connectivity level grows, these join-like questions become increasingly complex, requiring the aggregation of information from multiple chunks. As shown in Figure 12, to solve these queries, the model has to (1) Open all *Memberships* chunks whose `name` equals the queried person to collect the list of linked project titles. (2) For each collected project title, open the corresponding *Projects* chunk. (3) Extract the requested attribute (`domain`, `year started`, or `sponsor`) from each project chunk. (4) Return gathered attribute values.

For each connectivity level (1 through 8), we generate 50 distinct queries: 25 direct-retrieval and 25 join-like, totaling 400 distinct queries across all connectivity levels.

B Evaluation Setups

We document the datasets, task formulations, and the prompts used to generate *Compression-Planning Tokens (CPTs)*.

B.1 Prompting Interfaces

Notation. `{context}` = document(s) or passages; `{question}` = downstream query. CPTs are produced by appending **compression prompt** to **task template**. Inference uses (compressed) **task template** and **question template**.

B.1.1 LongBench

Task template. We use exactly the instructions defined in Appendix B of Bai et al. (2024a).

Question template. Refer to Appendix B of Bai et al. (2024a).

Compression prompt.

- *Single-Doc/Multi-Doc QA/QMSum:*

Before I give you the question, imagine you are a student memorizing this material according to the task you will have to perform. Repeat the context concisely yet comprehensively to aid memorization, preserving all critical details and creating a cheat sheet. Once done, I will give you the question.

- *Summarization:*

  Before solving the summarization task, imagine you are a student memorizing this material according to the task. Repeat the context concisely yet comprehensively, preserve all key details, and create a cheat sheet. Once done, I will prompt you to solve the task.

- *TREC:*

  Before I give you the new question, imagine you are a student memorizing this material according to the task that you will have to perform. Reason about the given examples and the expected output, ensuring you preserve all critical details and create a cheat sheet covering the entire context. Once you have done that, I give you the new question.

- *TriviaQA:*

  Before I give you the new passage and the question, imagine you are a student memorizing this material according to the task that you will have to perform. Reason about the given examples and the expected output, ensuring you preserve all critical details and create a cheat sheet covering the entire context. Once you have done that, I give you the new passage and question.

- *SAMSum*

  Before I give you the new dialogue to summarize, imagine you are a student memorizing this material according to the task that you will have to perform. Reason about the given examples and the expected output, ensuring you preserve all critical details and create a cheat sheet covering the entire context. Once you have done that, I give you the new dialogue.

- *Passage Count*

  Before solving the counting task, imagine you are a student memorizing this material according to the task that you will have to perform. Ensure you preserve all critical details to solve the task and create a cheat sheet covering the entire context. Once you have done that, I will prompt you to solve the task.

- *Passage Retrieval*

  Before solving the passage retrieval task and give you the abstract, imagine you are a student memorizing this material according to the task that you will have to perform. Ensure you preserve all critical details to solve the task and create a cheat sheet covering the entire context. Once you have done that, I will prompt you to solve the task giving you the abstract.

- *LCC*

  Before solving the code completion task, imagine you are a student memorizing this material according to the task that you will have to perform. Ensure you preserve all critical details to solve the task and create a cheat sheet covering the entire context. Once you have done that, I will prompt you to solve the code completion task.

- *RepoBench-p*

  Before giving you the last snippet of code and solving the code completion task, imagine you are a student memorizing this material according to the task that you will have to perform. Ensure you preserve all critical details to solve the task and create a cheat sheet covering the entire context. Once you have done that, I will prompt you to solve the code completion task giving you the last snippet of code.

B.1.2 RULER

Task template. Follow the format in Tables 7-8 of Hsieh et al. (2024).

Question template. Refer to Tables 7-8 of Hsieh et al. (2024).

Compression prompt.

You were given a task. Before I give you the question, imagine you are a student memorizing this material according to the task. Repeat the context concisely yet comprehensively, preserve all critical details, and create a cheat sheet to solve the task. Once done, I will give you the question.

B.1.3 Company Notes

Task template.

You are a helpful assistant who can answer the user query according ONLY to the Company Note provided. Provide detailed and accurate information based on the user's question. Here is the Company Note (HTML):

`{context}`

Question template.

Here is the user question:
`{question}`

Please provide your answer.

Compression prompt.

Before I give you the question, imagine you are a student memorizing this material according to the task you will have to perform. Repeat the context concisely yet comprehensively to aid memorization, preserving all critical details and creating a cheat sheet. Once done, I will give you the question.

B.1.4 Synthetic Dataset

Task template.

Answer the question based on the given passages. Only give the answer; no extra text. Answers should be concise lists separated by commas.
`{context}`

Question template (With Few shots).

> Here are some Examples:
> Question: "Which projects does [Person] belong to?"
> Answer: "[Project1], [Project2]"
> Question: "Which role does [Person] have?"
> Answer: "[Role1], [Role2]"
> Question: "Which departments is [Person] part of?"
> Answer: "[Department1], [Department2]"
> Question: "What are [Person]'s projects' domains?"
> Answer: "[Domain1], [Domain2]"
> Question: "What are [Person]'s projects' started years?"
> Answer: "[Year1], [Year2]"
> Question: "Who sponsors [Person]'s projects?"
> Answer: "[Sponsor1], [Sponsor2]"
>
> Question: {question}

Compression prompt.

> You were given a task. Before I give you the question, imagine you are a student memorizing this material according to the task. Repeat the context concisely yet comprehensively, preserve all critical details, and create a cheat sheet to solve the task. Once done, I will give you the question.

B.2 Examples of Generated CPTs

Figures 13-17 present representative CPTs generated for HOTPOTQA, Passage Retrieval, TREC, Passage Count, and LCC. Notably, the model often structures its CPTs like concise *study notes*, summarizing key entities, events, and relationships, mirroring how a human might distill and organize information before solving the task.

C Additional Results

C.1 BM25 Baseline

For completeness, we compare CACHENOTES against a *sparse* retrieval baseline using BM25. We keep the same chunking and budget setup as in the main experiments. Table 3 reports performance on LLAMA-3.1-8B-INSTRUCT across *Single-Doc QA* and *Multi-Doc QA*. We observe that BM25 slightly **outperforms the dense retriever for Single-Doc QA** (at all budgets), whereas the dense retriever **surpasses BM25 for Multi-Doc QA**. This suggests that term-based retrieval can remain competitive on short, self-contained questions, but dense retrievers are better at capturing semantic links across multiple documents. CACHENOTES consistently outperforms the best *query-aware* retriever (BM25 or dense) across all tasks and budgets, with especially large gains at lower token budgets.

HotpotQA

> To aid memorization, I will summarize the passages concisely yet comprehensively, preserving all critical details.
> **Passage 1: John Macklin**
>
> - John Macklin (1883-1949) was an American football player, coach, and athletics administrator.
> - He was the head football coach at Michigan Agricultural College (now Michigan State University) from 1911 to 1915, with a 29-5 record and a .853 winning percentage, the highest in Michigan State history.
>
> [. . .]
> **Passage 2: Don Coleman**
>
> - Don Coleman (1928-2017) was an American football player and coach.
> - He played high school football at Flint Central High School and college football at Michigan State University.
>
> [. . .]
> [continues for all the passages...]
> Now that I have summarized the passages, I'm ready to answer your question. Please go ahead and ask it!

Figure 13

C.2 Task-Focused CPTs Prompt for Code Completion

Motivation. LongBench's LCC task asks the model to predict the next line of code from a long context. The default LLM_{instr} prompt produces a sequence of CPTs summarizing *all* preceding code, but it does not explicitly guide the model to prioritize the lines that are most useful for code completion. We therefore replaced that generic prompt with a variant *task-focused* that (i) instructs the model to memorize context specifically for code completion and (ii) provides an inline example of how to highlight the salient variables and methods. [1]

Revised prompt. The full text is reproduced below:

> Before solving the code completion task, imagine you are a student memorizing this material according to the task you will have to perform. Ensure you preserve all critical details to solve the task and create a cheat sheet covering the entire context. Once you have done that, I will prompt you to solve the code completion task.

[1]We keep the rest of the evaluation pipeline unchanged.

Passage Retrieval

To prepare for the task, I will create a cheat sheet covering the entire context. Here's a summary of the 30 paragraphs:
Paragraph 1-5: CBC and Canadian Broadcasting

- CBC had originally decided not to transition to digital, but the CRTC granted permission to continue operating 22 repeaters in mandatory markets until August 31, 2012.
- The remaining transmitters were shut down in 2012.

Paragraph 6-10: Schooly D and Saturday Night! – The Album

- Schooly D's album received mixed reviews, with some critics praising his harsh and confrontational style. [. . .]

[. . .]
[continue for all paragraphs...]
Now that I have created a cheat sheet covering the entire context, I am ready to solve the task. Please provide the abstract, and I will determine which paragraph it is from.

Figure 14

TREC

After analyzing the given examples, I have identified the following categories:
1. **Individual**: Questions that ask for a person's name, identity, or characteristics.

- Examples: "Who was the first American poet to win the Nobel Prize for literature, in 1948?", "Who is the current UN Secretary General?", "Who was the first woman in space?"
- Expected output: A person's name or identity.

2. **Number of something**: Questions that ask for a quantity or count of something.

- Examples: "How many Americans fought for the British in the American Revolution?", "How many innings constitute an official baseball game?", "How many zeros are there in a trillion?"
- Expected output: A numerical value.

[. . .]
[continue for all the type of questions...]
Now, please provide the new question, and I will determine the type of question based on the categories above.

Figure 15

Since your task is code completion, write a cheat sheet tailored to help you write the correct next line(s) of code.

Highlight or list the specific lines or variables in the context that are most relevant for this task.

For example:

```
public class Example {
    private int count;
    public void increment() {
        count++;
    }
    public int getCount() {
```

Cheat sheet (for next line):

- We are in `getCount()`, which should return the current value of `count`.
- `count` is a private integer field.
- Convention: getter methods return the corresponding field.
- Relevant lines: declaration of `count`, method header

Next line will likely be:

```
return count;
```

Results. Table 4 reports automatic scores (higher is better) for three retrieval-token budgets. Injecting the task-focused prompt yields absolute gains of 11, 6, 1.5 points for 512-1024-2048 budgets, eliminating the gap with other methods for this high-locality tasks and confirming that carefully engineered prompts can help the model compress long contexts into more actionable cues.

D Company Data Annotation

To complement automatic grading, we conducted a human pairwise evaluation on the COMPANY NOTES dataset. For each question, the annotator was shown:

- the user question,
- a reference document containing the relevant company notes,
- two candidate answers (**A** and **B**), without knowing which system (RAG or CACHENOTES) produced them.

The annotator could consult the reference document to assess factual accuracy and contextual relevance, then selected the answer judged *better overall*. The evaluation was performed by a sin-

Passage Count

Based on the provided paragraphs, I have identified the following key points to create a cheat sheet:
Paragraph 1: Gun Violence in the U.S.

- The U.S. is ranked 4th out of 34 developed nations for the highest incidence rate of homicides committed with a firearm.
- Mexico, Turkey, and Estonia are ranked ahead of the U.S. in incidence of homicides.
[. . .]

Paragraph 2: Cromwell's Coaching Career

- Cromwell served as the head coach of the USC Trojans football program from 1909 to 1910 and from 1916 to 1918.
- He officiated USC games as early as 1903 and played in a 1905 game against Harvard School.
[. . .]

[. . .]
[continue for all paragraphs...]
Now that I have created a cheat sheet, I am ready to solve the task of counting non-repeating paragraphs.

Figure 16

gle annotator: a PhD student in computer science (NLP background, fluent in English, Europe-based) familiar with the company's domain and internal terminology.

Annotation Instructions. Participants received the following task description:

> You are evaluating answers to technical support questions based on internal company documentation. For each example, you will be shown:
>
> - a question,
> - a reference document (company notes),
> - two candidate answers: Answer A and Answer B.
>
> Your task is to compare the two answers and decide which one is better overall, using the company notes to verify accuracy.

Results. In this blind evaluation, CACHENOTES was preferred in **58.5%** of cases, closely matching the **62.7%** preference observed with the LLM-as-judge. This positive correlation suggests that CACHENOTES better preserves contextually relevant information, although its broader coverage sometimes allowed RAG's more concise, pragmatic answers to be favored by humans.

LCC

Based on the provided DuelContext class, I will create a cheat sheet to help with code completion:
Properties and Fields:

- m_Initiator: The mobile that initiated the duel.
- m_Participants: An ArrayList of participants in the duel.
[. . .]

Methods:

- CantDoAnything(Mobile mob): Returns whether a mobile can perform actions during the duel.
- IsFreeConsume(Mobile mob): Returns whether a mobile can consume items during the duel.
[. . .]

[. . .]
[continue for all methods...]
Now that I have created a cheat sheet, I'm ready to help with code completion. Please provide the task you would like me to complete.

Figure 17

Ethics. The annotator's institution approved the data collection protocol.

E Reproducibility

All experiments were executed with a fixed random seed (42) to ensure deterministic results. We release ready-to-run shell scripts and the processed datasets used in each experiment.

E.1 LongBench

We evaluate on the LONGBENCH benchmark with both LLAMA-3.1-8B-INSTRUCT and QWEN3-4B-INSTRUCT-2507. Tables and figures: Figures 4, 5, 9 in the main paper; Tables 3, 4 in the appendix. The complete results are available in Tables 5, 6.

Scripts.

- `longbench_llama.sh`: Llama experiments (Figure 4)
- `longbench_qwen.sh`: Qwen experiments (Figure 5)
- `longbench_cachenotes_llama_s.sh`: CPTs transferability (Figure 9)
- `longbench_bm25_llama.sh`: BM25 retriever (Table 3)
- `longbench_cachenotes_llama_lcc.sh`: One-shot CPTs (Table 4)

Single-Doc QA	512	1024	2048
RAG-BM25 (query-aware)	**26.82**	**32.34**	**39.29**
RAG-DENSE (query-aware)	26.42	31.77	38.67
CACHENOTES (query-agnostic)	**34.00**	**35.96**	**39.60**
Δ vs. best query-aware	**+7.18**	**+3.62**	**+0.31**
Multi-Doc QA	**512**	**1024**	**2048**
RAG-BM25 (query-aware)	24.61	34.25	41.31
RAG-DENSE (query-aware)	**29.19**	**37.75**	**43.01**
CACHENOTES (query-agnostic)	**38.91**	**41.67**	**44.71**
Δ vs. best query-aware	**+9.72**	**+3.92**	**+1.70**

Table 3: Performance of CACHENOTES compared to sparse (BM25) and dense retrieval RAG baselines on LLAMA-3.1-8B-INSTRUCT. Scores are F1.

Table 4: Impact of task-focused CPTs prompt on LCC.

Method	512	1024	2048
LCC (default prompt)	39.53	46.20	51.06
+ Task-focused prompt	**50.65**	**52.35**	**52.54**

Datasets.

- Core: LongBench
- RAG-preprocessed: LB-512, LB-1024, LB-2048
- Qwen variant: LongBench-Qwen
- CPTs transfer: LongBench-S
- BM25-preprocessed: BM25-512, BM25-1024, BM25-2048
- One-shot CPTs: LongBench-LCC

E.2 RULER

Figure 10 in the main paper. The complete results are available in Tables 7, 8.

Script.

- `ruler.sh`: Reproduces RULER experiments

Datasets.

- Core: RULER
- Qwen variant: RULER-Qwen

E.3 Synthetic Dataset

Figure 7. The complete results are available in Table 10.

Script.

- `run_synthetic_dataset.sh`

Datasets.

- Core: Synthetic Dataset
- RAG-preprocessed: 256, 512, 1024, 2048

E.4 Company Notes

Figure 6 in the main paper.

Script.

- `company_notes.sh`

Datasets.

- Core: Company Notes
- Qwen variant: Company Notes-Qwen
- RAG-preprocessed: 512, 1024

F Information About Use Of AI Assistants

In the preparation of this manuscript, we used an AI assistant to aid in coding and text rewriting.

G Results on QWEN3-32B

To verify the effectiveness of CACHENOTES on larger parameter scales (30B+), we evaluated QWEN3-32B on the RULER benchmark. Table 9 and Figure 18 demonstrate that the performance trends observed in smaller models hold for the 32B model: CACHENOTES consistently outperforms EXPECTED ATTENTION and remains competitive with the query-aware FINCH, particularly at higher compression rates. This confirms that our task-aware, query-agnostic compression strategy scales effectively to larger architectures without requiring modification.

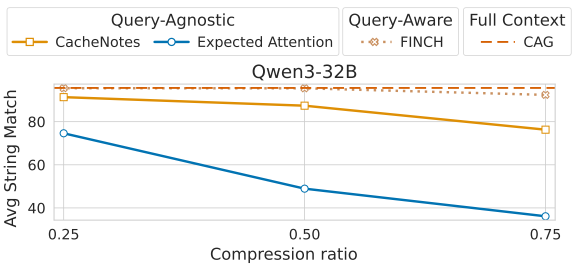


Figure 18: Results on RULER: CACHENOTES narrows the gap between query-aware and query-agnostic KV compression.

Table 5: LongBench Results across task categories and datasets for LLAMA-3.1-8B-INSTRUCT. Higher is better.

		Single-Doc QA			Multi-Doc QA			Summarization			Few-shot Learning			Synthetic Tasks		Code Completion	
		NarrativeQA	Qasper	MultifieldQA	HotpotQA	WikiMQA	MuSiQue	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PassageCount	PassageRetrieval	LCC	RepoBench-P
Full Context																	
Full Context	CAG	30.60	47.64	55.62	59.14	52.56	33.09	35.02	24.92	27.00	30.50	84.31	38.78	11.70	100.00	51.80	44.28
Budget: 512 tokens																	
Query-Aware	FINCH	27.66	38.44	51.67	52.59	48.41	26.83	26.55	23.23	23.73	1.00	85.98	16.62	13.17	99.00	54.17	45.59
	RAG	14.63	27.98	36.65	38.92	30.74	17.90	24.98	20.74	24.40	60.00	85.82	28.61	1.50	63.50	35.73	40.32
Query-Agnostic	CacheNotes	26.39	33.69	41.93	50.92	41.79	24.01	32.75	22.55	26.52	47.50	89.21	30.92	13.50	39.50	39.53	43.79
	Expected Attention	21.78	30.30	33.43	43.86	29.63	17.38	28.25	21.31	25.96	42.00	91.02	19.55	9.00	5.50	47.13	49.22
	KVZip	17.13	30.41	34.68	38.02	26.81	13.98	25.23	21.68	25.66	6.50	86.64	36.05	11.00	18.00	51.45	49.33
Budget: 1,024 tokens																	
Query-Aware	FINCH	28.91	41.21	53.13	54.86	50.48	26.62	28.47	23.83	25.86	1.00	86.98	20.87	10.98	99.00	53.57	46.15
	RAG	18.71	31.08	45.52	48.87	41.19	23.19	28.32	21.82	25.98	55.50	87.62	26.12	1.00	71.50	42.37	39.32
Query-Agnostic	CacheNotes	24.39	37.65	45.83	52.77	44.88	27.37	33.70	23.31	26.61	51.00	87.93	31.09	11.18	71.50	46.20	43.70
	Expected Attention	23.74	37.46	36.28	44.60	42.05	19.98	29.86	21.81	26.85	43.50	91.17	26.72	9.50	15.50	53.55	49.47
	KVZip	20.35	34.77	44.47	43.85	28.91	16.87	27.60	22.29	26.30	11.00	86.36	37.18	6.12	39.00	55.92	48.68
Budget: 2,048 tokens																	
Query-Aware	FINCH	30.88	44.60	55.50	57.54	51.60	28.14	30.73	24.42	26.64	7.00	89.46	32.15	11.67	99.00	53.09	45.22
	RAG	23.18	40.08	52.74	54.17	44.92	29.93	30.94	23.20	26.60	49.50	88.99	18.32	8.50	86.00	47.72	42.14
Query-Agnostic	CacheNotes	27.18	42.56	49.07	53.55	51.70	28.88	33.86	23.42	27.32	37.50	89.36	36.19	12.50	91.00	51.06	43.77
	Expected Attention	27.32	45.67	42.59	51.79	45.93	25.90	31.78	23.13	27.02	26.50	90.49	35.01	6.50	29.50	54.19	48.98
	KVZip	22.17	44.15	52.08	49.44	46.12	18.54	30.05	23.89	26.95	34.00	86.24	37.46	10.00	62.50	55.13	48.17

Table 6: LongBench Results across task categories and datasets for QWEN3-4B-INSTRUCT-2507. Higher is better.

		Single-Doc QA			Multi-Doc QA			Summarization			Few-shot Learning			Synthetic Tasks		Code Completion	
		NarrativeQA	Qasper	MultifieldQA	HotpotQA	WikiMQA	MuSiQue	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PassageCount	PassageRetrieval	LCC	RepoBench-P
Full Context																	
Full Context	CAG	30.45	43.71	47.73	52.21	37.03	18.56	30.27	22.71	23.97	76.50	84.97	40.13	3.10	92.58	62.59	54.37
Budget: 512 tokens																	
Query-Aware	FINCH	24.48	35.65	42.79	39.44	31.37	12.65	23.93	22.60	21.36	31.75	82.52	37.78	2.75	65.85	58.55	49.03
	RAG	12.39	27.63	36.16	36.32	29.94	12.02	23.53	20.71	22.04	67.50	85.68	35.61	1.00	62.68	40.02	43.37
Query-Agnostic	CacheNotes	19.86	28.65	39.06	30.90	26.78	6.80	29.02	20.90	23.71	75.25	84.48	39.21	1.98	71.03	46.51	47.53
	Expected Attention	10.69	24.94	23.39	12.49	14.44	2.69	26.15	19.70	23.34	66.25	80.05	36.09	4.22	6.67	48.63	51.17
	KVZip	10.74	26.86	28.35	12.45	16.87	2.27	24.07	20.40	23.03	19.50	83.74	34.30	4.63	12.86	55.82	55.21
Budget: 1,024 tokens																	
Query-Aware	FINCH	26.82	40.04	46.85	43.53	36.44	14.51	26.61	22.85	23.45	46.50	82.76	38.05	2.35	82.81	61.20	50.42
	RAG	16.52	32.36	43.31	41.65	34.34	14.77	25.54	21.73	23.17	71.50	85.87	37.18	1.91	69.62	48.27	42.35
Query-Agnostic	CacheNotes	25.05	35.84	40.49	38.03	33.11	11.65	29.72	21.30	23.83	77.00	84.19	38.35	1.91	80.39	53.77	48.22
	Expected Attention	12.33	32.85	29.83	18.63	23.26	4.41	27.91	20.04	23.80	72.75	82.26	37.02	2.09	7.92	56.77	51.25
	KVZip	12.93	35.88	33.16	14.65	20.52	3.27	26.56	20.77	23.58	43.00	84.84	35.96	2.35	22.33	59.70	54.92
Budget: 2,048 tokens																	
Query-Aware	FINCH	28.32	42.68	46.66	48.21	38.26	16.69	28.75	22.65	23.66	66.00	83.90	39.09	1.25	92.68	62.31	54.44
	RAG	19.95	35.84	48.00	47.40	36.17	18.50	27.05	21.77	23.62	74.00	85.50	37.32	2.03	82.00	56.30	44.69
Query-Agnostic	CacheNotes	25.67	39.99	44.01	37.69	36.00	18.02	29.92	21.73	23.91	76.50	84.79	38.45	1.36	85.60	59.93	52.08
	Expected Attention	15.77	38.27	35.55	30.31	29.88	6.42	29.28	20.93	23.95	74.75	83.59	38.43	2.44	13.62	61.61	52.71
	KVZip	19.21	40.70	43.45	26.21	27.23	4.99	28.59	21.85	23.91	63.75	85.74	37.50	2.88	39.29	63.29	54.35

Table 7: RULER results across datasets and compression ratios for LLAMA-3.1-8B-INSTRUCT. Higher is better.

		cwe	fwe	niah_multik_1	niah_multik_2	niah_multik_3	niah_multiq	niah_multiv	niah_single_1	niah_single_2	niah_single_3	qa_1	qa_2	vt
Full Context														
Full Context	CAG	99.62	94.87	99.80	100.00	99.80	99.90	99.90	100.00	100.00	100.00	87.80	62.40	99.88
Compression Ratio: 0.25														
Query-Aware	FINCH	99.30	94.33	98.80	87.00	95.60	99.55	98.80	100.00	100.00	99.80	87.00	62.40	99.96
Query-Agnostic	CacheNotes	99.52	95.20	100.00	93.80	76.80	99.65	99.70	100.00	97.80	80.00	82.80	62.60	99.92
	Exp. Attention	99.64	95.47	99.20	98.80	92.40	94.05	94.05	99.40	100.00	32.00	86.80	60.80	97.44
Compression Ratio: 0.50														
Query-Aware	FINCH	97.44	91.33	98.60	68.20	82.60	99.10	96.55	100.00	99.20	91.60	87.60	61.60	99.92
Query-Agnostic	CacheNotes	99.52	95.00	98.00	72.00	72.40	99.20	98.80	100.00	89.60	79.40	74.40	58.20	99.92
	Exp. Attention	99.30	95.93	95.40	87.60	28.00	82.45	86.60	97.40	94.00	4.60	80.60	58.00	88.48
Compression Ratio: 0.75														
Query-Aware	FINCH	87.06	85.33	97.60	45.60	14.40	96.85	89.45	100.00	96.20	28.80	86.20	60.80	99.64
Query-Agnostic	CacheNotes	96.92	92.53	83.60	49.00	13.60	97.50	97.95	98.80	68.80	79.00	59.80	54.40	99.80
	Exp. Attention	92.22	91.47	61.20	30.40	0.40	43.50	48.05	96.00	78.40	0.20	66.40	51.40	66.88

Table 8: RULER results across datasets and compression ratios for QWEN3-4B-INSTRUCT-2507. Higher is better.

		cwe	fwe	niah_multik_1	niah_multik_2	niah_multik_3	niah_multiq	niah_multiv	niah_single_1	niah_single_2	niah_single_3	qa_1	qa_2	vt
Full Context														
Full Context	CAG	96.00	87.80	100.00	100.00	99.80	100.00	99.75	100.00	100.00	100.00	84.20	62.00	99.96
Compression Ratio: 0.25														
Query-Aware	FINCH	95.90	86.80	100.00	100.00	99.80	99.90	99.55	100.00	100.00	100.00	82.60	61.20	99.96
Query-Agnostic	CacheNotes	94.88	87.80	100.00	99.60	58.20	100.00	99.65	100.00	100.00	100.00	78.60	60.40	99.96
	Exp. Attention	95.10	89.67	60.60	99.00	43.80	54.60	41.50	100.00	75.60	0.00	80.60	60.80	99.88
Compression Ratio: 0.50														
Query-Aware	FINCH	83.94	85.13	100.00	95.60	99.20	100.00	97.00	100.00	100.00	97.80	83.60	59.80	99.96
Query-Agnostic	CacheNotes	84.20	87.07	100.00	66.40	51.60	99.50	99.20	100.00	100.00	100.00	67.20	58.40	99.96
	Exp. Attention	86.82	89.73	13.80	49.60	2.80	7.25	6.60	95.00	24.00	0.00	70.20	58.40	91.72
Compression Ratio: 0.75														
Query-Aware	FINCH	54.00	82.47	100.00	48.20	43.20	98.75	86.90	100.00	100.00	24.40	81.20	58.40	99.92
Query-Agnostic	CacheNotes	72.54	85.60	98.20	30.80	17.40	97.35	96.40	100.00	99.80	100.00	57.00	55.40	99.96
	Exp. Attention	43.04	86.53	0.00	3.00	0.00	0.05	0.00	47.00	2.60	0.00	47.00	44.40	13.96

Table 9: RULER results across datasets and compression ratios for QWEN3-32B. Higher is better.

		cwe	fwe	niah_multikey_1	niah_multikey_2	niah_multikey_3	niah_multiquery	niah_multivalue	niah_single_1	niah_single_2	niah_single_3	qa_1	qa_2	vt
Full Context														
Full Context	Full	99.64	92.73	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	85.80	65.20	100.00
Compression Ratio: 0.25														
Query-Aware	FINCH	99.50	92.00	100.00	100.00	99.80	100.00	100.00	100.00	99.80	100.00	86.00	64.60	100.00
Query-Agnostic	CacheNotes	99.56	92.47	100.00	98.80	74.00	100.00	100.00	100.00	99.60	100.00	82.20	64.20	100.00
	Expected Attention	99.58	91.87	59.20	100.00	85.00	61.50	60.80	98.80	62.80	1.00	83.80	66.00	99.52
Compression Ratio: 0.50														
Query-Aware	FINCH	99.26	91.87	100.00	99.60	100.00	100.00	99.95	100.00	99.80	99.60	86.20	64.80	100.00
Query-Agnostic	CacheNotes	99.60	93.33	100.00	69.80	57.40	100.00	100.00	100.00	99.80	100.00	71.40	59.20	100.00
	Expected Attention	98.62	92.13	3.00	79.80	23.00	2.20	2.95	85.80	9.00	0.00	77.80	62.60	99.20
Compression Ratio: 0.75														
Query-Aware	FINCH	94.46	87.27	100.00	96.60	97.80	99.95	97.65	100.00	100.00	79.00	84.80	64.00	99.88
Query-Agnostic	CacheNotes	99.34	89.13	100.00	36.20	21.60	100.00	99.60	100.00	99.60	100.00	53.80	55.60	100.00
	Expected Attention	80.62	87.87	0.00	10.60	0.60	0.00	0.00	89.60	0.00	0.00	58.60	54.40	87.28

Table 10: Connectivity-wise results for Direct and Join-like Queries for LLAMA-3.1-8B-INSTRUCT. Higher is better.

	Direct Retrieval								Join-like							
	phase1	phase2	phase3	phase4	phase5	phase6	phase7	phase8	phase1	phase2	phase3	phase4	phase5	phase6	phase7	phase8
Full Context																
CAG	0.80	0.71	0.78	0.73	0.53	0.67	0.65	0.52	0.40	0.35	0.29	0.23	0.48	0.58	0.49	0.45
Budget: 256 tokens																
RAG	0.00	0.00	0.01	0.01	0.03	0.06	0.02	0.05	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.02
CacheNotes	0.56	0.55	0.39	0.22	0.25	0.33	0.20	0.39	0.25	0.23	0.31	0.34	0.39	0.16	0.41	0.45
Budget: 512 tokens																
RAG	0.00	0.00	0.03	0.08	0.09	0.08	0.08	0.09	0.03	0.05	0.01	0.05	0.04	0.03	0.00	0.03
CacheNotes	0.63	0.72	0.51	0.28	0.41	0.48	0.44	0.53	0.34	0.38	0.33	0.31	0.48	0.37	0.59	0.68
Budget: 1,024 tokens																
RAG	0.41	0.45	0.56	0.56	0.54	0.47	0.50	0.45	0.08	0.20	0.15	0.13	0.23	0.10	0.14	0.13
CacheNotes	0.56	0.69	0.57	0.32	0.44	0.55	0.47	0.43	0.31	0.38	0.36	0.38	0.56	0.46	0.54	0.59
Budget: 2,048 tokens																
RAG	0.75	0.78	0.72	0.74	0.71	0.75	0.73	0.70	0.25	0.19	0.32	0.24	0.43	0.35	0.31	0.31
CacheNotes	0.53	0.68	0.53	0.56	0.48	0.55	0.56	0.55	0.31	0.34	0.41	0.37	0.56	0.51	0.56	0.59