CACHENOTES: Offline Task-Aware KV Cache Compression for Reasoning-Intensive Knowledge Tasks

Anonymous ACL submission

Abstract

Integrating external knowledge into Large Language Models (LLMs) is vital, yet methods like Retrieval-Augmented Generation (RAG) face limitations with broad queries requiring multi-source synthesis, while long-context models are computationally prohibitive.

We propose Task-Aware Key-Value Cache Compression (CACHENOTES), a novel *query-agnostic* framework that generates a reusable, compact cache tailored to a specific task. Unlike prior approaches, we first create a task-specific 'cheat-sheet' summary that guides a one-time compression of the corpus into a reusable KV-cache. This enables LLMs to efficiently answer diverse, reasoning-intensive queries using the compressed cache, eliminating repeated retrieval or context expansion.

Experiments on LongBench show that CACHENOTES outperforms standard RAG by up to 4 F1 points at a 20× compression rate, and delivers up to 4× lower latency, while remaining competitive with state-of-the-art query-aware baselines. Additional results on real-world enterprise and synthetic datasets demonstrate that CACHENOTES is especially effective for multi-hop and broad-coverage queries.

1 Introduction

Incorporating external information into Large Language Models (LLMs) significantly enhances their utility across various applications, enabling them to generate more informed and accurate outputs (Petroni et al., 2019).

While Retrieval-Augmented Generation (RAG) efficiently inject relevant evidence for narrow, focused queries, they often struggle with questions that require synthesizing information dispersed across multiple documents or sections of text (Barnett et al., 2024). Such multi-hop or broad queries are bottlenecked by retrieval's reliance on local similarity and its inability to capture global relationships or latent links in the corpus, making it challenging to surface all relevant context and often introducing noise or redundancy (Yu et al., 2024).

Recent advancements have extended LLMs' ability to process longer contexts (Reid et al., 2024; Li et al., 2025a). This progress opens up the possibility of processing entire corpora as input, offering a compelling alternative for tasks requiring holistic understanding. However, this approach comes with significant computational costs, as handling large inputs requires substantial memory resources, particularly on GPUs, which creates a scalability bottleneck (Liu et al., 2023). Furthermore, as context length increases, models can struggle to discern and use the relevant information buried within extensive text (Liu et al., 2024a; 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 **Task-Aware Key-Value Cache Compression (CACHENOTES)**, a query-agnostic yet task-guided framework that creates a reusable, compact cache for an entire corpus. Unlike prior work in query-aware cache compression, which recomputes a compressed context for each query (Rehg, 2024; Xu et al., 2025; Li et al., 2025b; Corallo and Papotti, 2024) or generic, prompt-based methods (Kim et al., 2024), our approach performs a one-time, offline compression directed by a concise, human-written task description and a model-generated "cheat-sheet" that distills corpus content most critical for that task. This cheat-sheet guides cross-attention-based token selection, producing a persistent cache that can be rapidly leveraged by the LLM

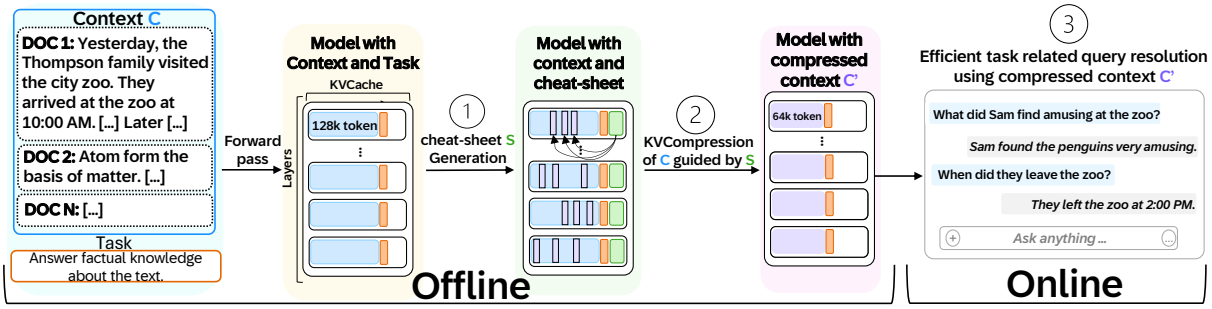

Figure 1: An illustration of our compression strategy. (1) A cheat-sheet S is generated from the corpus C, guided by a task description (e.g., factual QA). (2) The original KV cache (here, 128k tokens) is then compressed (e.g., to 64k tokens) using the cheat-sheet S to retain the most salient information. (3) At inference time, the LLM can answer task-related questions using only the pre-compressed cache, as if it had access to the entire (uncompressed) corpus.

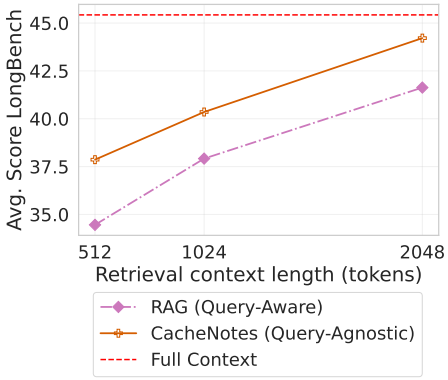

Figure 2: Aggregated results on 16 datasets of LongBench of KV cache compression methods compared to RAG. Our cheatsheet-guided KV compression approach achieves results exceeding RAG when the context length is much smaller than average corpus size of 10k tokens.

at inference, with no retrieval or recompression.

Figure 1 provides an overview of our approach. The process, initiated by the task description and guided by the generated cheat-sheet, produces a compact KV cache representation that retains all task-critical details. Importantly, this compression happens only once offline, creating a persistent representation that can be reused for any query within that task's domain. The method can be applied to any decoder-only model and applicable to a variety of knowledge-intensive tasks, without model fine-tuning. Examples of generated cheat-sheets and task descriptions are in Appendix B.

Our method, delivers performance that significantly surpasses existing query-agnostic compression techniques and closely approaches the effectiveness of more computationally intensive query-aware compression. Figure 2 shows the quality performance of our KV cache compression method against RAG on the LongBench benchmark (Bai et al., 2024). With high compression rates 20x and 10x and 5x (corpus compressed to 512, 1024 and 2048 tokens, respectively) our method outperforms RAG despite being query-agnostic. With compression rate of 5x, our method achieves about 97.3% of the full-context performance, demonstrating remarkable information retention.

Experimental results across diverse tasks using LLAMA 3.1 (Dubey et al., 2024) and QWEN 2.5 (Yang et al., 2024), on the LongBench benchmark, demonstrate that our task-aware compression method consistently outperforms existing query-agnostic methods and retrieval-based approach. Furthermore, experiments on real world enterprise data and custom synthetic datasets highlight the superior capability of our method in handling broad, multifaceted queries. Notably, in scenarios requiring the synthesis of widely distributed information, our approach significantly outperforms RAG, establishing compression as a key enabler for scaling LLM reasoning beyond retrieval-based methods.

2 Problem Formulation

The primary challenge this work addresses is enabling LLMs to efficiently and effectively perform reasoning-intensive knowledge tasks that require access to extensive external corpora (Petroni et al., 2021; Su et al., 2024a). As discussed in Section 1, prevailing methods present a difficult trade-off: RAG is efficient but struggles to synthesize dispersed evidence for broad, multi-hop queries, while processing entire corpora with long-context models offers comprehensiveness but at a prohibitive computational and memory cost. KV cache compression is a promising method to create compact representations of large contexts, offering a solution to this trade-off (Li et al., 2024; Liu et al., 2024b). However, existing compression strategies themselves introduce a critical dilemma regarding

their applicability and performance.

Ideally, we would like to pre-digest the entire corpus $\mathcal{D}$ *once*, distilling it into a compact, task-aware key–value cache $(\widetilde{\mathbf{K}}, \widetilde{\mathbf{V}})$ that can be plugged into the model for *any* downstream query. Such a query-agnostic cache would remove the need for retrieval or on-the-fly compression altogether: inference would amount to a single forward pass over the prompt, with no dependence on corpus size and latencies close to those of small-context generation.

A fundamental challenge in compressing long-context representations, however, is achieving precisely this *query-agnostic* compression. Empirical results show that existing methods degrade sharply—especially at high compression ratios—often falling behind both full-context processing and RAG (Jegou et al., 2024).

Conversely, *query-aware* compression shrinks the KV footprint by guiding the compression with the user query (Corallo and Papotti, 2024; Li et al., 2025b), offering potentially superior efficiency at the cost of increased query dependency. In fact, in multi-query settings, repeatedly running the compressor is computationally prohibitive, undermining the very goal of avoiding large-scale retrieval or excessive context expansion.

Research Question. *Can we design a query-agnostic compression method that retains the efficiency benefits of a precomputed cache while delivering 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. We describe how we obtain a single, reusable cache that covers an entire task's external knowledge.

3.1 Motivation via In-Context Learning

Modern LLMs are capable of remarkable in-context learning (Brown et al., 2020; Dong et al., 2024): given a sufficiently rich prefix, they infer the intended task and relevant knowledge with no parameter updates. To clarify the distinction between retrieval and compression strategies, we use a pedagogical analogy:

RAG, for instance, is akin to a student accessing reference material through a search engine *during* the test, retrieving snippets as needed for each question. Long-context models are like students who have memorized the entire corpus beforehand, leveraging this extensive internal knowledge to answer any question. Query-aware compression is more targeted, analogous to a student who scans their material *during* the test, focusing only on the most relevant information for a particular query.

However, students typically prepare for exams by organizing their knowledge *before* test time, often creating concise ""cheat-sheets"" that capture the essential concepts and facts, allowing them to quickly recall key points during the test. We hypothesize that a task-specific, pre-computed summary ("cheat-sheet") can serve as an effective scaffold for compressing external corpora into compact, reusable representations. Unlike prior query-aware methods that require a new compressed cache for each query, our method decouples cache construction from inference, amortizing compression cost across all queries for a given task and corpus.

3.2 Task-Aware KV Compression

We propose a task-aware KV compression approach consisting of three distinct stages:

(1) Creation of the Cheat-sheet (Offline) Given a full corpus C and a task description T (e.g., "open-domain QA"), we use an instruction-following language model LLM_{inst} to produce a concise summary S.

$$S = \text{LLM}_{\text{inst}}(C, T) \tag{1}$$

The model is prompted: *Before I give you the question, imagine you are a student memorizing this material according to the task you will perform (specified in Task Description). Repeat the context concisely yet comprehensively to aid memorization, preserving all critical details. Create a cheat sheet covering the entire context.*

Task descriptions can be customized or augmented with few-shot demonstrations to steer the LLM_{inst} towards generating a cheat-sheet S that is optimally aligned with the task's requirements.

(2) Layer-wise KV Compression (Offline) Context and cheat-sheet are concatenated together:

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

where n is the token length of the long context and m is the token length of the cheat-sheet. This concatenated input is processed by the LLM and at each attention layer, the guiding principle is that the cheat-sheet $\mathbf{S}$, embodying the distilled essence

of the corpus for task $\mathbf{T}$, provides query vectors $\mathbf{Q_s}$ that probe the context $\mathbf{C}$ to identify its most relevant segments. To achieve this, we extract the query vector of the cheat-sheet tokens Q_s and the key vector of the context tokens K_c, then compute the Cross-attention between them[1]:

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

To ensure that the attention scores reflect both the alignment and the relative importance of the cheat-sheet values, the raw Cross-attention scores $\mathbf{W}^{(S,C)}$ are scaled by the magnitude (norm) of the value vectors $\mathbf{V_s}$ associated with the cheat-sheet tokens. This step weights the attention scores, ensuring that cheat-sheet tokens with stronger value representations (higher norms) exert a proportionally greater influence on identifying salient context tokens (Jegou et al., 2024).

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

We then select the top indices from the cache of C, corresponding to the highest attention scores:

$$indices = \text{TopK}\left(\text{avg}(\mathbf{W}^{(S,C)}_{norm}), k\right) \quad (5)$$

where $\text{avg}(\cdot)$ denotes the aggregation operation over the cheat-sheet token dimension for each context token.

From the full Key ($\mathbf{K_C}$) and Value ($\mathbf{V_C}$) vectors of the original context $\mathbf{C}$ (derived from its processing through the LLM's layers), we gather the compressed key-value pairs $\widetilde{\mathbf{K}}, \widetilde{\mathbf{V}}$ corresponding to these selected $indices$. A rerotation step is applied to the positional embeddings of the key vectors in $\widetilde{\mathbf{K}}$ to ensure they maintain correct relative positional information in the new compressed sequence. In the case of Rotary Position Embeddings (Su et al., 2024b), this involves recalculating the corresponding sine and cosine components to reflect the relative token offsets accurately, ensuring that their positional information remains consistent (Xiao et al., 2024a; Corallo and Papotti, 2024).

(3) Efficient Query Resolution (Online) When facing a new question $\mathbf{q}_{\text{new}}$ from the same domain, at each attention layer, the $keys$ and $values$ of the question are appended to the precomputed cache:

$$\widetilde{\mathbf{K}} \leftarrow \begin{bmatrix} \widetilde{\mathbf{K}} \\ \mathbf{k}^{q_{\text{new}}} \end{bmatrix}, \quad \widetilde{\mathbf{V}} \leftarrow \begin{bmatrix} \widetilde{\mathbf{V}} \\ \mathbf{v}^{q_{\text{new}}} \end{bmatrix}. \quad (6)$$

[1]Cross-attention computation is tractable even for long contexts since its complexity is $O(mnd)$ which remains efficient in practice since m can be set such that $m << n$.

No further compression or retrieval is necessary. The LLM conditions on the updated $\widetilde{\mathbf{K}}$ and $\widetilde{\mathbf{V}}$ to answer, reusing relevant external knowledge. This is the only operation performed at inference time, making it efficient as it only involves a single forward pass for the typically short question and scales linearly with the size of the precomputed cache.

4 Experimental Setup

4.1 Models

Our experiments use LLAMA-3.1-8B-INSTRUCT (Dubey et al., 2024) and QWEN2.5-14B-INSTRUCT (Yang et al., 2024). The latter model is configured with a context length of up to 32,768 tokens. To handle inputs exceeding this limit, we employ the YaRN technique (Peng et al., 2024), as specified in their model card. We compare our method against two query-aware baselines, RAG and FINCH (Corallo and Papotti, 2024), as well as a query-agnostic baseline, EXPECTED ATTENTION (Jegou et al., 2024), and a Full Context upper bound. Additionally, for the query-agnostic setting, we performed an ablation study where the models were conditioned exclusively on the task description similar to the approach of "catalyst prompt" introduced by Kim et al. (2024), we refer to this baseline as CAP-G (TASK-SPECIFIC). We also included an additional baseline where the responses were generated solely based on the cheat-sheet, without applying corpus compression, to assess the effect of corpus compression on model performance.

For RAG, we use BGE-LARGE-EN-V1.5 (Xiao et al., 2024b) as the retriever, filling the entire context with the top-k retrieved chunks, each containing 256 tokens. The retrieved chunks are ordered according to their relative position within the original long document. To ensure a fair comparison with the other baselines if the concatenated chunks exceed the context length used in the other experiments, we truncate the input to match the maximum context size. This truncation is performed by taking the first half of the reduced context from the beginning and the remaining half from the end.

Greedy decoding is employed for all experiments (Vijayakumar et al., 2016; Shao et al., 2017) ($temperature = 1.0$, $top_p = 1.0$) both for cheat-sheet generation and answer generation. During cheat-sheet generation, the maximum number of newly generated tokens m is set to 2,048 using the prompt in 3.2 for all experiments. The same

model used for compression (either LLAMA-3.1 or QWEN2.5) is also used for cheat-sheet generation. Task description prompts T are provided in Appendix C. Code and scripts required to reproduce the results in the paper will be released upon publication.

4.2 Datasets and Metrics

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

LongBench (Bai et al., 2024) is a benchmark designed for long-context understanding. It covers 16 datasets across six tasks, including single-document and multi-document QA, summarization, code completion, few-shot learning, and a synthetic task. The benchmark has an average context length of 10k tokens. LongBench uses traditional evaluation metrics, such as F1 for question answering tasks (Rajpurkar et al., 2016), ROUGE-L for summarization tasks (Lin, 2004), and Edit Similarity for the Code Completion task (Svyatkovskiy et al., 2020). For the size of the answer output, we use the original values in the dataset.
Data Availability. The LongBench dataset is publicly accessible.

Company Notes Motivated by the need to evaluate KV Cache Compression on *real-world* data, we curate a industrial dataset composed of long technical documents drawn from enterprise support and troubleshooting portals. Each document is provided in raw HTML and contains configuration guides, best-practice notes, and resolution procedures. We selected 49 documents ranging from 16,205 to 31,906 tokens, with an average length of 22,401 tokens, measured using the LLAMA-3.1-8B-INSTRUCT tokenizer. Following the approach of Edge et al. (2024), we generate five *global queries* per document to ensure comprehensive coverage, resulting in 245 query-context pairs, *Global queries* are generated using prompts and configurations of Wei et al. (2025) using GPT-4.1-MINI. For the size of the answer output, we use 512 tokens.

To assess response quality, we adopt the grading framework proposed by Wei et al. (2025), which employs three core metrics:

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

We use GPT-4.1-MINI as the evaluator, comparing system outputs for each query in the test set. Each response is assessed against the target metric and original query. The evaluation prompt and configuration follow the approach of Wei et al. (2025), where the model is instructed to select the better candidate between two possible responses[2]. Our results show a strong correlation with human judgment, supporting the reliability of this automated evaluation approach.
Data Availability. 13 of the company notes used in this study are publicly available and the corresponding query-context pairs will be released.

Synthetic Dataset To assess the impact of inter-chunk connectivity on RAG and KV Cache Compression, 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 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, consistent with the approach used in LongBench. Full dataset construction details are provided in Appendix A. For the size of the answer output, we use 256 tokens.
Data Availability. Our Synthetic dataset will be released publicly upon publication.

5 Results and Discussions

We discuss four research questions:

1. When Does CACHENOTES Surpass RAG? Figures 3–4 show that CACHENOTES consistently matches or exceeds RAG on LongBench QA. For instance, on *Single-Doc QA* the method yields an average F1 that is 3-4 points higher than RAG at a

[2] Wei et al. (2025) also introduce a *User-Friendliness* metric. In our pre-experiments, most of the answers were judged 'not user-friendly' due to their technical tone, so we omit this dimension to focus on metrics relevant to our use case.

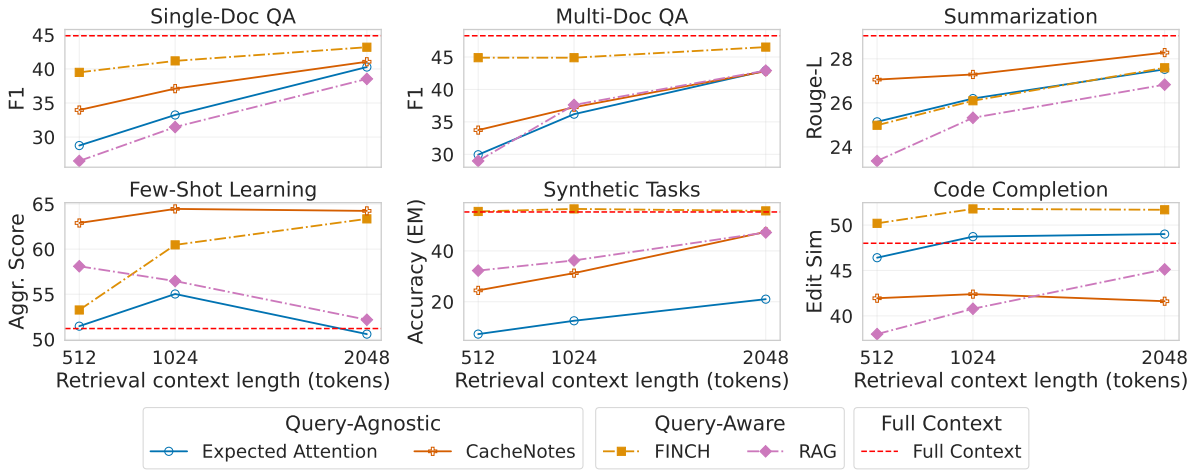

Figure 3: Performance results on Longbench for LLAMA-3.1-8B-INSTRUCT.

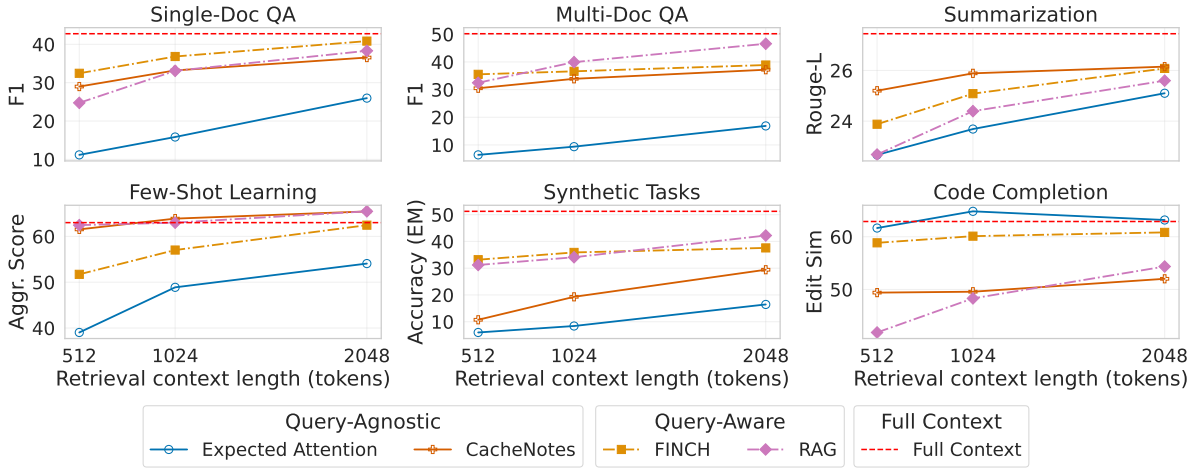

Figure 4: Performance results on Longbench for QWEN-2.5-14B-INSTRUCT.

1024-token retrieval budget for both Llama Qwen, while remaining competitive on *Multi-Doc QA*.

The advantage widens on our *Company Notes* corpus (Fig. 5), where technical documents require reasoning over long, densely interconnected prose. Here, human-graded helpfulness and insight scores favor CACHENOTES by 15–20 wins per aspect at 1024-token context, confirming that similarity search alone often overlooks dispersed evidence. This performance gain aligns with human judgments collected in our blind evaluation, where CACHENOTES was preferred 58.46% of the time, close to the 62.7% preference by the LLM as judge. This consistency indicates that CACHENOTES captures more contextually relevant information, although its surface-level coverage can lower human scores, at times allowing RAG's more pragmatic answers to win out.

To isolate this effect, we created a controlled synthetic set that contrasts *direct-retrieval* queries (answer lies in one chunk) with *join-like* queries (answer must be assembled across chunks). CACHENOTES again tracks full-context performance on join-like queries, whereas RAG's F1 falls by ≈12 points as connectivity grows (Fig. 6). These results echo the LongBench pattern: RAG thrives when a single chunk suffices, but loses recall when information is scattered.

> **Takeaway:** RAG remains a strong baseline for narrow, self-contained questions. CACHENOTES, however, is better suited to broader, multi-hop reasoning because it pre-computes a task-aware summary of the entire corpus, preserving links that retrieval may miss.

2. Why Is Compression Still Useful When a Cheat-Sheet Exists? We performed an ablation on the two LongBench QA tasks, comparing three settings: (i) generation directly from the cheat-sheet, (ii) CACHENOTES conditioned on that same cheat-sheet, and (iii) CAP-G (TASK-SPECIFIC)

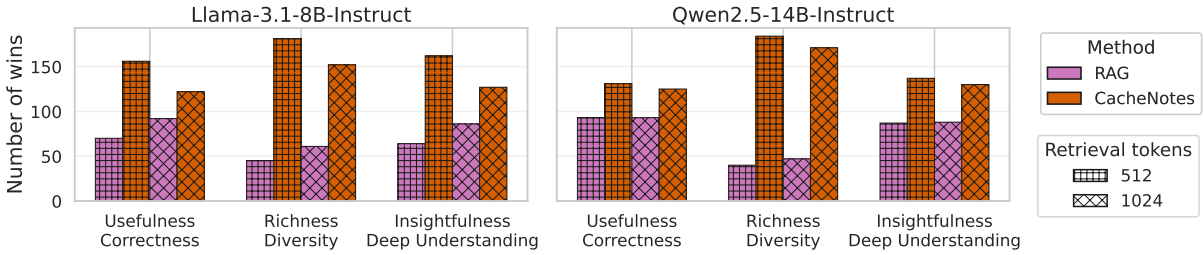

Figure 5: Performance results on Company Notes for LLAMA-3.1-8B-INSTRUCT and QWEN2.5-14B-INSTRUCT.

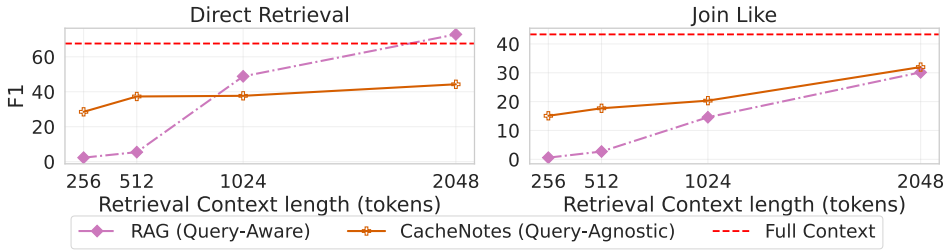

Figure 6: Performance results on Synthetic dataset for LLAMA-3.1-8B-INSTRUCT.

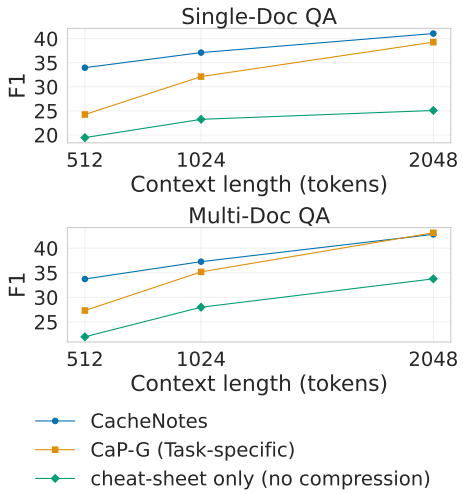

Figure 7: Ablation on KV Compression vs. using cheat-sheet only. KV vectors recalled from the cheat-sheet enhance downstream performance.

where compression is conditioned only by the task description. Figure 7 shows that using the cheat-sheet *alone* lags behind our compression by 4–7 F1 points across the 512-, 1024-, and 2048-token budgets in both Single-Doc and Multi-Doc QA. Even the lighter variant that sees only the task description surpasses the cheat-sheet baseline, underscoring that token-level, context-aware KV representations preserve more cues.

Takeaway: Pre-computing KV vectors yields richer signals than presenting the model with a standalone textual cheat-sheet, compression retains token-wise importance weights that summaries cannot capture.

3. Can CACHENOTES Rival Query-Aware Compression? Figures 3–4 reveal that, despite being query-agnostic, CACHENOTES matches or outperforms the query-aware method FINCH on two LongBench tasks. On *Summarization*, it scores 2–3 ROUGE-L points higher than FINCH across the 512–2048-token retrieval budgets, and on *Few-Shot Learning* it gains 4–6 exact-match points—surpassing even the full-context upper bound in several settings. We attribute this to the cheat-sheet's role as a globally coherent view of the corpus, which naturally benefits abstractive summarization and example-driven generalization.

The picture changes for *Code Completion*. Here, CACHENOTES trails the query-agnostic baseline *Expected Attention* by roughly 5 Edit-Similarity points, while FINCH remains competitive. Manual inspection suggests a task-mismatch: the generated cheat-sheets often summarize entire files rather than zooming in on the next-line prediction needed for completion. Providing a few-shot exemple that mirrors the "predict-the-next-line" objective—or an explicit task tag in the compression prompt—should help align the compressed representation with this highly local task.

Takeaway: CACHENOTES rivals (and sometimes surpasses) query-aware FINCH on tasks that reward a global view of the corpus. For locality-sensitive tasks such as code completion, users can steer cheat-sheet generation—e.g. by inserting task-specific examples into LLM_{inst} see Appendix 3.

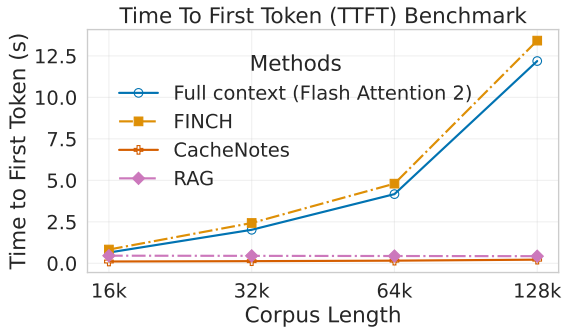

Figure 8: Time to first token with increasing corpus length (question length=512).

4. What Is the Impact on Practical Deployment? Figure 8 reports the *Online* time-to-first-token (TTFT) when the corpus grows from 16k to 128k tokens, under a fixed 8 k-token retrieval budget and a 512-token prompt (single-GPU, batch = 1). At 128k tokens, TTFT rises to 12.4 s for *Flash-Attention 2* full-context decoding and 13.3 s for FINCH, whereas RAG delivers the first token in 6.1 s. By contrast, CACHENOTES responds in 0.23 s, about 4× faster than RAG and 54× faster than the full-context baseline. Similar gaps hold because FINCH's query-time compression and Full Context's prefill scales roughly quadratically with corpus length, RAG requires retrieval and prefilling of the retrieved chunks which scales quadratically with the length of the chunks, while CACHENOTES reuses an offline-computed cache whose complexity scales linearly with the cache size.

Takeaway: Latency and memory savings make CACHENOTES a favorable drop-in for high-throughput, latency-sensitive workloads.

6 Related Work

Storing the full KV cache is a memory bottleneck in LLMs, scaling with model depth and input length. Recent work focuses on compressing this cache, broadly categorized as:

Query-agnostic compression methods compress the cache independently of downstream tasks, offering fast inference but potentially discarding useful context (Zhang et al., 2023; Devoto et al., 2024; Xiao et al., 2024a; Feng et al., 2025). These often select salient tokens using heuristics (e.g., attention weights) from the context itself and can struggle when optimal context heavily depends on the query.

Query-aware compression techniques compress the cache per example at inference time. This optimizes for each query, dynamically selecting critical tokens for higher quality, but incurs higher latency and limits scalability due to per-query computation (Rehg, 2024; Xu et al., 2025; Li et al., 2025b; Corallo and Papotti, 2024).

Our Method introduces a model-generated, task-aware "cheat-sheet" as an explicit proxy for what the model deems critical for a given task. This creates a middle ground: more flexible and contextually rich than prior query-agnostic approaches, yet far more efficient than query-aware methods by precomputing a reusable, task-specific cache.

Reasoning-Intensive Retrieval. Standard dense retrievers struggle with multi-hop reasoning in knowledge-intensive tasks, often missing crucial evidence spread across documents. Reasoning-aware retrieval methods (Shao et al., 2025; Wang et al., 2024; Trivedi et al., 2023) address this by coupling retrieval with model reasoning, but this increases inference complexity and often requires per-query recomputation. Our approach differs by decoupling reasoning from inference, shifting this effort offline by pre-compressing and distilling relevant knowledge.

7 Conclusion

We presented a task-aware compression approach that enhances the ability of LLMs to consume large corpora by efficiently populating the KV cache with condensed contextual representations.

While our work demonstrates significant promise, our experiments primarily use relatively small corpora. Expanding context window capabilities in open LLMs will make our approach increasingly relevant for realistic, large-scale corpora. Further research is needed to address other challenges before widespread deployment in production systems can be realized, particularly concerning the merging of compressed caches and the computational scalability of the compression process.

Future directions include head-wise and layer-wise compression, leveraging prior findings that some heads and layers are less critical and can be selectively compressed (Feng et al., 2024; Zhang et al., 2024). Additionally, our results highlight a complementary strength between KV compression (excels in broad queries) and RAG (more effective for narrow queries). This raises the question of whether a hybrid approach could further enhance retrieval: compressing the corpus offline for global coverage while dynamically fetching top-K chunks online to better address narrow queries.

Limitations

While our work demonstrates that large corpora can be effectively compressed into manageable KV caches for efficient LLM inference, there are important limitations that warrant further investigation.

Merging Compressed KV Caches In many real-world applications, knowledge is distributed across large or growing collections of documents. While our method enables efficient offline compression of individual document KV caches, merging these precomputed caches—so the model can reason seamlessly across multiple sources—remains an open problem. A core difficulty is that independently-compressed caches lack cross-document attention: simply concatenating caches does not recover dependencies between documents, and positional alignment becomes nontrivial. Recent works such as CacheBlend (Yao et al., 2025) address merging for uncompressed caches via selective recomputation, while methods like KVLink (Yang et al., 2025) propose positional alignment and bridging tokens for improved reuse. However, a robust and general approach for merging *compressed* caches remains unsolved. Promising future directions include (i) developing positional alignment or bridging-token strategies to restore cross-document connections; (ii) selective recomputation of boundary tokens in compressed caches; and (iii) adaptive merging schemes (e.g., clustering or reweighting similar keys) to reduce redundancy without degrading information fidelity.

Resource Requirements Another limitation concerns the scalability of our cross-attention-based compression. Our method relies on computing attention scores between the cheat-sheet and the entire corpus. In scenarios where either the context (n) or the cheat-sheet (m) becomes very large, the $O(mnd)$ cost can become memory hungry because its memory scales as $O(mn)$. In practice, this requires selecting a smaller m (a more aggressively summarized cheat-sheet), or partitioning the corpus into smaller, chunks (as in Corallo and Papotti (2024)), and then compressing each chunk depended with previous compressed chunk.

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. 2024. 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.

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.

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.

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. 2024. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. *CoRR*, abs/2407.11550.

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*.

Simon Jegou, Maximilian Jeblick, and David Austin. 2024. kvpress.

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.

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

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*.

Haoyang Li, Yiming Li, Anxin Tian, Tianhao Tang, Zhanchao Xu, Xuejia Chen, Nicole Hu, Wei Dong, Qing Li, and Lei Chen. 2024. A survey on large language model acceleration based on kv cache management. *arXiv preprint arXiv:2412.19442*.

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. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, Michael Maire, Henry Hoffmann, Ari Holtzman, and Junchen Jiang. 2024b. Cachegen: Kv cache compression and streaming for fast large language model serving. In *Proceedings of the ACM SIGCOMM 2024 Conference*, ACM SIGCOMM '24, page 38–56, New York, NY, USA. Association for Computing Machinery.

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*.

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. 2024a. Bright: A realistic and challenging benchmark for reasoning-intensive retrieval. *CoRR*, abs/2407.12883.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024b. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

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*.

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, 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. 2025. 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.

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*.

Xuan Zhang, Cunxiao Du, Chao Du, Tianyu Pang, Wei Gao, and Min Lin. 2024. Simlayerkv: A simple framework for layer-level kv cache reduction. *arXiv preprint arXiv:2410.13846*.

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 9 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. 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

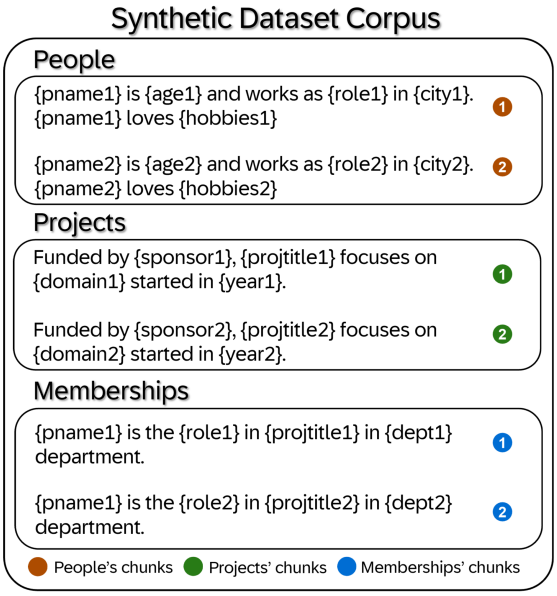

Figure 9: Overview of our synthetic dataset. In this example, the connectivity level is set to 2.

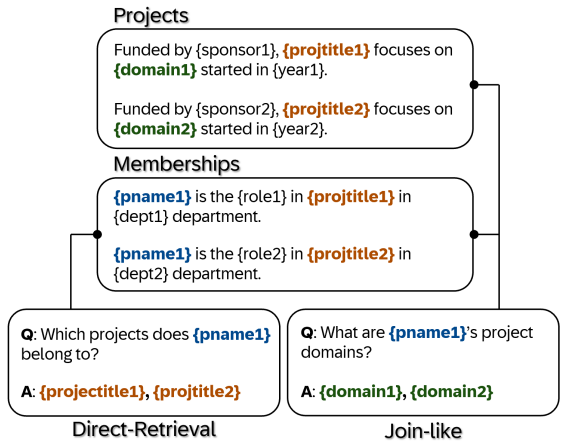

Figure 10: Overview of our questions. In this example, the connectivity level is set to 2.

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 10).

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 10 in order to solve this queries 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 require 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 10, in order to solve this queries 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 Examples of Generated Cheat-sheet

Below are examples of generated cheat-sheets for HotPotQA in Table 1 and QAsper in Table 2. Interestingly, the model often includes plausible questions and answers in its cheat-sheets, mimicking

the way a student might create study notes to actively engage with the material.

Inference Query	What title did the author of Det norske Theater's first production hold ?
Task Description	Answer the question based on the given passages. Only give me the answer and do not output any other words.
Long Document	**Passage 1**: Det Norske Jernkompani Det Norske Jernkompani also known as Jernkompaniet or Det store Jern Compagni, was a Norwegian iron company. It was established in 1618 [...] **Passage 2**: Det norske Theater (Bergen) Det norske Theater is a former theatre in Bergen, Norway, and regarded as the first pure Norwegian stage theatre. It opened in 1850 (1850) by primus motor, violinist Ole Bull, and closed in 1863 (1863), after a bankruptcy. The theatre's first production was Holberg's comedy Den Vægelsindede, and the opening was on 2 January 1850 [...]
Cheat-sheet	Here is the context in concise yet comprehensive form: Det Norske Jernkompani was a Norwegian iron company established in 1618. Det norske Theater was a theatre in Bergen, Norway, and regarded as the first pure Norwegian stage theatre. Språkåret 2013 was a national year of celebration of languages in Norway. Ludvig Holberg was a writer, essayist, philosopher, historian, and playwright born in Bergen, Norway. Lucie Wolf was a Norwegian stage actress. The Feast at Solhaug was the first publicly successful drama by Henrik Ibsen. Per Sunderland was a Norwegian stage actor and film actor. The Nynorsk Literature Prize is awarded annually by Noregs Mållag, Det Norske Teatret, and Det Norske Samlaget. Svein Tindberg is a Norwegian actor. Aloysius Valente was a Norwegian dancer, choreographer and stage instructor. **QA:** What is the name of the Norwegian iron company established in 1618?**Answer:** `Det Norske Jernkompani.`

Table 1: Examples of generated cheat-sheets on HotPotQA. **Inference Query** is only used online, while the **Task Description** offline.

Inference Query	How does this approach compare to other WSD approaches employing word embeddings?
Task Description	You are given a scientific article and a question. Answer the question as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the article, write 'unanswerable'. If the question is a yes/no question, answer 'yes', 'no', or 'unanswerable'. Do not provide any explanation.
Long Document	**Introduction.** Language modelling initially relied on one-hot word encoding, which captured only alphabetical order, not semantic similarity. Vector space models enabled learning lower-dimensional representations and capturing semantic relations, improving tasks such as question answering, stance detection, and claim verification (BIBREF0–2). Recent models (BIBREF3, BIBREF4) exploit the idea that words in similar contexts share semantic similarity. However, these models cannot resolve words with multiple meanings (polysemy). [...]
Cheat-sheet	The article discusses a method for learning multi-sense word embeddings using a KL divergence-based energy function. The method models word embeddings as Gaussian mixtures, where each word is represented by a mixture of Gaussian distributions. The KL divergence between the Gaussian mixtures is used as the energy function to capture the similarity and entailment between words. The method is compared to other approaches, such as w2g and w2gm, and is shown to perform better on benchmark word similarity and entailment datasets. **Cheat Sheet:** **Method:** KL divergence-based energy function for multi-sense word embeddings. **Representation:** Words as mixtures of Gaussians. **Energy Function:** KL divergence between mixtures. **Approximation:** Uses upper/lower bounds for tractability. **Baselines:** Compared to w2g (single Gaussian), w2gm (mixture with expected likelihood kernel). **Performance:** Outperforms baselines on benchmark datasets. **QA:** What is the name of the proposed model in the article? **Answer:** `GM_KL`. What is the name of the dataset used to train the proposed model? **Answer:** `Text8`.

Table 2: Examples of generated cheat-sheets on QAsper. **Inference Query** is only used online, while the **Task Description** offline.

C Task Description Prompts

Here we detail the task description prompt used in the experiments. Each is shown to the language model verbatim. All experiments have been executed with a fixed seed (42) to ensure deterministic results.

Company Notes

> 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 questions, ensuring that the responses are relevant and informative.

Synthetic Dataset

> Answer the question based on the given passages. Only give me the answer and do not output any other words. Answers should be concise and formatted as lists, separated by commas if multiple items are present.

For LongBench we use the task instruction prompts supplied in their dataset.

D Ablation: Task-Focused Cheatsheet Prompt for LonBench–LCC

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 "cheatsheet" summarizing *all* preceding tokens, but it does not explicitly guide the model to prioritize the lines that are most useful for code completion. We therefore replaced that generic summary 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 [3]

Revised prompt. The full text is reproduced below:

```
Since your task is code completion,
    write a cheat sheet that is 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() {
Cheatsheet (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 3 reports automatic scores (higher is better) for three retrieval-token budgets. Injecting the task-focused prompt yields consistent absolute gains of ≈2 points at all budgets, confirming that carefully engineered prompts can help the model compress long contexts into more actionable cues.

Table 3: Impact of task-focused cheatsheet prompt on LCC.

Method	512	1024	2048
LCC (default prompt)	35.49	36.27	32.35
+ Task-focused prompt	**37.51**	**38.48**	**34.25**

E Company Data Annotation

In our human evaluation setup, the annotator follows the same grading protocol as used in the LLM-based evaluation. For each question, the annotator is presented with the question itself, a reference document containing the relevant company notes, and two candidate answers—without knowing which system (RAG or CACHENOTES) produced each answer. The annotator may consult the company notes to assess the factual accuracy and relevance of each response with respect to the question, and is asked to select the better answer. The evaluation was conducted by a single annotator: a PhD student in computer science with a background in NLP, fluent in English, based in Europe, and familiar with both the company's domain and its internal terminology. Instructions given to participant for annotation:

```
You are evaluating answers to technical
    support questions based on internal
    company documentation. For each
    example, you will be shown:

A question
```

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

```
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.
```

The institution of the annotator approved the data collection protocol.

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.