DEFERRED CONTEXTUALIZATION FOR EFFICIENT LONG-CONTEXT KV REUSE

Anonymous authors
Paper under double-blind review

ABSTRACT

Shared-corpus long-context QA repeatedly queries the same document blocks. Per-query re-encoding is expensive and several cache reuse mechanisms have been proposed recently offering varying levels of quality-speed tradeoffs. We analyze this tradeoff through **C4**: four factors that govern whether reusable caches preserve multi-block reasoning quality: **C**ompactness of relevant blocks relative to the instruction, **C**ontiguity among them, **C**ross-attention across them, and **C**leanliness, requiring each relevant block to be encoded without attending to unrelated predecessors. Through controlled experiments we show that each factor matters: accuracy drops when relevant blocks are far from the instruction or from one another; independently cached blocks lose cross-block interaction; and encodings contaminated by irrelevant blocks cannot be fully recovered by decode-time masking alone.

Motivated by the **C4** lens, we propose PRISM, a training-free KV reuse pipeline. PRISM encodes each block independently alongside the instruction, yielding clean reusable KV states, and reassigns relevant blocks to compact, contiguous positions at query time. PRISM established cross-block dependencies during query time through two operations: iterative span extraction, which conditions each block's extraction on the query and on spans extracted from preceding blocks; and in-place KV recomputation, which updates only the extracted span states with full cross-attention. Across long-context QA, claim verification, and synthetic graph traversal tasks, PRISM recovers much of the quality of per-query re-encoding while reducing time-to-first-token by nearly $5\times$, without fine-tuning or full re-encoding.

1 INTRODUCTION

Long-context question answering (QA) over shared corpora is increasingly central to real-world language model deployments. A personal assistant reasoning over months of chat history, an enterprise copilot synthesizing a codebase, or a reader querying a multi-chapter textbook all share a common structure: a fixed collection of text is queried repeatedly by a stream of user requests. The dominant computational bottleneck is the repeated prefilling of the same content, while the primary quality challenge is reliable reasoning over evidence scattered across multiple retrieval units, which we call *blocks*; depending on the application, a block may correspond to a chat session, chapter, document chunk, or other natural segment.

The most direct way to eliminate redundant prefill is **full-context reuse**: encode the entire corpus once, store its key-value (KV) cache, and serve all future queries from this cache. This is efficient, but it fixes each block's KV states according to the context in which that block was encoded, before the query is known. In a full-corpus cache, blocks relevant to a query may be far from the instruction, separated from one another by unrelated blocks, and computed with causal access to preceding blocks that are irrelevant to the query.

Prior work[1] has shown that long-context models can degrade when relevant information is distant, diluted by irrelevant content, or embedded in very long contexts (Hong et al., 2025; Du et al., 2025; Wang et al., 2025; Zhang et al., 2026). At the other extreme, **per-query re-encoding** retrieves a small set of blocks via fast indices such as BM25, and encodes them fresh for each query under full causal attention. This provides query-specific KV states for the retrieved blocks, but requires repeated

[1] A more detailed related work appears in Appendix C

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

prefill computation. Recent cache-reuse methods relax different parts of this tradeoff. Block-level caching methods precompute KV states for smaller units and compose them at inference time using modified attention masks or position layouts (Ma et al., 2025; Lu et al., 2025). These methods reduce repeated prefill, but require supervised fine-tuning so the model can consume the resulting cache layout. Other methods address cache-composition gaps with bridge tokens or selective recomputation (Yang et al., 2025; Yao et al., 2024; Hu et al., 2025; Teng et al., 2026); bridge-token methods require fine-tuning, while recomputation-based methods add online computation. Together, these approaches suggest that cache reuse depends not only on which blocks are retrieved, but also on how their KV states are formed, positioned, and contextualized at query time. To formalize these requirements, we propose the **C4 framework**. Given a query, let the *retrieved blocks* denote the subset of blocks selected as candidate evidence, by a retriever or an oracle in controlled analysis. C4 identifies four factors governing how KV cache reuse affects multi-block QA quality:

- **Compactness**: Are retrieved blocks assigned positions close to the instruction anchor?
- **Contiguity**: Are retrieved blocks assigned adjacent positions in the query-time context?
- **Cross-attention**: Are dependencies among retrieved blocks reflected in the KV states?
- **Cleanliness**: Are retrieved block states formed without attention to irrelevant blocks?

Full-context reuse preserves cross-block attention but forces corpus-level positions and encodes relevant blocks after attending to non-retrieved blocks. Independent block caching improves cleanliness, and position reassignment can yield compact, contiguous layouts, but cached states remain block-local. Per-query re-encoding satisfies all four factors but repeats prefill for every query.

Moving a relevant block farther from the instruction produces a steady accuracy drop (compactness). Increasing the positional gap between relevant blocks degrades performance even with only relevant evidence present (contiguity). Preventing attention across retrieved blocks during prefill causes a large multi-hop accuracy drop (cross-attention). When relevant blocks are encoded after non-retrieved blocks, decode-time masking recovers only part of the loss, suggesting interference is already absorbed into the cached states (cleanliness).

Guided by this diagnosis, we propose **PRISM**, a training-free inference pipeline for reusable KV caches. PRISM encodes each block independently near the instruction anchor and stores pre-RoPE KV states, enabling position reassignment at query time. Given a query, it retrieves relevant blocks and places them contiguously near the instruction, yielding compact and contiguous caches. Yet independently cached blocks lack cross-block context. PRISM addresses this through *deferred contextualization*: it extracts query-relevant spans from retrieved blocks, iteratively conditioning extraction on the query and on spans from earlier blocks using attention heads, then recomputes KV states for **only** selected span tokens under causal attention. This yields a hybrid cache that restores cross-block information at reasoning-critical spans without re-encoding the full retrieved context.

We evaluate PRISM on LongMemEval and CLIPPER, two long-context benchmarks requiring reasoning over evidence distributed across multiple blocks, and on a synthetic graph traversal task that directly probes cross-block dependency recovery. Across models and retrieval settings, PRISM recovers much of the accuracy of per-query re-encoding while substantially reducing time-to-first-token, without fine-tuning or full re-encoding.

Our contributions are: (1) We introduce the **C4 framework**, identifying Compactness, Contiguity, Cross-attention, and Cleanliness as four factors governing KV cache reuse quality. (2) We provide controlled experiments isolating the effect of each C4 factor on long-context reasoning. (3) We propose **PRISM**, a training-free KV reuse pipeline combining independent block caching, query-time position reassignment, iterative span extraction, and targeted in-place KV recomputation. (4) We show that PRISM approaches the quality of per-query re-encoding while substantially reducing time-to-first-token across long-context QA, claim verification, and synthetic graph traversal tasks.

2 Problem Setup and Cache-Reuse Pipelines

We consider a shared-corpus QA setting. A fixed corpus is partitioned into blocks $\mathcal{B} = \{B_1, \ldots, B_N\}$, where each block may be a chat session, document chunk, chapter, or other natural unit. The corpus is queried repeatedly by user queries q_t. For each query, a retriever returns an ordered subset of candidate evidence blocks $\tilde{\mathcal{R}}(q_t) = (B_{i_1}, \ldots, B_{i_m})$, approximating the unknown set of truly relevant blocks. In oracle experiments, $\tilde{\mathcal{R}}(q_t)$ is replaced by ground-truth evidence blocks, allowing us to

study cache reuse independently of retrieval errors. Let I denote the task instruction prefix. The quality target is RE-ENCODE: for every query, the retrieved blocks are encoded from scratch as a contiguous sequence under standard causal attention: $\hat{y}_{\text{RE-ENCODE}} = \arg\max_y p_\theta(y \mid I, \tilde{\mathcal{R}}(q_t), q_t)$. Our goal is to approximate RE-ENCODE while avoiding full prefill over $\tilde{\mathcal{R}}(q_t)$ on every request. A cache-reuse pipeline can be described by three design choices.

Offline cache construction. Blocks are prefilled once and their KV states stored. The cache may be constructed by encoding the full corpus jointly or each block independently. Joint encoding lets later blocks attend to earlier ones but exposes retrieved blocks to non-retrieved blocks; independent encoding avoids this exposure but produces block-local states lacking cross-block dependencies.

Query-time context assembly. Given q_t and $\tilde{\mathcal{R}}(q_t)$, the system assembles cached states into a query-time context, determining position indices and attention patterns. A system may retain original positions or reassign retrieved blocks adjacently near the instruction, and may allow query attention to all cached blocks or restrict it to the retrieved subset.

Answer generation and repair. The model generates the answer from the assembled cache. Some methods use cached states directly; others perform query-time computation—attention steering, bridge-token insertion, or selective KV recomputation—to compensate for missing interactions. Re-encode satisfies all four C4 factors by construction; common cache-reuse strategies satisfy only subsets.

Full-context reuse. The full corpus is encoded once under causal attention and reused for all queries. This preserves cross-block attention, but retrieved blocks may be far from the instruction, separated by unrelated content, and encoded after attending to non-retrieved blocks—violating compactness, contiguity, and cleanliness.

Full-context reuse with query masking. Restricting the query's decode-time attention to retrieved blocks reduces distraction during generation, but does not change the KV states formed during corpus-level encoding. The encoding-time violations remain unresolved.

Independent block caching. Each block is encoded separately and cached offline, improving cleanliness. However, if blocks retain original corpus positions they may remain distant and non-contiguous, and their KV states lack cross-block dependencies.

Independent block caching with position reassignment. If caches permit position reassignment, retrieved blocks can be placed contiguously near the instruction at query time—satisfying compactness, contiguity, and cleanliness. The cached states remain block-local, isolating the remaining gap to cross-block contextualization.

PRISM builds on this last configuration. It uses independent block caches with query-time position reassignment, then performs iterative span extraction followed by in-place KV recomputation to contextualize selected span tokens across retrieved blocks. Section 3 first studies the four C4 factors in isolation, and Section 4 presents the PRISM pipeline.

3 ANALYZING THE IMPACT OF THE C4 FACTORS ON CACHE REUSE

We next study how the four C4 factors affect cache-reuse quality. We use controlled experiments on ground-truth relevant blocks from LongMemEval, removing retrieval quality as a confound. Each experiment varies one property of the cache layout, position assignment, or attention pattern while keeping the relevant evidence fixed. These experiments are designed to separate the main effects of the C4 factors, while noting where factors cannot be perfectly disentangled.

Compactness requires retrieved blocks to be positioned close to the instruction anchor. We consider single-hop queries with exactly one relevant block, encoded at offsets ranging from ≈ 40 to ≈ 110k tokens after the instruction anchor. The query is positioned immediately after the block, so evidence content and query–block adjacency are held fixed; only the block's positional distance from the instruction varies. Figure 2a plots accuracy as a function of encoding position. Accuracy remains high near the instruction anchor and decreases with distance, declining sharply beyond $\sim$30k tokens.

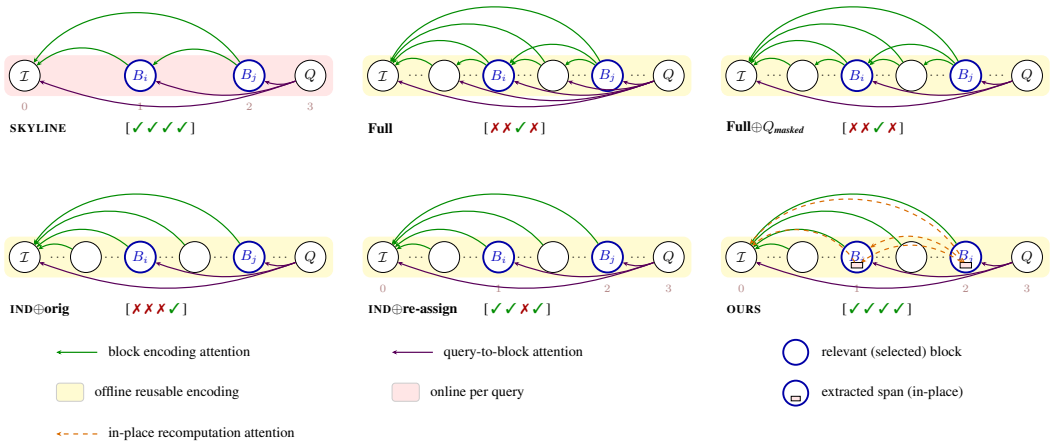


Figure 1: Different cache-reuse pipelines (described in Section 2) and their C4 scores in brackets [**C**ompactness **C**ontiguity **C**ross-attention **C**leanliness]; ✓ = satisfied, ✗ = not satisfied. Blue-bordered circles denote the two relevant blocks B_i, B_j; yellow shading indicates offline reusable encodings, with arrows denoting attention patterns. Pink indices denote reassigned position offsets where applicable. For PRISM, pink squares denote extracted token spans; orange dashed arrows denote the cross-block context introduced by in-place KV recomputation at selected token positions. Existing cache-reuse pipelines satisfy only subsets of the C4 criteria. PRISM starts from clean, compact, and contiguous block caches, then selectively contextualizes extracted spans to address the remaining Cross-attention gap without fine-tuning or full re-encoding.

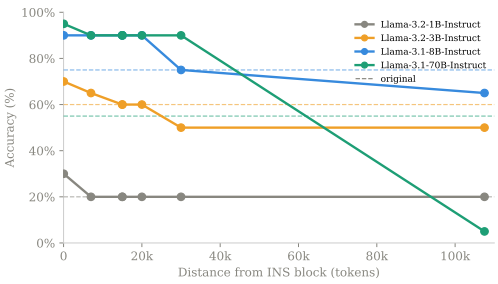
Because the context contains no competing blocks and the query remains adjacent to the evidence, this degradation cannot be explained by retrieval failure or decode-time distraction. The result supports Compactness as a distinct factor: cached states become less reliable when relevant evidence inherits distant positions.

Contiguity requires retrieved blocks to occupy adjacent positions. We use multi-block queries, fixing the first relevant block at ≈ 40 tokens and varying the inter-block gap from 0 to 50k tokens. Attention is restricted to relevant blocks, removing irrelevant-context distraction; the experiment tests whether positional fragmentation hurts reasoning even when correct evidence is present. Figure 2b shows accuracy decreasing as the gap grows. For the 8B model, accuracy stays near 70% up to 10k-token gaps, then drops sharply beyond 30k tokens to 35% at a 50k gap. This 35-point drop shows that multi-block reasoning depends not only on retrieving the right blocks but on arranging them as a coherent positional context. This motivates query-time position reassignment for reusable block caches.

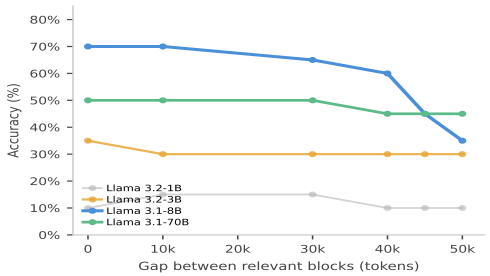
Cross-Attention requires that KV states reflect dependencies among retrieved blocks. We encode all relevant blocks contiguously after the instruction anchor, holding Compactness and Contiguity fixed, and compare three attention variants: **(i) Vanilla**, full causal attention across all retrieved blocks; **(ii) Block**, causal attention within each block only; and **(iii) BlockSink**, within-block attention plus attention to the instruction anchor. Figure 2c shows a pronounced gap between full causal attention and block-isolated variants. For Llama-8B, Vanilla achieves 70% while Block degrades to 20%—a 50-point drop. BlockSink recovers only marginally to 25%, showing that attention to a shared instruction prefix does not substitute for direct inter-block dependency formation. This explains why independently cached blocks can remain inaccurate even when clean, compact, and contiguous: their KV states were computed without access to other retrieved blocks.

Cleanliness requires retrieved block states to be formed without attention to non-retrieved blocks. We place relevant blocks contiguously at three corpus positions: beginning (*First*, ≈ 40 tokens), middle (≈ 55k tokens), and end (≈ 110k tokens), with irrelevant blocks filling the remainder, and encode under full causal attention. We compare two decoding conditions: **(i) full attention** to all corpus blocks, and **(ii) masked decode** attending only to relevant blocks.

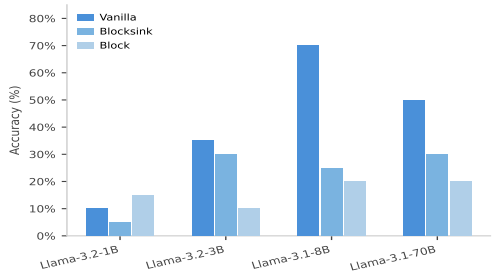
This experiment asks whether decode-time masking is sufficient once relevant block states have been formed inside a larger corpus context. Figure 2d shows two trends. First, under full attention, accuracy is uniformly low ($\approx$30–35%) regardless of position, confirming that irrelevant corpus



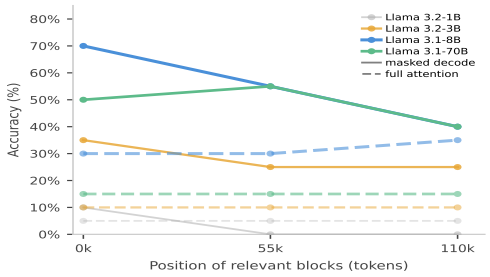
(a) **Compactness**. The relevant block is encoded at increasing token offsets from the instruction anchor. Accuracy is highest when the block is encoded near the instruction and decreases with positional distance.



(b) **Contiguity**. Relevant blocks are encoded with varying inter-block gaps, with the first block positioned immediately after the instruction anchor. Accuracy decreases as the inter-block positional gap increases.



(c) **Cross-attention**. Relevant blocks are encoded contiguously near the instruction. *Vanilla* applies full causal attention across relevant blocks; *Block* restricts each block to local causal attention; *BlockSink* additionally permits attention to the instruction anchor.



(d) **Cleanliness**. Relevant blocks are encoded inside a full corpus context with irrelevant blocks. Solid lines denote masked decoding, where the query attends only to relevant blocks; dashed lines denote full attention over the corpus cache. Masking recovers some accuracy, but weakens after longer irrelevant prefixes.

Figure 2: Impact of the C4 factors on reasoning accuracy. Each experiment changes a controlled property of the cache layout or encoding procedure while keeping the relevant evidence fixed. Note: the Llama-3.1-70B model exhibited anomalous accuracy patterns inconsistent with model scale; similar inversions have been reported in Zhang et al. (2025). We repeated experiments on Llama-3.3-70B and observed consistent trends.

context is harmful during generation. Second, masked decoding substantially improves accuracy at *First* (70%), but provides weaker recovery at *Middle* (55%) and *Last* (40%). Thus, decode-time masking removes only part of the problem: some interference persists once relevant blocks have been encoded in the presence of irrelevant prefixes.

Putting It Together. The four experiments show that cache reuse quality depends on more than retrieving the right evidence. Retrieved blocks should be positioned near the instruction, arranged contiguously, protected from non-retrieved context, and supplied with cross-block information for multi-hop reasoning. Per-query re-encoding satisfies all four properties but repeats prefill for every query. Independent block caching satisfies Cleanliness and can support Compactness and Contiguity through position reassignment, but lacks Cross-attention. This remaining gap motivates PRISM: a training-free pipeline that starts from clean independently cached blocks and restores cross-block information at selected token positions.

4 PRISM: Deferred Contextualization for KV Reuse

The C4 analysis motivates a cache-reuse pipeline with three requirements: cache blocks independently to avoid contamination from non-retrieved blocks, place retrieved blocks compactly and contiguously at query time, and introduce the cross-block dependencies lost by independent caching.

Offline block caching. For each corpus block B_j, PRISM encodes B_j independently as the sole content *but following the instruction prefix* I. Since no other corpus blocks are present during this prefill, the resulting cache is formed without attention to non-retrieved blocks, preserving Cleanliness. To support query-time position reassignment, PRISM stores key states before RoPE rotation.

Query-time context assembly. Given the retrieved sequence $\tilde{\mathcal{R}}(q_t) = (B_{r_1}, \ldots, B_{r_m})$, PRISM assigns the blocks contiguous positions after I. Applying RoPE with these reassigned positions places the blocks near I. This addresses Compactness and Contiguity without re-encoding the retrieved context. The assembled cache remains block-local since the states of B_{r_i} were computed without access to $B_{r_1}, \ldots, B_{r_{i-1}}$. PRISM addresses this remaining Cross-attention gap through span extraction followed by recomputation.

Conditional Attention-based Extraction. PRISM first extracts the parts of each retrieved block that are most likely to participate in cross-block reasoning. Two highlights of PRISM's extraction process are: (1) non-generative attention scoring for efficiency, (2) each extraction conditioned on previous extractions to thread together dependencies across blocks. Each block is divided into chunks. For a chunk $s \subset B_{r_i}$, PRISM computes attention scores using a fixed set of query-focused retrieval heads $\mathcal{Q}$. The context for the query attention vector for block B_i is $c_i = [I,\ q_t,\ S_1, \ldots, S_{i-1}]$, where $S_1, \ldots, S_{i-1}$ are chunks extracted from earlier retrieved blocks.

Let $\alpha^{\text{content}}(s)$ denote the mean attention from the content-bearing tokens in c_i, namely $\{q_t, S_1, \ldots, S_{i-1}\}$, to chunk s, averaged over heads in $\mathcal{Q}$. Let $\alpha^{\text{base}}(s)$ denote the corresponding baseline attention from instruction or prefix tokens. Following prior attention-score calibration methods (Chen et al., 2025; Zhang et al., 2025), we define the residual score as $\alpha^{\text{res}}(s) = \alpha^{\text{content}}(s) - \alpha^{\text{base}}(s)$. Let μ_i and σ_i denote its mean and standard deviation over candidate chunks in B_{r_i}. PRISM selects chunks with $\alpha^{\text{res}}(s) > \mu_i + \lambda\sigma_i$, expands them to sentence or paragraph boundaries, and merges adjacent selections to obtain S_i.

Threading Spans In-Place Across Blocks. Next PRISM recomputes KV states for the selected token positions. Let $\mathcal{T}_i$ denote the token positions covered by S_i in block B_{r_i}, and let $\mathcal{T} = \bigcup_{i=1}^{m} \mathcal{T}_i$.

PRISM recomputes only the key and value states for tokens in $\mathcal{T}$ while conditioning it on all tokens before it like in standard causal attention. The recomputed states are replaced place at their original positions in the assembled cache. The resulting cache is hybrid:

$$\mathcal{H}(q_t) = \text{recomputed KV states on } \mathcal{T} \ \cup \ \text{block-local KV states elsewhere.}$$

Answer generation. The model generates the answer autoregressively from the hybrid cache, $\hat{y}_t = \arg\max_y p_\theta(y \mid I, \mathcal{H}(q_t), q_t)$. No model weights are updated, and no task-specific fine-tuning is required. Algorithm 1 summarizes the full pipeline.

5 Experiments

Datasets and Tasks. We evaluate PRISM on three long-context benchmarks: LongMemEval (Wu et al., 2025), CLIPPER (Pham et al., 2025), and LaRA (Li et al., 2025).

- **LongMemEval** evaluates long-term memory in chat assistants across information extraction, multi-session reasoning, temporal reasoning, knowledge updates, and abstention. Answers may require information across multi-turn, multi-session histories.
- **CLIPPER** evaluates document-level verification of claim pairs in narrative texts, where claims may require evidence from multiple chapters.
- **LaRA** evaluates long-context question answering over diverse document types. We use financial reports and academic papers from the *Reasoning* task in the 128K split, with documents of approximately 110K tokens.

We partition LongMemEval histories by chat session and CLIPPER books by chapter. LongMemEval histories contain 30–40 sessions, totaling approximately 115K tokens. For LongMemEval and CLIPPER, blocks average approximately 3K tokens, and queries typically require 2–6 evidence blocks. For LaRA, we group consecutive paragraphs into blocks of at most 1,024 tokens, yielding approximately 100 blocks per document.

Table 1: Accuracy comparison across retrieval methods on LongMemEval, CLIPPER, and LaRa.

	Model	FULL	Re-encode	BlockSink	CacheBlend	EPIC	PRISM
Oracle							
LME	Llama-3.1-8B	0.41	0.74	0.48	0.67	0.53	**0.70**
	Qwen2.5-7B	0.45	0.65	0.53	0.58	0.54	**0.59**
	Qwen2.5-14B	0.47	0.75	0.57	0.67	0.57	**0.70**
	Qwen3-8B	0.42	0.67	0.51	0.56	0.53	**0.59**
	Qwen3-14B	0.53	0.73	0.53	0.64	0.56	**0.68**
CLIPPER	Llama-3.1-8B	0.27	0.48	0.46	0.47	0.42	**0.50**
	Qwen2.5-7B	0.43	0.56	0.51	0.50	**0.55**	0.51
	Qwen2.5-14B	0.34	0.56	0.46	0.50	0.47	**0.52**
	Qwen3-8B	0.38	0.61	0.50	**0.54**	0.42	**0.54**
	Qwen3-14B	0.50	0.74	0.61	0.62	0.54	**0.65**
Retrieved							
LME	Llama-3.1-8B	0.41	0.64	0.43	0.55	0.49	**0.61**
	Qwen2.5-7B	0.45	0.60	0.47	0.50	0.50	**0.53**
	Qwen2.5-14B	0.47	0.62	0.49	0.55	0.48	**0.60**
	Qwen3-8B	0.42	0.63	0.47	0.50	0.48	**0.52**
	Qwen3-14B	0.53	0.65	0.49	0.51	0.48	**0.60**
CLIPPER	Llama-3.1-8B	0.27	0.43	0.36	0.40	0.35	**0.41**
	Qwen2.5-7B	0.43	0.45	0.40	0.41	**0.44**	0.42
	Qwen2.5-14B	0.34	0.48	0.37	0.40	0.40	**0.43**
	Qwen3-8B	0.38	0.55	0.40	0.41	0.32	**0.44**
	Qwen3-14B	0.50	0.65	0.50	0.54	0.43	**0.57**
LaRa	Llama-3.1-8B	0.85	0.79	0.66	**0.81**	0.70	0.72
	Qwen2.5-7B	0.76	0.87	**0.83**	0.77	**0.83**	0.79
	Qwen2.5-14B	0.85	0.87	0.83	0.81	0.83	**0.89**
	Qwen3-8B	0.79	0.89	0.83	0.81	0.83	**0.87**
	Qwen3-14B	0.91	0.85	0.83	0.83	0.83	**0.85**

Implementation Details. We evaluate PRISM on five LLMs: Llama-3.1-8B-Instruct (L8), Qwen2.5-7B-Instruct (Q7), Qwen2.5-14B-Instruct (Q14), Qwen3-8B-Instruct (Q3-8), and Qwen3-14B-Instruct (Q3-14). For LongMemEval and CLIPPER, we evaluate oracle retrieval and QRRetriever (Zhang et al., 2025). Oracle retrieval uses ground-truth sessions and gold chapters, respectively. For LaRA, we follow the benchmark's retrieval setup (Li et al., 2025), combining dense retrieval and BM25 through reciprocal rank fusion. We select the top-5 blocks and retain their original document order, yielding approximately 4.8K tokens of retrieved context per question. Within each retrieval setting, all retrieval-based methods use the same selected blocks, while FULL uses the complete context. For span extraction, we use QRHead (Zhang et al., 2025) to identify query-focused retrieval heads. Following QRHead, we select one head set per model on LongMemEval and reuse it across benchmarks. Unless otherwise stated, we use candidate chunks of 10 tokens and an extraction threshold of $\lambda = 1.0$. All experiments run on a single NVIDIA Blackwell B200 MIG instance with 90 GB VRAM. We use Gemini-3.1-Pro as the LLM judge for LongMemEval and LaRA.

Baselines. We compare against several existing cache-reuse baselines: (1) **FULL**: reuses the full-context cache described in Section 2. (2) **RE-ENCODE**: No cache reuse as described in Section 2. (3) **BlockSink**: Our offline block encoding without re-contextualization. (4) **CACHEBLEND** (Yao et al., 2024): selects tokens for recomputation using early-layer KV deviations, independently of the query. (5) **EPIC** (Hu et al., 2025): recomputes the first `legoK` $= 32$ tokens of each cached block in the query-time context.

Main Results. Table 1 reports accuracy across retrievers. On LongMemEval, PRISM nearly matches RE-ENCODE. With L8, it obtains 0.70 vs. 0.74 under oracle retrieval and 0.61 vs. 0.64

with QRRetriever. With Qwen2.5-14B-Instruct, the corresponding scores are 0.70 vs. 0.75 and 0.60 vs. 0.62. The gains over block-local reuse are larger: for Llama-3.1-8B-Instruct with QRRetriever, BlockSink and EPIC obtain 0.43 and 0.49, while PRISM reaches 0.61. This gap establishes the importance of cross-attention.

Figure 3 reports latency under oracle retrieval. PRISM reduces TTFT relative to Re-encode from 1.60 s to 0.36 s for Llama-3.1-8B-Instruct and from 3.00 s to 0.63 s for Qwen2.5-14B-Instruct. It also has lower TTFT than CacheBlend for every model, while remaining slower than EPIC due to its online extraction and selective recomputation. Thus, PRISM trades a small amount of query-time computation for most of the accuracy recovered by full re-encoding.

Figure 3: Latency under oracle retrieval for LongMemEval. We report time-to-first-token / total time (TTFT/TT), both in seconds. Full latency results are in Tables 3–4 in the Appendix.

LLM	FULL	Re-encode	EPIC	CacheBlend	PRISM
L8	0.17 / 12.37	1.60 / 6.29	0.12 / 3.36	0.50 / 4.86	0.36 / 4.02
Q7	0.09 / 35.11	1.29 / 4.71	0.08 / 5.11	0.41 / 4.37	0.29 / 4.46
Q14	0.35 / 39.26	3.00 / 10.92	0.23 / 9.61	0.83 / 11.62	0.63 / 9.85
L3	0.11 / 18.44	1.04 / 3.39	0.10 / 2.57	0.37 / 4.45	0.30 / 2.35
Q3-8	0.27 / 58.84	1.85 / 7.77	0.15 / 5.92	0.57 / 6.30	0.37 / 6.07
Q3-14	0.32 / 57.40	2.63 / 10.53	0.20 / 8.45	0.75 / 10.09	0.55 / 9.27

FULL gives a different failure mode. It has low TTFT because the corpus cache is precomputed, but the cache violates the C4 conditions: relevant blocks may be distant, non-contiguous, and encoded after non-retrieved context. On LongMemEval with Llama-3.1-8B-Instruct, FULL obtains 0.41, compared to 0.70 for PRISM under oracle retrieval. Its total time is also high because decoding attends over the full corpus cache, whereas PRISM decodes from a compact retrieved cache.

On CLIPPER, PRISM matches or exceeds the strongest cache-reuse baseline for four of five models under both retrieval settings. The gains are consistent across most models, although Re-encoding remains more accurate in this setting. Under oracle retrieval, PRISM also exceeds Re-encode with Llama-3.1-8B-Instruct (0.50 versus 0.48). On LaRA, PRISM is the strongest cache-reuse method for three of five models. It achieves the highest overall accuracy with Qwen2.5-14B-Instruct (0.89 versus 0.87 for Re-encode) and approaches Re-encode with Qwen3-8B-Instruct (0.87 versus 0.89).

Ablations. Table 2 ablates the two decisions in PRISM: how evidence spans are selected, and how they are threaded across retrieved blocks. Extraction variants: OIE uses PRISM's conditional attention-based extraction but answers from extracted spans alone; OE extracts independently per block; OGE uses model-generated evidence; ALLH uses all attention heads instead of QRHead-selected retrieval heads. Threading variants: BSIE appends extracted spans to the query without modifying the block cache; STEER (Ye et al., 2026; Zhang et al., 2024) boosts decode-time attention toward selected positions without rewriting KV states; EPIC recomputes a fixed prefix of each block; SNIPPET answers from BM25-selected sentences without the block cache.

Table 2: Ablations on LongMemEval with Llama-3.1-8B-Instruct under oracle retrieval.

Method	Type	Acc.
OIE	conditional attention-based extraction	0.64
OE	parallel; not conditional	0.62
OGE	generative; not attention-based	0.61
ALLH	ALL heads; not retrieval heads	0.12
PRISM	Threading Spans in-place across blocks	0.70
BSIE	Threading Spans outside blocks	0.60
Steer	Boosting; No Threading	0.59
EPIC	Threading blocks w/ prefix (32 tokens)	0.53
EPIC	Above with 215 tokens	0.61
SNIPPET	BM25-selected sentences, No KV	0.50

Conditional extraction. Conditioning on earlier extractions improves over parallel extraction (OIE's 0.64 vs. OE's 0.62). Generated evidence is close (OGE's 0.61) but it is autoregressive, may drift from the source, and lacks fixed positions to rewrite.

Retrieval heads. Replacing QRHead-selected heads with all heads drops the extraction-only variant to 0.12 and PRISM+ALLH to 0.57 (vs. 0.70); averaging over all heads dilutes the retrieval signal.

In-place threading. BSIE keeps extracted spans outside the blocks as extra text reducing accuracy to 0.6. STEER boosts attention towards spans and gives 0.59. Neither is sufficient: PRISM recomputes spans at their original positions so the cache itself carries cross-block context. The gain shows that stale block-local states must be rewritten, not merely re-weighted or repeated.

Static versus selected recomputation. EPIC recomputes fixed prefixes: 32 tokens yields 0.53; 215 tokens (matching PRISM's budget) yields 0.61, still well below 0.70. SNIPPET reaches 0.50, confirming that selected text alone does not substitute for contextualized reusable states. Additional analyses of BM25 retrieval, retrieval depth, extraction settings, and cross-corpus head transfer appear in Appendix Tables 6–9.

Mechanistic Evaluation of Deferred Contextualization.

We use a synthetic chain traversal task to test whether PRISM recovers cross-block dependencies. Each block contains directed edge records of the form `source:destination`. Across m blocks, one edge per block forms a chain $k_0 \to v_1 \to \cdots \to v_m$. Given k_0, the model must return the terminal node v_m, the unique node with no outgoing edge. We evaluate 100 instances using exact-match accuracy. Appendix B.1 describes distractor construction and the prompt format.

Figure 4 reports exact-match accuracy on 2-hop and 3-hop chain traversal tasks across increasing block sizes. We compare RE-ENCODE, PRISM, and variants BlockSink, OIE, BSIE, EPIC (defined in Table 2) to separate the effects of block-local cache reuse, iterative span extraction, query enhancement, and in-place KV recomputation.

PRISM performs best among cache-reuse methods, especially as hop count and block size increase (note RE-ENCODE does not reuse cache). The comparison separates two effects: conditional extraction helps identify the relevant edge in each block, while in-place recomputation writes the selected evidence back into the cache with access to preceding retrieved blocks. This supports the central claim that deferred contextualization requires both targeted span selection and rethreading them to establish dependency among relevant spans.

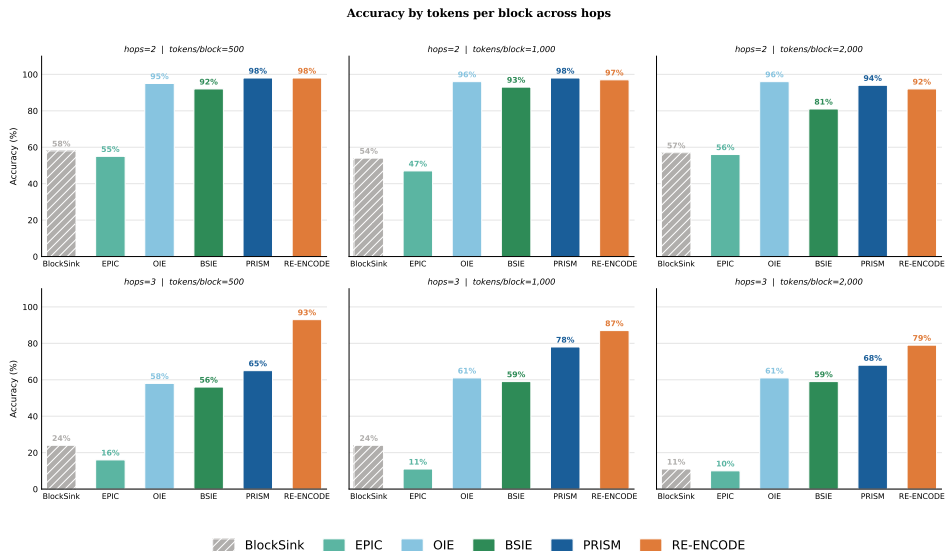


Figure 4: Effectiveness of deferred contextualization methods on synthetic graph traversal datasets

Conclusion. We introduced C4 to characterize the factors governing reasoning quality under KV cache reuse. Guided by this analysis, PRISM combines independent block caching and position reassignment with conditional span extraction and in-place KV recomputation. Experiments on three benchmarks and synthetic graph traversal demonstrate the benefits of selective contextualization, with up to $5\times$ lower TTFT than Re-encode on LongMemEval.

Limitations. PRISM's extraction step performs one forward pass per retrieved block, which means latency scales linearly with retrieval set size. For settings with many retrieved blocks, this overhead may become non-trivial, though in our experiments with top-5 retrieval it remains well within the TTFT budget. Our method should ideally be evaluated on more benchmarks and even larger LLMs.

USE OF GENERATIVE AI

We have not used generative AI tools for generating synthetic data, formulating mathematical claims or proofs, designing methodology or experiments, implementing methods, or interpreting results.

We used generative AI tools to assist with English-language grammar correction, paraphrasing, and text compression for improving clarity and conciseness in the writing. All AI-assisted text was reviewed and verified by the authors for accuracy and fidelity to the original intended meaning.

REFERENCES

Shijie Chen, Bernal Jimenez Gutierrez, and Yu Su. Attention in large language models yields efficient zero-shot re-rankers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=yzloNYH3QN.

Yufeng Du, Minyang Tian, Srikanth Ronanki, Subendhu Rongali, Sravan Babu Bodapati, Aram Galstyan, Azton Wells, Roy Schwartz, Eliu A Huerta, and Hao Peng. Context length alone hurts LLM performance despite perfect retrieval. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 23281–23298, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.1264. URL https://aclanthology.org/2025.findings-emnlp.1264/.

K. Hong, A. Troynikov, and J. Huber. Context rot: How context degradation affects llm performance. https://research.trychroma.com/context-rot, 2025. Accessed: 2026-03-01.

Junhao Hu, Wenrui Huang, Weidong Wang, Haoyi Wang, tiancheng hu, zhang qin, Hao Feng, Xusheng Chen, Yizhou Shan, and Tao Xie. EPIC: Efficient position-independent caching for serving large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=qjd3ZUiHRT.

Kuan Li, Liwen Zhang, Yong Jiang, Pengjun Xie, Fei Huang, Shuai Wang, and Minhao Cheng. LaRA: Benchmarking retrieval-augmented generation and long-context LLMs – no silver bullet for LC or RAG routing. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=CLF25dahgA.

YUCHENG LI, Huiqiang Jiang, Qianhui Wu, Xufang Luo, Surin Ahn, Chengruidong Zhang, Amir H. Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, and Lili Qiu. SCBench: A KV cache-centric analysis of long-context methods. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=gkUyYcY1W9.

Songshuo Lu, Hua Wang, Yutian Rong, Zhi Chen, and Yaohua Tang. TurboRAG: Accelerating retrieval-augmented generation with precomputed KV caches for chunked text. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6588–6601, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.334. URL https://aclanthology.org/2025.emnlp-main.334/.

Dongyang Ma, Yan Wang, and Tian Lan. Block-attention for efficient prefilling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=7zNYY1E2fq.

Han Peng, Jinhao Jiang, Zican Dong, Wayne Xin Zhao, and Lei Fang. CAFE: Retrieval head-based coarse-to-fine information seeking to enhance multi-document QA capability. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12966–12978, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.655. URL https://aclanthology.org/2025.emnlp-main.655/.

Chau Minh Pham, Yapei Chang, and Mohit Iyyer. CLIPPER: Compression enables long-context synthetic data generation. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=akHq1QcqeZ.

Xin Teng, Canyu Zhang, Shaoyi Zheng, Danyang Zhuo, Tianyi Zhou, and Shengjie Wang. Infoflow kv: Information-flow-aware kv recomputation for long context, 2026. URL https://arxiv.org/abs/2603.05353.

Meiyun Wang, Takeshi Kojima, Yusuke Iwasawa, and Yutaka Matsuo. Lost in the distance: Large language models struggle to capture long-distance relational knowledge. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 4536–4544, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.256. URL https://aclanthology.org/2025.findings-naacl.256/.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=pZiyCaVuti.

Jingbo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. KVLink: Accelerating large language models via efficient KV cache reuse. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=oDcAGSXZZP.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. Cacheblend: Fast large language model serving for rag with cached knowledge fusion, 2024. URL https://arxiv.org/abs/2405.16444.

Xi Ye, Wuwei Zhang, Fangcong Yin, Howard Yen, and Danqi Chen. Dysco: Dynamic attention-scaling decoding for long-context lms, 2026. URL https://arxiv.org/abs/2602.22175.

Alex L. Zhang, Tim Kraska, and Omar Khattab. Recursive language models, 2026. URL https://arxiv.org/abs/2512.24601.

Qingru Zhang, Chandan Singh, Liyuan Liu, Xiaodong Liu, Bin Yu, Jianfeng Gao, and Tuo Zhao. Tell your model where to attend: Post-hoc attention steering for llms, 2024. URL https://arxiv.org/abs/2311.02262.

Wuwei Zhang, Fangcong Yin, Howard Yen, Danqi Chen, and Xi Ye. Query-focused retrieval heads improve long-context reasoning and re-ranking. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 23791–23805, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1214. URL https://aclanthology.org/2025.emnlp-main.1214/.

APPENDIX

A PRISM PIPELINE

B ABLATIONS.

B.1 SYNTHETIC TASK CONSTRUCTION

Each instance contains a chain distributed across m blocks, with block i containing the edge (v_{i-1}, v_i), where $v_0 = k_0$. All other records are distractors sampled from a pool of random 4-character hexadecimal strings. We use two distractor types. *Fresh* distractors introduce new source-destination pairs. *Override* distractors simulate sequential updates by assigning a new destination to a previously seen distractor source; in each block, approximately 25% of prior distractor sources are overridden. Figure 5 illustrates the full prompt structure.

Algorithm 1 PRISM Inference Pipeline

Input: block corpus $\mathcal{B} = \{B_1, \ldots, B_N\}$, instruction I, query q_t, threshold λ, QR heads $\mathcal{Q}$
Output: answer $\hat{y}_t$
Offline cache construction
for each block $B_j \in \mathcal{B}$ **do**
  Encode B_j independently as the sole content after I
  Store pre-RoPE keys $\mathbf{K}_j^0$ and values $\mathbf{V}_j$
end for
Query-time context assembly
Retrieve blocks for q_t
Sort retrieved blocks by original corpus order, yielding $\tilde{\mathcal{R}}(q_t) = (B_{r_1}, \ldots, B_{r_m})$ with $r_1 < \cdots < r_m$
Assign contiguous positions to retrieved blocks immediately after I
Apply RoPE to $\mathbf{K}_{r_1}^0, \ldots, \mathbf{K}_{r_m}^0$ using reassigned positions
Form assembled block-local cache $\mathcal{C}(q_t)$
Iterative span extraction
$\mathcal{S} \leftarrow \emptyset$
for $i = 1$ to m **do**
  Construct extraction context $c_i = [I, q_t, S_1, \ldots, S_{i-1}]$
  Score chunks in B_{r_i} by letting c_i attend to the cached states of B_{r_i}
  for each candidate chunk $s \subset B_{r_i}$ **do**
    Compute residual attention score

$$\alpha^{\text{res}}(s) = \alpha^{\text{content}}(s) - \alpha^{\text{prefix}}(s)$$

  end for
  Compute μ_i, σ_i over chunk scores in B_{r_i}
  Select chunks with $\alpha^{\text{res}}(s) > \mu_i + \lambda\sigma_i$
  Expand selected chunks to sentence or paragraph boundaries and merge adjacent spans
  Let S_i be the resulting span set; $\mathcal{S} \leftarrow \mathcal{S} \cup \{S_i\}$
end for
In-place KV recomputation
Let $\mathcal{T}_i$ be the token positions covered by S_i in B_{r_i}, and let $\mathcal{T} = \bigcup_i \mathcal{T}_i$
for $i = 1$ to m **do**
  Recompute KV states only for positions in $\mathcal{T}_i$
  Each selected token attends to I, earlier retrieved blocks $B_{r_1}, \ldots, B_{r_{i-1}}$, and earlier tokens in B_{r_i}
  Write recomputed states back into $\mathcal{C}(q_t)$ at positions $\mathcal{T}_i$
end for
Form hybrid cache $\mathcal{H}(q_t)$ with recomputed states on $\mathcal{T}$ and cached block-local states elsewhere
Answer generation
Generate $\hat{y}_t = \arg\max_y p_\theta(y \mid I, \mathcal{H}(q_t), q_t)$
return $\hat{y}_t$

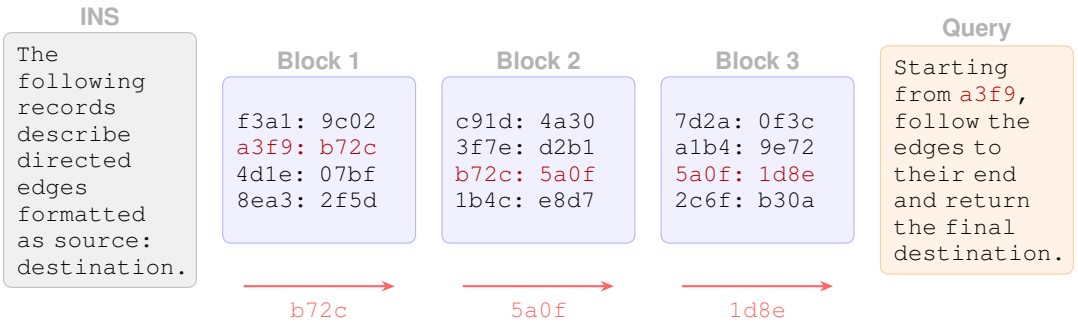


Figure 5: Prompt structure for the synthetic chain traversal task (N=3 blocks, H=3 hops). Each data block contributes one chain edge (**red**); remaining records are distractors. The model must traverse the full path from `a3f9` to the terminal destination `1d8e`.

Table 3: Latency under Oracle and QRRetriever (top-5) retrieval (LongMemEval). We report time-to-first-token / total time (TTFT/TT), both in seconds.

LLM	FULL	Re-encode	BlockSink	EPIC	CacheBlend	PRISM
Oracle						
L8	0.17 / 12.37	1.60 / 6.29	0.03 / 2.05	0.12 / 3.36	0.50 / 4.86	0.36 / 4.02
Q7	0.09 / 35.11	1.29 / 4.71	0.03 / 4.34	0.08 / 5.11	0.41 / 4.37	0.29 / 4.46
Q14	0.35 / 39.26	3.00 / 10.92	0.07 / 8.47	0.23 / 9.61	0.83 / 11.62	0.63 / 9.85
L3	0.11 / 18.44	1.04 / 3.39	0.02 / 1.70	0.10 / 2.57	0.37 / 4.45	0.30 / 2.35
Q3-8	0.27 / 58.84	1.85 / 7.77	0.05 / 4.85	0.15 / 5.92	0.57 / 6.30	0.37 / 6.07
Q3-14	0.32 / 57.40	2.63 / 10.53	0.06 / 10.53	0.20 / 8.45	0.75 / 10.09	0.55 / 9.27
QRRetriever (top-5)						
L8	0.17 / 12.37	1.87 / 5.78	0.03 / 1.78	0.14 / 3.22	0.58 / 5.61	0.39 / 4.23
Q7	0.09 / 35.11	1.51 / 5.26	0.03 / 4.75	0.09 / 4.65	0.47 / 4.77	0.32 / 4.75
Q14	0.35 / 39.26	3.51 / 11.32	0.07 / 8.20	0.25 / 9.41	0.96 / 11.46	0.70 / 11.04
L3	0.11 / 18.44	1.21 / 3.45	0.03 / 1.79	0.11 / 2.61	0.42 / 3.16	0.33 / 2.53
Q3-8	0.27 / 58.84	2.17 / 8.98	0.05 / 4.72	0.17 / 5.76	0.66 / 6.55	0.41 / 6.77
Q3-14	0.32 / 57.40	3.08 / 12.50	0.06 / 12.37	0.21 / 8.78	0.86 / 10.73	0.60 / 10.40

Table 4: Latency under Oracle and QRRetriever (top-5) retrieval (CLIPPER). We report time-to-first-token / total time (TTFT/TT), both in seconds.

LLM	FULL	Re-encode	BlockSink	EPIC	CacheBlend	PRISM
Oracle Retrieval						
L8	0.13 / 44.66	0.24 / 6.17	0.03 / 6.83	0.06 / 6.07	0.12 / 6.76	0.16 / 6.83
Q7	0.08 / 59.51	0.21 / 4.61	0.03 / 4.49	0.05 / 4.43	0.11 / 4.92	0.16 / 4.98
Q14	0.28 / 50.77	0.43 / 7.07	0.04 / 6.54	0.10 / 6.32	0.19 / 8.36	0.27 / 6.76
Q3-8	0.23 / 92.83	0.28 / 5.40	0.03 / 5.55	0.07 / 5.47	0.14 / 6.24	0.20 / 6.19
Q3-14	0.26 / 53.84	0.40 / 7.01	0.04 / 7.43	0.09 / 7.45	0.17 / 7.94	0.25 / 7.89
QRRetriever (top-5)						
L8	0.13 / 44.66	0.90 / 10.81	0.03 / 10.25	0.09 / 11.11	0.31 / 14.38	0.51 / 12.47
Q7	0.08 / 59.51	0.73 / 7.52	0.03 / 8.23	0.07 / 8.45	0.26 / 8.99	0.52 / 9.13
Q14	0.28 / 50.77	1.66 / 10.03	0.07 / 11.37	0.17 / 12.07	0.52 / 11.94	1.06 / 12.29
Q3-8	0.23 / 92.83	1.04 / 9.56	0.04 / 11.04	0.11 / 9.76	0.35 / 12.14	0.64 / 13.41
Q3-14	0.26 / 53.84	1.47 / 11.85	0.06 / 11.53	0.15 / 11.48	0.46 / 13.09	0.99 / 13.72

Table 5: Ablation results on LongMemEval. **Snippet** selects the top-5 sentences using BM25 from the top sessions per retriever. **Steer** applies decode-time attention steering. **OIE** uses iterative extraction without BlockSink. **BSIE** combines BlockSink with iterative extraction.

Model	Retriever	Snippet	Steer	OIE	BSIE	PRISM
L8	ORACLE	0.50	0.59	0.64	0.60	0.70
	BM25	0.37	0.48	0.50	0.48	0.51
	QRRETRIEVER	0.46	0.54	0.56	0.54	0.61
L3	ORACLE	0.42	0.40	0.54	0.53	0.58
	BM25	0.33	0.39	0.43	0.45	0.45
	QRRETRIEVER	0.43	0.46	0.48	0.48	0.51
Q14	ORACLE	0.51	0.63	0.73	0.70	0.70
	BM25	0.39	0.47	0.54	0.54	0.50
	QRRETRIEVER	0.47	0.55	0.69	0.61	0.60
Q7	ORACLE	0.47	0.47	0.48	0.59	0.59
	BM25	0.34	0.39	0.35	0.45	0.42
	QRRETRIEVER	0.42	0.44	0.43	0.53	0.53

Table 6: Abalation results on LongMemEval with BM25.

Model	NaiveReuse	Re-encode	BlockSink	EPIC	CacheBlend	PRISM
Llama-3.2-3B	0.26	0.38	0.29	0.33	0.38	0.45
Llama-3.1-8B	0.41	0.52	0.38	0.41	0.48	0.51
Qwen2.5-7B	0.45	0.47	0.43	0.39	0.45	0.42
Qwen2.5-14B	0.47	0.56	0.45	0.43	0.47	0.50
Qwen3-8B	0.42	0.50	0.40	0.41	0.43	0.47
Qwen3-14B	0.53	0.54	0.41	0.43	0.44	0.50

Table 7: Accuracy-latency trade-off as the number of retrieved blocks increases (LongMemEval, L8, BM25). PRISM's latency advantage over Re-encode grows with k: Re-encode TTFT increases from 1.14 s at $k = 5$ to 13.29 s at $k = 20$, while PRISM increases from 0.33 s to 1.53 s, widening the TTFT gap from 3.5× to 8.7×. Accuracy decreases for all methods as more BM25-retrieved blocks are added, including Re-encode, indicating that the drop is driven by additional distractor blocks rather than PRISM specifically.

k	Method	Acc	TTFT (s)
5	BlockSink	0.38	0.03
	PRISM	0.51	0.33
	Re-encode	0.52	1.14
10	BlockSink	0.29	0.04
	PRISM	0.41	0.69
	Re-encode	0.49	3.64
20	BlockSink	0.17	0.08
	PRISM	0.34	1.53
	Re-encode	0.43	13.29

Table 8: Sensitivity of PRISM to the threshold λ and chunk size. Across a 4× range of λ (TH = 0.5–2.0), accuracy changes by at most ±0.02 relative to the paper setting (TH = 1.0), with no statistically significant differences under McNemar's test. Increasing the chunk size from C10 to C30 also leaves accuracy unchanged. We use TH = 1.0 throughout, without benchmark- or model-specific tuning.

Config	Acc	Δ	McNemar p
C10 TH0.5	0.71	+0.02	0.17
C10 TH1.0 (paper)	0.70	–	–
C10 TH1.5	0.68	−0.02	0.21
C10 TH2.0	0.71	+0.01	0.57
C30 TH1.0	0.70	+0.00	1.00

Table 9: Cross-corpus transfer of calibrated QRHeads: heads calibrated on Natural Questions (NQ) vs. LongMemEval (LME), evaluated on LongMemEval (L8). NQ-derived heads perform at least as well as LME-derived heads across all three retrievers, indicating that the calibrated head set transfers across corpora rather than depending on the evaluation benchmark.

Retriever	LME-derived heads	NQ-derived heads
Oracle	0.70	0.72
QRRetriever	0.61	0.62
BM25	0.51	0.51

Question: ⟨*question*⟩
Relevant details from Block 1: ⟨S_1⟩
Relevant details from Block 2: ⟨S_2⟩
⋮
Relevant details from Block i-1: ⟨S_{i-1}⟩
Your task is to extract relevant facts verbatim from the Block that are needed to answer the question or continue the reasoning chain.

Figure 6: Extraction prompt template used in PRISM's iterative snippet extraction. The accumulated snippets $S_1, \ldots, S_{i-1}$ from previous blocks provide iterative context for extracting from the i-th block. When $i = 1$, the extraction context is empty.

C Related Work

Long-context QA over shared corpora connects two lines of work: understanding why long contexts degrade model quality, and designing inference systems that reuse KV caches without giving up the benefits of query-specific context construction.

Long-Context Degradation. Prior work has shown that long-context models can degrade even when the answer evidence is present in the context. Hong et al. (2025) study performance decay as context grows and irrelevant content competes with relevant evidence. Du et al. (2025) show that degradation can persist even under strong retrieval settings and decode-time masking, suggesting that long-context failures are not only a retrieval problem. Wang et al. (2025) identify a complementary distance effect: models struggle to connect relational evidence when relevant pieces are far apart and separated by noisy spans. These findings motivate the Compactness, Contiguity, and Cleanliness factors in our C4 framework.

Full-Context Reuse and Retrieval-Augmented Generation. Full-context reuse encodes the corpus once and amortizes prefill across queries, but the resulting cache fixes each block's KV states according to its corpus-level position and encoding context. Our C4 analysis makes this failure mode explicit: full-context reuse can preserve cross-block attention, but it violates Compactness, Contiguity, and Cleanliness when relevant blocks are distant, fragmented, or encoded after unrelated blocks. Standard retrieval-augmented generation takes the opposite approach: it retrieves a query-specific subset and re-encodes it under full causal attention. This gives high-quality KV states for the retrieved blocks, but repeats prefill computation for every query. Decode-time approaches such as CAFE (Peng et al., 2025) reduce distraction by steering attention toward retrieved content, but they operate after offline KV states have already been formed. This is complementary to PRISM, which targets the cache construction and query-time contextualization steps.

Block-Level KV Caching and Cache Composition. Block-Attention (Ma et al., 2025) and TurboRAG (Lu et al., 2025) precompute KV caches for smaller units and compose them at inference time, reducing repeated prefill but changing the attention structure used by the model. These methods improve efficiency, but recovering quality can require fine-tuning or specialized attention layouts. KVLink (Yang et al., 2025) introduces trainable bridge tokens to reconnect independently cached blocks, while CacheBlend (Yao et al., 2024), EPIC (Hu et al., 2025), and InfoFlow-KV (Teng et al., 2026) use selective recomputation to refresh parts of a reused cache. PRISM follows the same broad goal of repairing independently reusable caches, but differs in how it chooses what to repair: it uses iterative, query-conditioned span extraction to identify the token positions whose KV states are recomputed. Thus, PRISM requires no fine-tuning and avoids full re-encoding, while still introducing targeted cross-block contextualization at query time.

Retrieval Heads and Attention-Guided Extraction. Zhang et al. (2025) identify query-focused retrieval heads whose attention scores help locate query-relevant tokens in long contexts. PRISM uses these heads as a non-generative extraction signal over independently cached blocks. Unlike chain-of-thought prompting or generative summarization, this extraction step selects spans from the source blocks rather than producing new intermediate text. The extracted spans then determine which token positions are recomputed under full causal attention, linking attention-guided extraction to targeted KV repair.