Some Papers on LLM

Date: March 4, 2026

Transformer

- Input Tokens
- ↓
- Token Embedding
- ↓
- + Positional Encoding
- ↓
- Hidden Representation
- ↓
- Transformer Block × N
- ├── Multi-Head Self Attention
- | ↓
- | Add & LayerNorm
- |
- ├── Feed Forward (MLP)
- | ↓
- | Add & LayerNorm
- ↓
- Final Hidden States
- ↓
- Linear Projection (LM Head)
- ↓
- Logits
- ↓
- Softmax
- ↓
- Next Token Probability

Attention

$$X \in \mathbb{R}^{T \times d_{\text{model}}}$$

$$W_Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, \quad W_K \in \mathbb{R}^{d_{\text{model}} \times d_k}, \quad W_V \in \mathbb{R}^{d_{\text{model}} \times d_v}$$

$$Q = XW_Q \in \mathbb{R}^{T \times d_k}$$

$$K = XW_K \in \mathbb{R}^{T \times d_k}$$

$$V = XW_V \in \mathbb{R}^{T \times d_v}$$

$$S = \frac{QK^\top}{\sqrt{d_k}} \in \mathbb{R}^{T \times T} \quad S_{t,i} = \frac{q_t \cdot k_i}{\sqrt{d_k}} \quad O(T^2 d_k)$$

$$\tilde{S} = S + M$$

$$A = \text{softmax}(\tilde{S}) \in \mathbb{R}^{T \times T}$$

$$O = AV \in \mathbb{R}^{T \times d_v}$$

KV cache

- **Autoregressive generation**
- At step t, the model generates the next token using previous tokens.
- Without KV Cache
- Per-step computation: O(T^2)
- With KV Cache
- Per-step computation: O(T)

Efficient Streaming Language Models with Attention Sinks

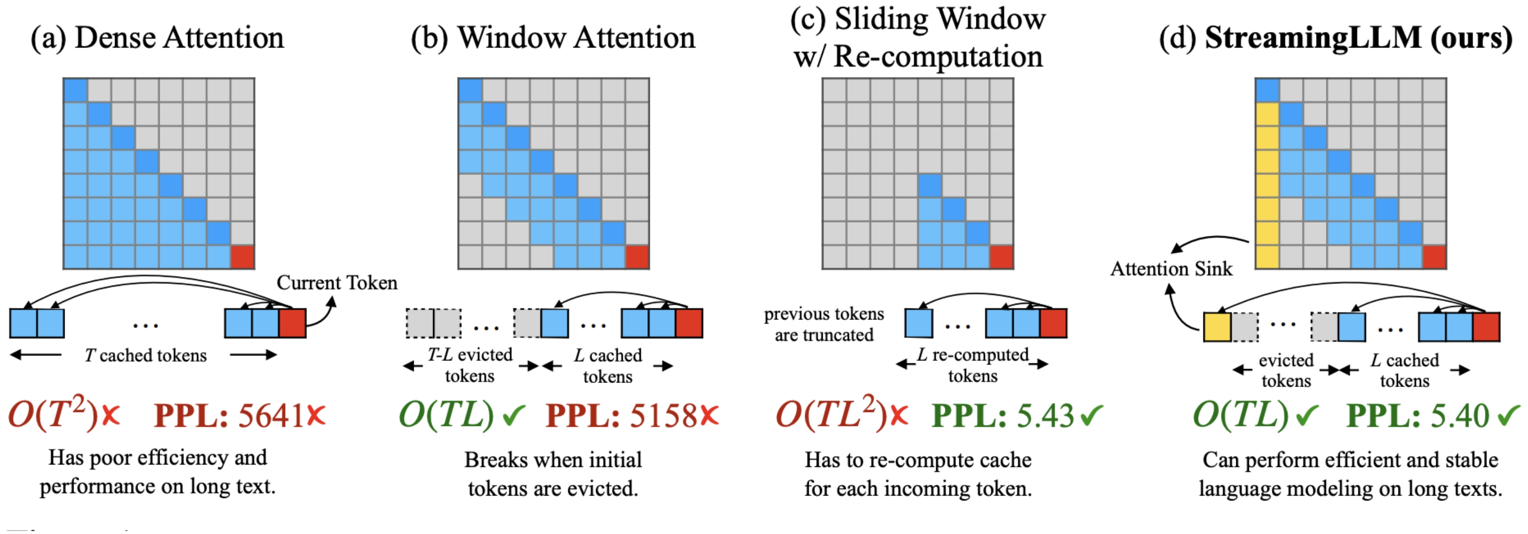
Guangxuan Xiao[1*] **Yuandong Tian**[2] **Beidi Chen**[3] **Song Han**[1,4] **Mike Lewis**[2]

[1] Massachusetts Institute of Technology [2] Meta AI
[3] Carnegie Mellon University [4] NVIDIA

`https://github.com/mit-han-lab/streaming-llm`

Motivation and Background (StreamingLLM)

- Streaming chat/agent workloads require continuous decoding; full attention makes KV cache grow with sequence length
- Window attention saves memory but perplexity degrades when early tokens are evicted
- Common belief says recent tokens are enough; this paper shows that assumption is incomplete



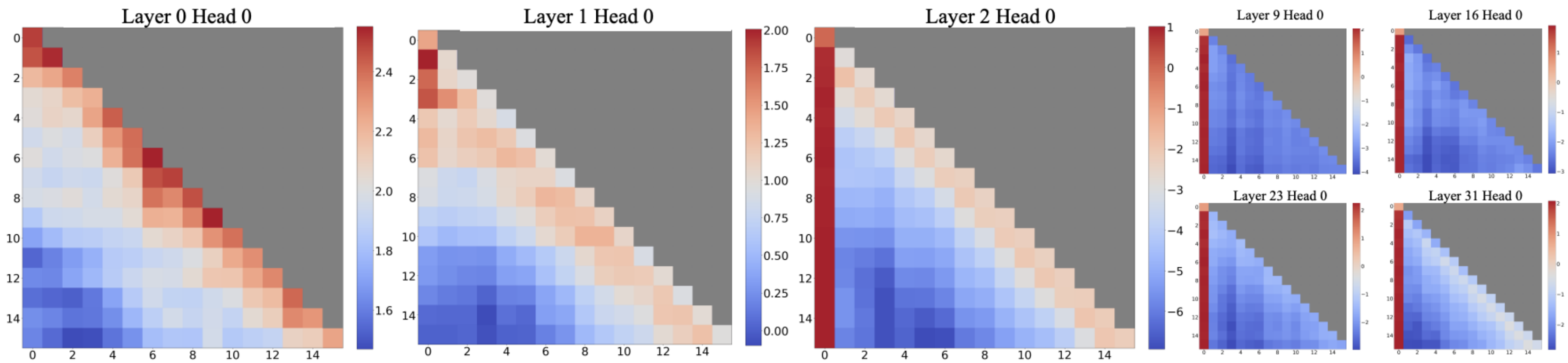
(a) Dense Attention
Current Token
T cached tokens
O(T²)✗ PPL: 5641✗
Has poor efficiency and performance on long text.
(b) Window Attention
T-L evicted tokens
L cached tokens
O(TL)✓ PPL: 5158✗
Breaks when initial tokens are evicted.
(c) Sliding Window w/ Re-computation
previous tokens are truncated
L re-computed tokens
O(TL²)✗ PPL: 5.43✓
Has to re-compute cache for each incoming token.
(d) StreamingLLM (ours)
Attention Sink
evicted tokens
L cached tokens
O(TL)✓ PPL: 5.40✓
Can perform efficient and stable language modeling on long texts.

Problem Definition

- Goal: stable, low-cost, effectively unbounded streaming inference without changing model weights
- Constraint: cannot keep full historical KV; quality should stay close to full attention
- Question: why does pure window attention fail, and which historical tokens are truly critical?

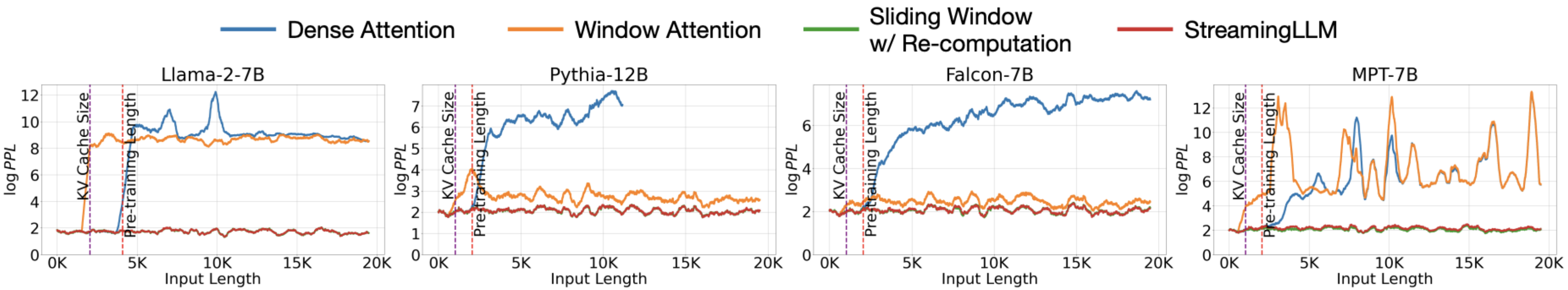
Proposed Solution: How and Why

- Observation: initial tokens consistently absorb attention mass in many heads (attention sinks)
- Method: cache policy becomes initial sink tokens + recent window tokens
- Why it works: sink tokens provide a stable destination for softmax attention and avoid instability after eviction
- Extension: adding a dedicated sink token during pretraining can reduce sink-token requirements at inference


Layer 0 Head 0
0
2
4
6
8
10
12
14
2.4
2.2
2.0
1.8
1.6
Layer 1 Head 0
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Layer 2 Head 0
1
0
−1
−2
−3
−4
−5
−6
Layer 9 Head 0
Layer 16 Head 0
Layer 23 Head 0
Layer 31 Head 0

Why do LLMs break when removing initial tokens' KV?

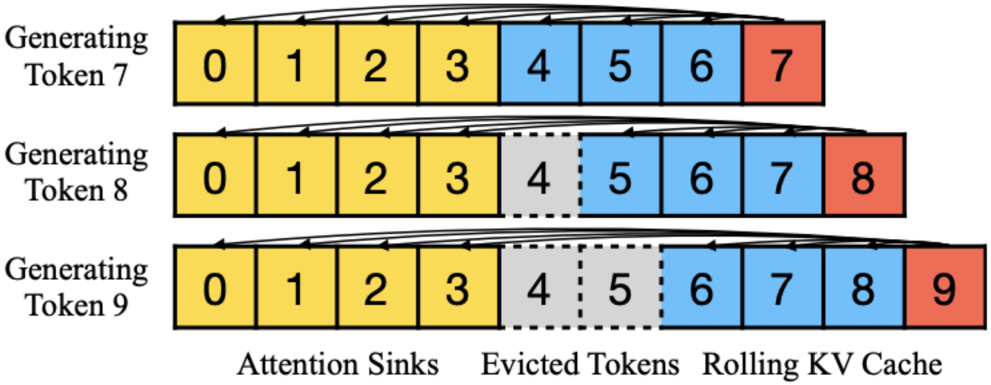
$$\mathrm{SoftMax}(x)_i = \frac{e^{x_i}}{e^{x_1} + \sum_{j=2}^{N} e^{x_j}}, \quad x_1 \gg x_j, j \in 2, \ldots, N$$

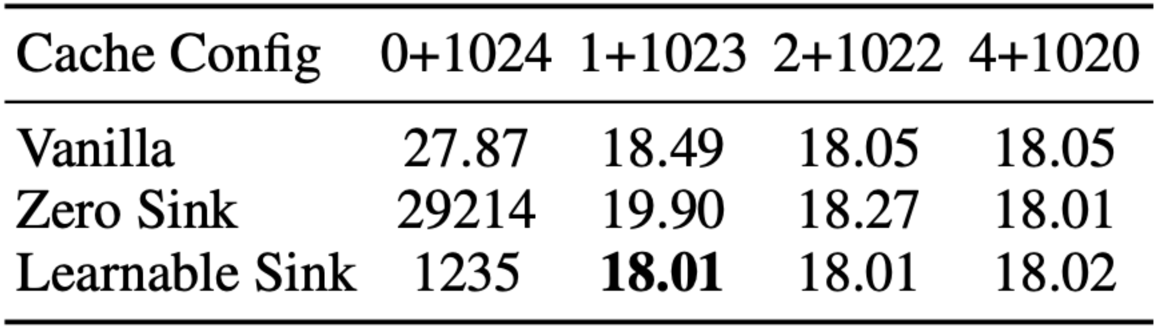

Dense Attention
Window Attention
Sliding Window w/ Re-computation
StreamingLLM
Llama-2-7B
Pythia-12B
Falcon-7B
MPT-7B
log PPL
KV Cache Size
Pre-training Length
Input Length
0K
5K
10K
15K
20K
0
2
4
6
8
10
12
1
3
5
7

Llama-2-13B	PPL (↓)
0 + 1024 (Window)	5158.07
4 + 1020	5.40
4"\n"+1020	5.60

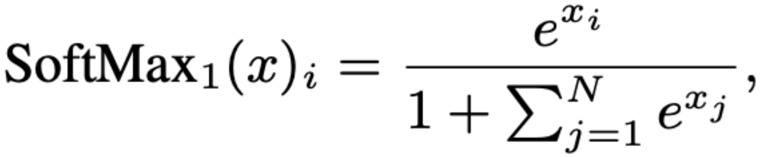
Cache Config	0+2048	1+2047	2+2046	4+2044	8+2040
Falcon-7B	17.90	12.12	12.12	12.12	12.12
MPT-7B	460.29	14.99	15.00	14.99	14.98
Pythia-12B	21.62	11.95	12.09	12.09	12.02

Cache Config	0+4096	1+4095	2+4094	4+4092	8+4088
Llama-2-7B	3359.95	11.88	10.51	9.59	9.54


Generating Token 7
0 1 2 3 4 5 6 7
Generating Token 8
0 1 2 3 4 5 6 7 8
Generating Token 9
0 1 2 3 4 5 6 7 8 9
Attention Sinks
Evicted Tokens
Rolling KV Cache



Cache Config	0+1024	1+1023	2+1022	4+1020
Vanilla	27.87	18.49	18.05	18.05
Zero Sink	29214	19.90	18.27	18.01
Learnable Sink	1235	**18.01**	18.01	18.02



$$\mathrm{SoftMax}_1(x)_i = \frac{e^{x_i}}{1 + \sum_{j=1}^{N} e^{x_j}},$$

Two ways:
add zero tokens to the front (no change of models)
Train new model with a fixed sink from scratch

RetrievalAttention: Accelerating Long-Context LLM Inference via Vector Retrieval

Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, Chen Chen, Fan Yang, Yuqing Yang, Lili Qiu

Motivation and Background

- Long-context bottlenecks include both compute and GPU-memory movement of KV cache
- Static/heuristic pruning often hurts complex retrieval-heavy tasks
- ANNS appears suitable for top-k attention retrieval, but naive usage performs poorly

Problem Definition

- Goal: training-free reduction of latency and memory footprint for long-context attention
- Core challenge: strong OOD gap between Query and Key distributions breaks off-the-shelf ANNS indexing
- Requirement: preserve near-full-attention accuracy while drastically reducing KV access

Proposed Solution: How and Why

- How-1: offload most KV to CPU and build attention-aware vector indexes; keep a small static KV set on GPU
- How-2: guide index construction with prefill query signals to mitigate Q-K OOD mismatch
- How-3: run CPU dynamic retrieval and GPU static attention in parallel, then combine partial outputs
- Why: leverages CPU memory capacity and GPU parallelism while avoiding full scans and heavy transfers

Prompt Length	128K	256K	512K	1M
Total Latency (s)	32.8	111	465	1,765
FFN (s)	7.6	15	31	70
Attention (s)	25.2	96	434	1,695
GPU Memory KV Cache (GB)	15.6	31.2	62.5	125

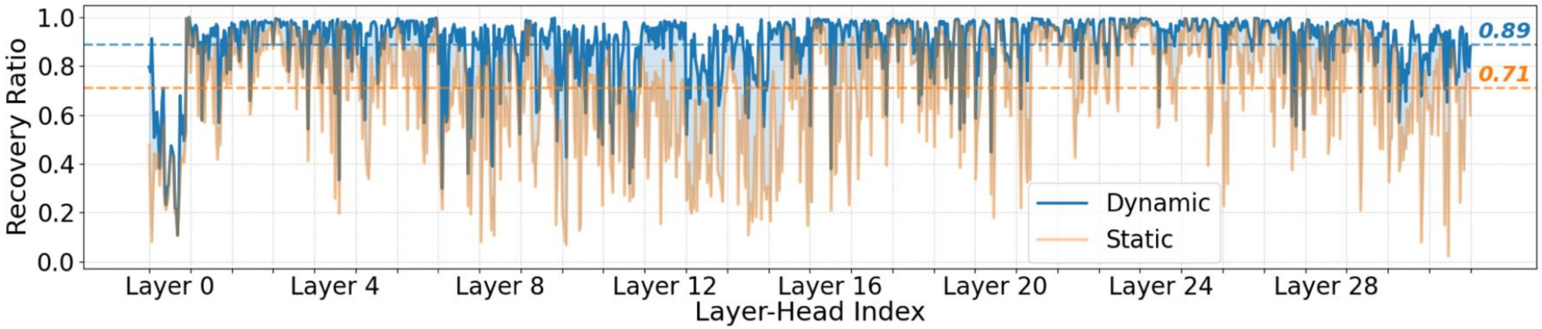
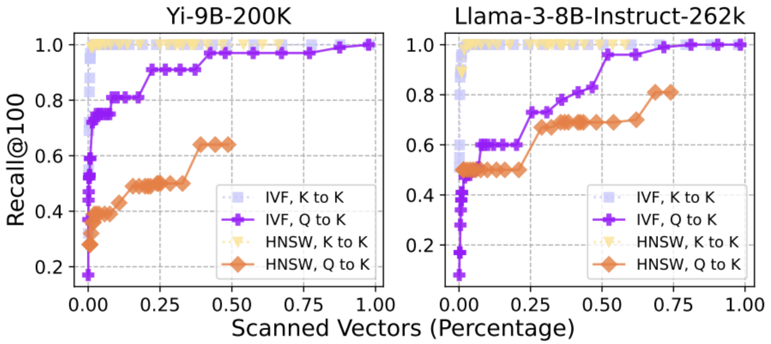
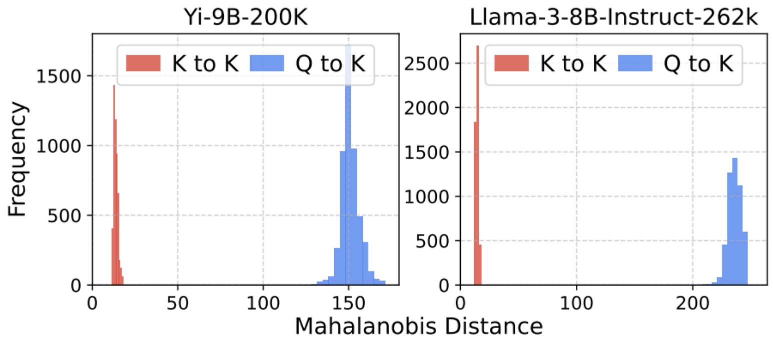


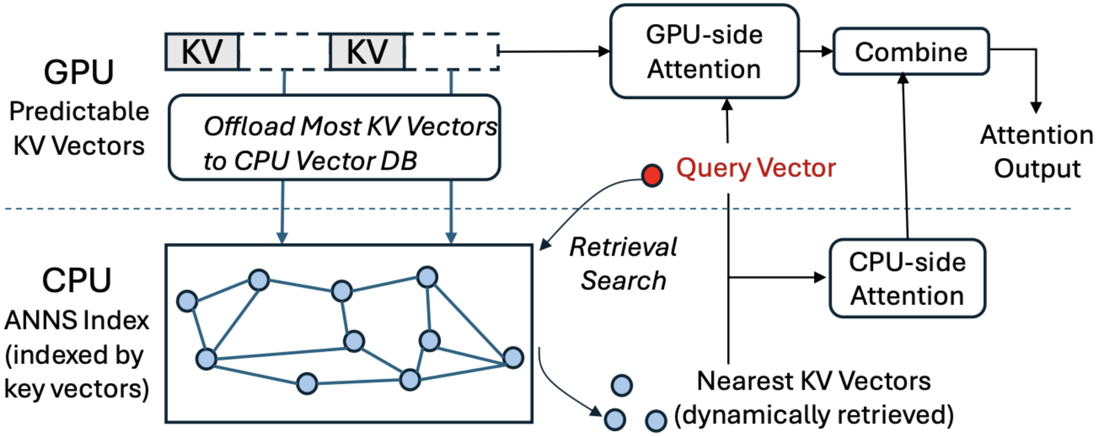
Figure 2: The dynamic sparsity of each layer and head in Llama-3-8B model in the KV retrieval test of 100,000 tokens. The blue curve shows that using dynamically selected top-1000 critical tokens achieves an average recovery ratio of 89%, indicating high attention sparsity. In contrast, the orange curve reveals that statically using the initially determined top-1000 critical tokens from the generation of the first token to generate subsequent tokens drops the average recovery ratio to 71%.



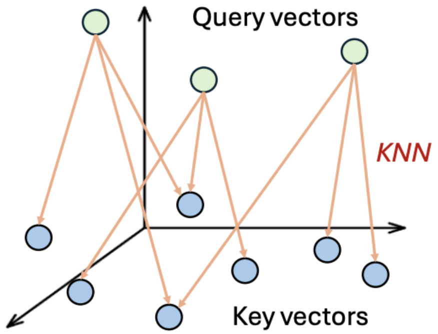
(a) ANNS index performance.



(b) Different distribution.



(a) Overall design of RetrievalAttention.



(b) Key building procedure of OOD-aware index.

$$\mathbf{o}_t = \sum_{i \in \mathcal{I}_{t,\epsilon}} a_{t,i} \cdot \mathbf{v}_i + \cancel{\sum_{i \notin \mathcal{I}_{t,\epsilon}} a_{t,i} \cdot \mathbf{v}_i} \approx \sum_{i \in \mathcal{I}_{t,\epsilon}} \tilde{a}_{t,i} \cdot \mathbf{v}_i \quad \text{where} \quad \tilde{a}_{t,i} = \frac{e^{z_i}}{\sum_{j \in \mathcal{I}_{t,\epsilon}} e^{z_j}} \qquad (2)$$

QUEST: Query-Aware Sparsity for Efficient Long-Context LLM Inference

Jiaming Tang [*,1,2] **Yilong Zhao** [*,1,3] **Kan Zhu** [3] **Guangxuan Xiao** [2] **Baris Kasikci** [3] **Song Han** [2,4]

Motivation and Background

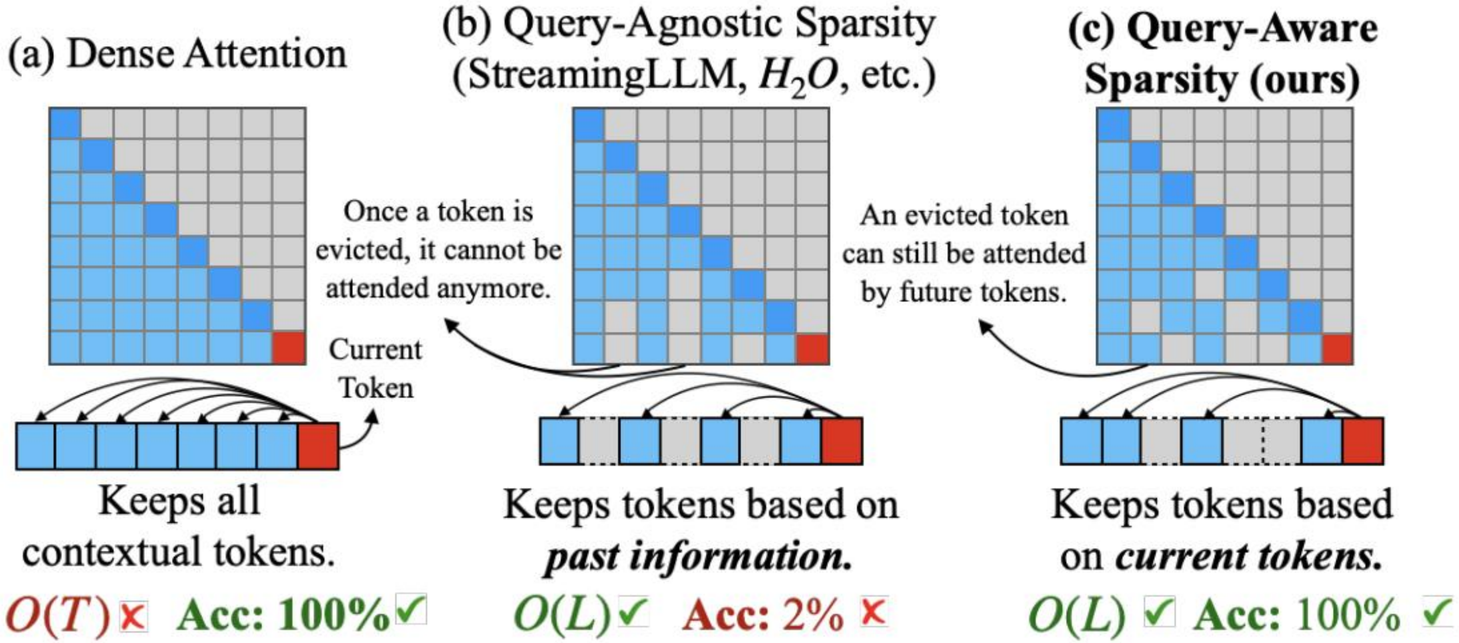
- Context windows have expanded to 128K and beyond, making attention memory traffic a dominant bottleneck
- Many prior methods rely on static/history-based importance and can drop tokens that become critical later
- Key observation: the same token can switch from unimportant to critical under a different query

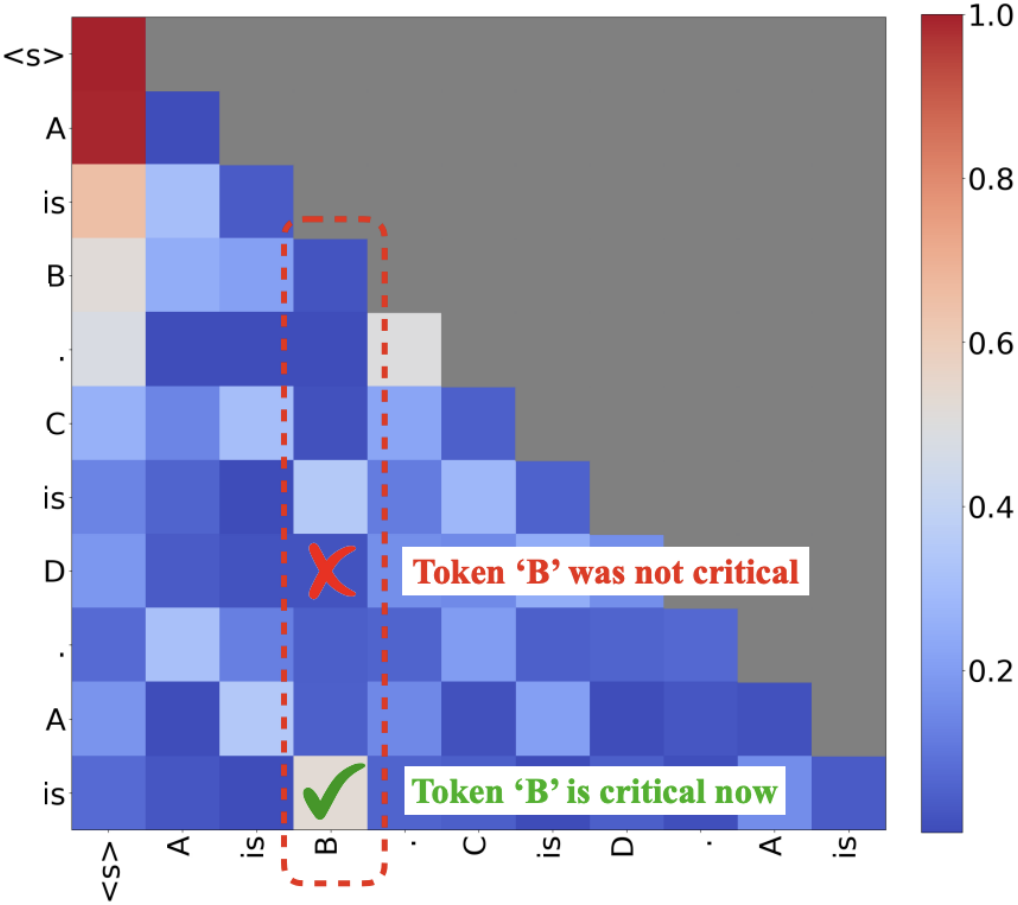
Problem Definition

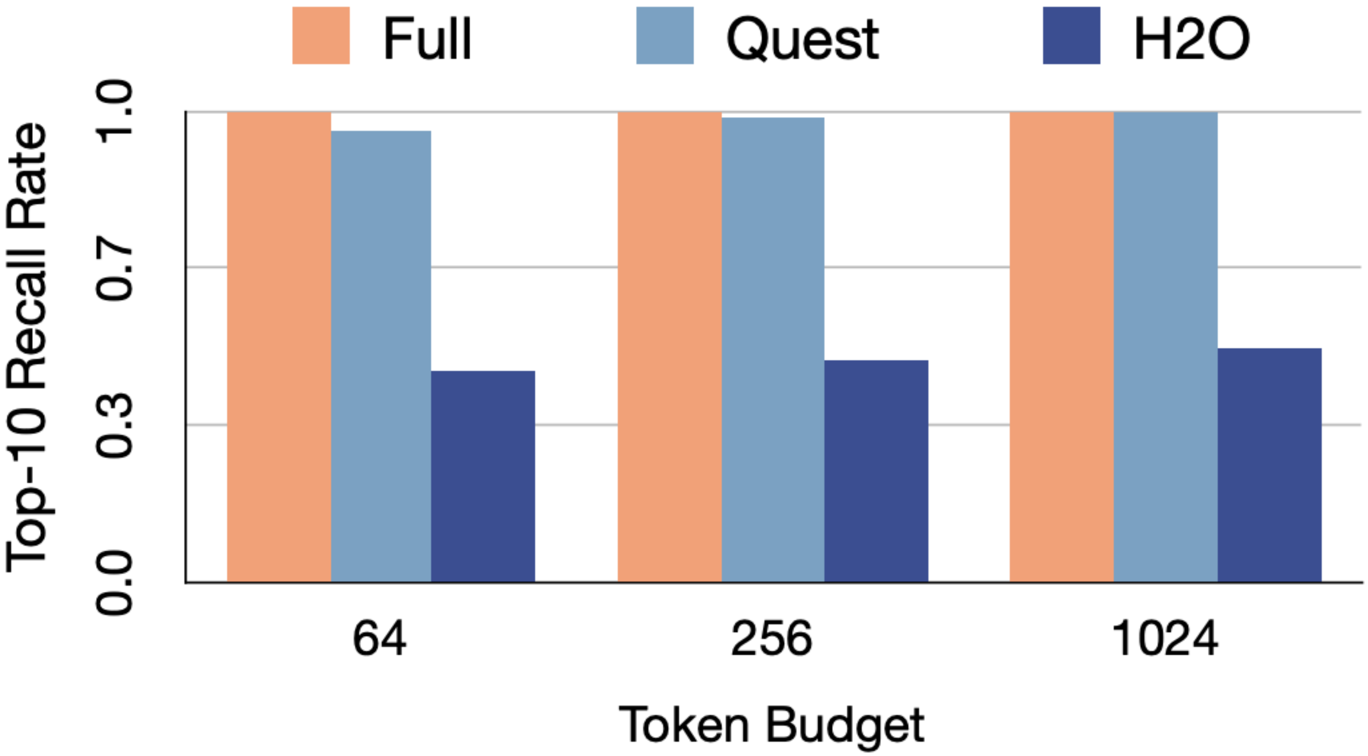
- Goal: dynamically identify KV regions critical to the current query and avoid full loading
- Constraint: criticality estimation must be cheap enough to preserve net speedup
- Design choice: select at page granularity to balance accuracy and overhead

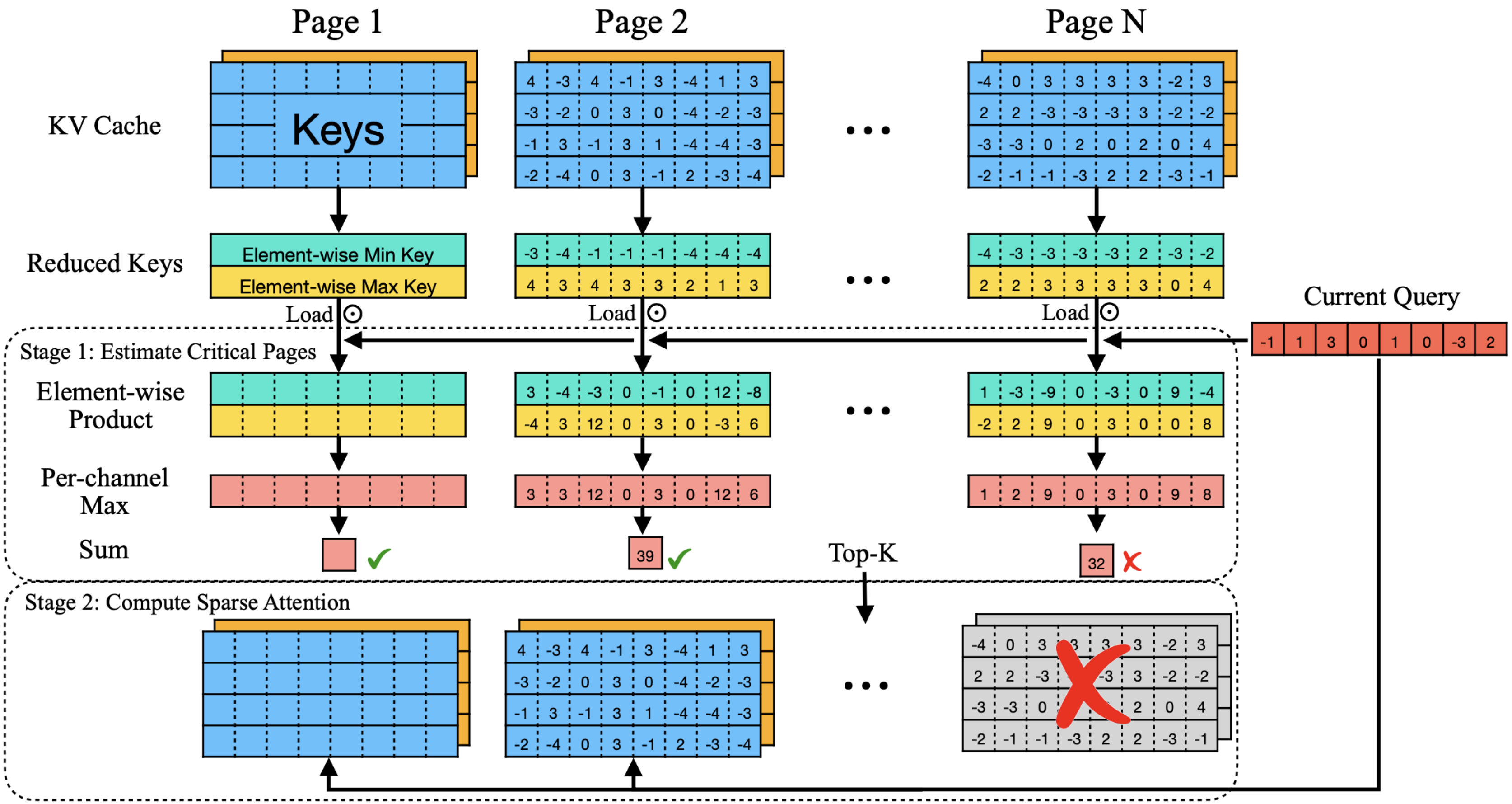
Proposed Solution: How and Why

- How-1: maintain per-page min/max key metadata
- How-2: estimate page criticality with current query using a lightweight upper-bound style scoring
- How-3: run regular attention only on Top-K selected pages
- Why: query-aware page-level selection captures dynamic relevance while reducing memory movement


(a) Dense Attention
Current Token
Keeps all contextual tokens.
$O(T)$ ✗ Acc: 100% ✓
(b) Query-Agnostic Sparsity (StreamingLLM, H_2O, etc.)
Once a token is evicted, it cannot be attended anymore.
Keeps tokens based on ***past information.***
$O(L)$ ✓ Acc: 2% ✗
(c) Query-Aware Sparsity (ours)
An evicted token can still be attended by future tokens.
Keeps tokens based on ***current tokens.***
$O(L)$ ✓ Acc: 100% ✓


<s>
A
is
B
.
C
is
D
.
A
is
1.0
0.8
0.6
0.4
0.2
Token ‘B’ was not critical
Token ‘B’ is critical now


Full
Quest
H2O
Top-10 Recall Rate
1.0
0.7
0.3
0.0
64
256
1024
Token Budget


Page 1
Page 2
Page N
KV Cache
Keys
Reduced Keys
Element-wise Min Key
Element-wise Max Key
Load ⊙
Current Query
-1 1 3 0 1 0 -3 2
Stage 1: Estimate Critical Pages
Element-wise Product
Per-channel Max
Sum
39
32
Top-K
Stage 2: Compute Sparse Attention

Algorithm 1 Token Criticality Estimation

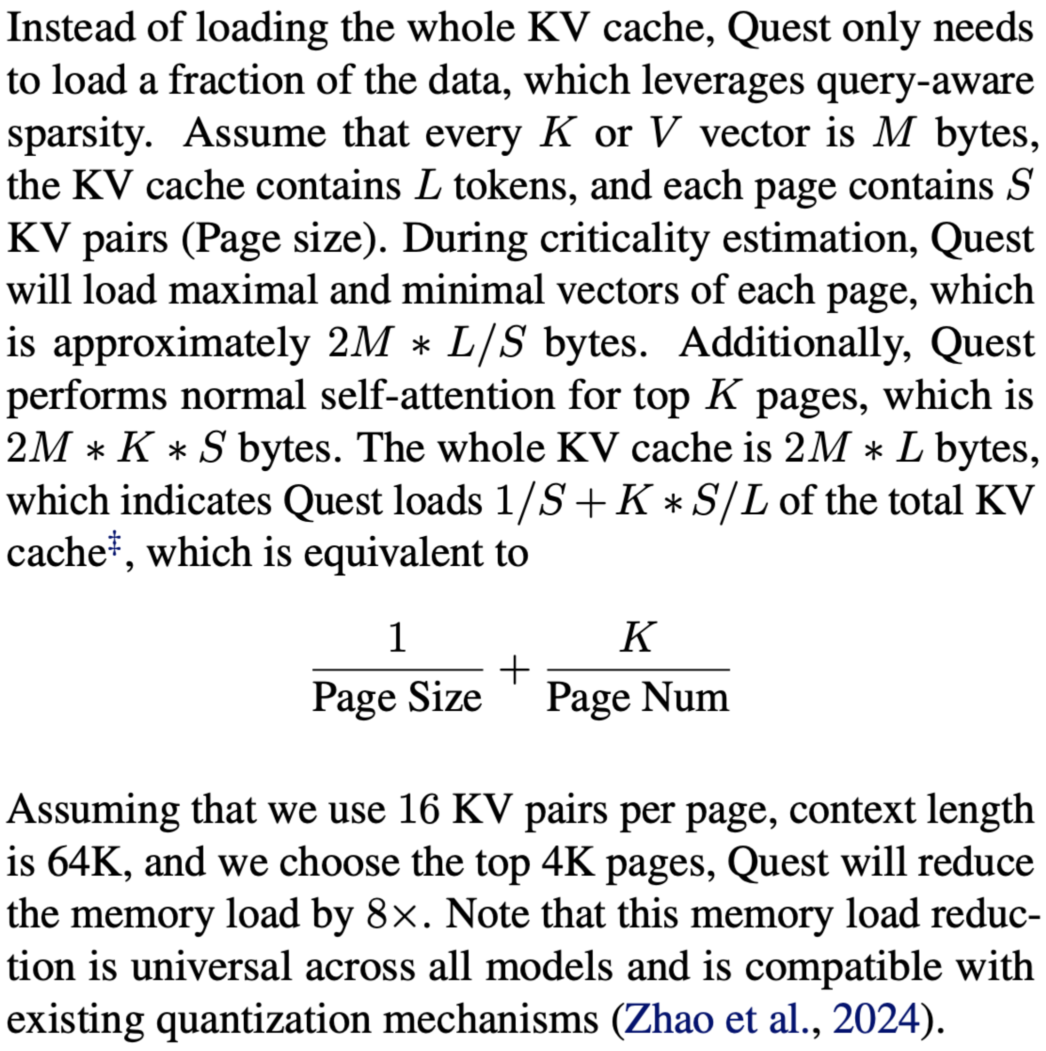
```
**When inserting new token to KV cache:**
**Input:** Key vector $K$, Dimension of hidden states $dim$,
Current maximal vector $M_i$, Current minimal vector $m_i$

**for** $i = 1$ **to** $dim$ **do**
    $M_i$ = $\max(M_i, k_i)$
    $m_i$ = $\min(m_i, k_i)$
**end for**

**When perform self-attention:**
**Input:** Query vector $Q$, Dimension of hidden states $dim$,
Current maximal vector $M_i$, Current minimal vector $m_i$

Initialize $score = 0$.
**for** $i = 1$ **to** $dim$ **do**
    $score$ += $MAX(q_i * max, q_i * min)$
**end for**
```

Typo



Instead of loading the whole KV cache, Quest only needs to load a fraction of the data, which leverages query-aware sparsity. Assume that every K or V vector is M bytes, the KV cache contains L tokens, and each page contains S KV pairs (Page size). During criticality estimation, Quest will load maximal and minimal vectors of each page, which is approximately $2M * L/S$ bytes. Additionally, Quest performs normal self-attention for top K pages, which is $2M * K * S$ bytes. The whole KV cache is $2M * L$ bytes, which indicates Quest loads $1/S + K * S/L$ of the total KV cache[‡], which is equivalent to

$$\frac{1}{\text{Page Size}} + \frac{K}{\text{Page Num}}$$

Assuming that we use 16 KV pairs per page, context length is 64K, and we choose the top 4K pages, Quest will reduce the memory load by $8\times$. Note that this memory load reduction is universal across all models and is compatible with existing quantization mechanisms (Zhao et al., 2024).

DuoAttention: Efficient Long-Context LLM Inference with Retrieval and Streaming Heads

Guangxuan Xiao[1] * **Jiaming Tang**[1] **Jingwei Zuo**[2] **Junxian Guo**[1,3]
Shang Yang[1] **Haotian Tang**[1] **Yao Fu**[4] **Song Han**[1,5]
[1] MIT [2] Tsinghua University [3] SJTU [4]University of Edinburgh [5] NVIDIA
`https://github.com/mit-han-lab/duo-attention`

Motivation and Background

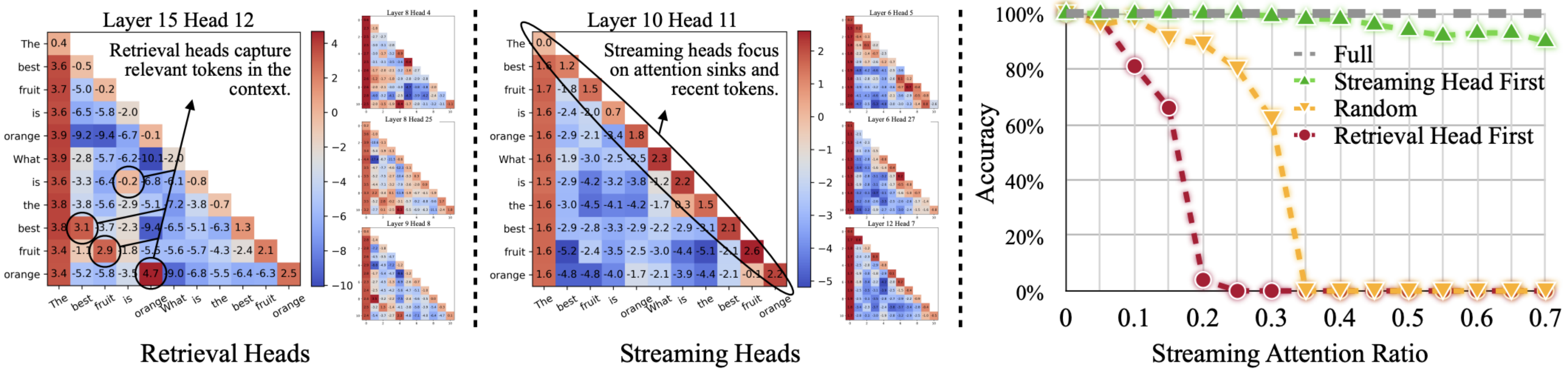
- Keeping full KV for all heads is expensive and unnecessary for many heads
- Token-level pruning can damage long-range retrieval when important tokens are removed
- This work compresses at head level based on functional head roles

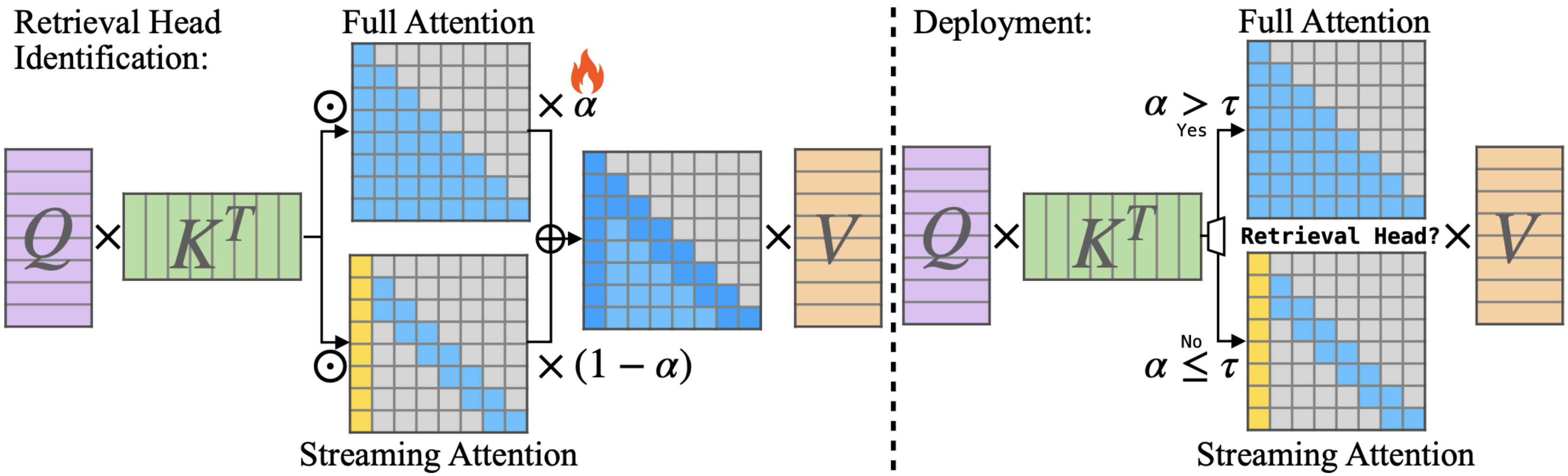
Problem Definition

- Goal: reduce decoding and prefilling memory/latency while preserving long-context quality
- Main difficulty: accurately detect non-compressible retrieval heads
- System requirement: work across MHA and GQA and remain compatible with existing inference stacks

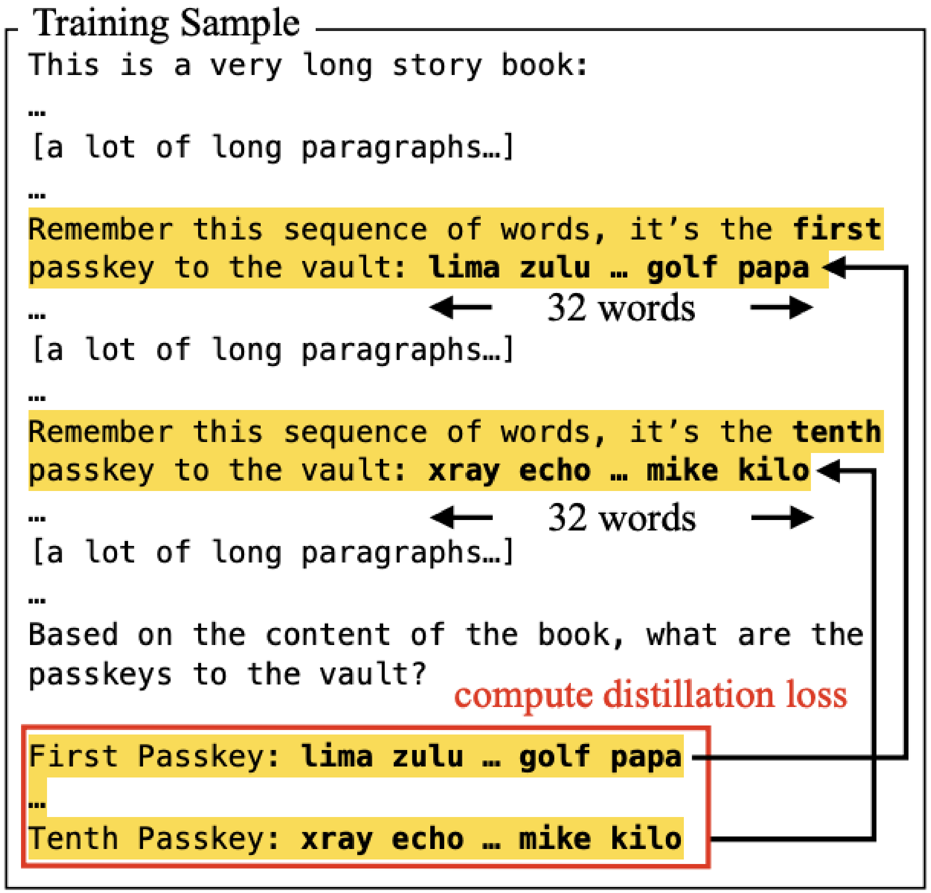
Proposed Solution: How and Why

- How-1: partition heads into Retrieval Heads and Streaming Heads
- How-2: full KV for retrieval heads, constant-length KV (recent + sinks) for streaming heads
- How-3: identify retrieval heads via optimization on synthetic data rather than static pattern inspection
- Why: concentrates compression on heads less sensitive to long-range context, improving efficiency with low quality loss

Layer 15 Head 12
Retrieval heads capture relevant tokens in the context.
The 0.4
best 3.6 -0.5
fruit 3.7 -5.0 -0.2
is 3.6 -6.5 -5.8 -2.0
orange 3.9 -9.2 -9.4 -6.7 -0.1
What 3.9 -2.8 -5.7 -6.2 -10.1 -2.0
is 3.6 -3.3 -6.4 -0.2 -6.8 -6.1 -0.8
the 3.8 -3.8 -5.6 -2.9 -5.1 -7.2 -3.8 -0.7
best 3.8 3.1 -3.7 -2.3 -9.4 -6.5 -5.1 -6.3 1.3
fruit 3.4 -1.1 2.9 -1.8 -5.5 -5.6 -5.7 -4.3 -2.4 2.1
orange 3.4 -5.2 -5.8 -3.5 4.7 -9.0 -6.8 -5.5 -6.4 -6.3 2.5
The best fruit is orange What is the best fruit orange
4 2 0 −2 −4 −6 −8 −10
Layer 8 Head 4
Layer 8 Head 25
Layer 9 Head 8
Retrieval Heads
Layer 10 Head 11
Streaming heads focus on attention sinks and recent tokens.
The 0.0
best 1.6 1.2
fruit 1.7 -1.8 1.5
is 1.6 -2.4 -2.0 0.7
orange 1.6 -2.9 -2.1 -3.4 1.8
What 1.6 -1.9 -3.0 -2.5 -2.5 2.3
is 1.5 -2.9 -4.2 -3.2 -3.8 -1.2 2.2
the 1.6 -3.0 -4.5 -4.1 -4.2 -1.7 0.3 1.5
best 1.6 -2.9 -2.8 -3.3 -2.9 -2.2 -2.9 -3.1 2.1
fruit 1.6 -5.2 -2.4 -3.5 -2.5 -3.0 -4.4 -5.1 -2.1 2.6
orange 1.6 -4.8 -4.8 -4.0 -1.7 -2.1 -3.9 -4.4 -2.1 -0.1 2.2
The best fruit is orange What is the best fruit orange
2 1 0 −1 −2 −3 −4 −5
Layer 6 Head 5
Layer 6 Head 27
Layer 12 Head 7
Streaming Heads
100%
80%
60%
40%
20%
0%
Accuracy
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7
Streaming Attention Ratio
Full
Streaming Head First
Random
Retrieval Head First


Retrieval Head Identification:
Full Attention
$\odot$
$\times \alpha$
Q
$\times$
K^T
$\oplus$
$\times$
V
$\odot$
$\times (1 - \alpha)$
Streaming Attention
Deployment:
Full Attention
$\alpha > \tau$
Yes
Q
$\times$
K^T
Retrieval Head?
$\times$
V
No
$\alpha \leq \tau$
Streaming Attention

Synthetic dataset



Training Sample

This is a very long story book:

…

[a lot of long paragraphs…]

…

Remember this sequence of words, it’s the **first** passkey to the vault: **lima zulu … golf papa**

…

[a lot of long paragraphs…]

…

Remember this sequence of words, it’s the **tenth** passkey to the vault: **xray echo … mike kilo**

…

[a lot of long paragraphs…]

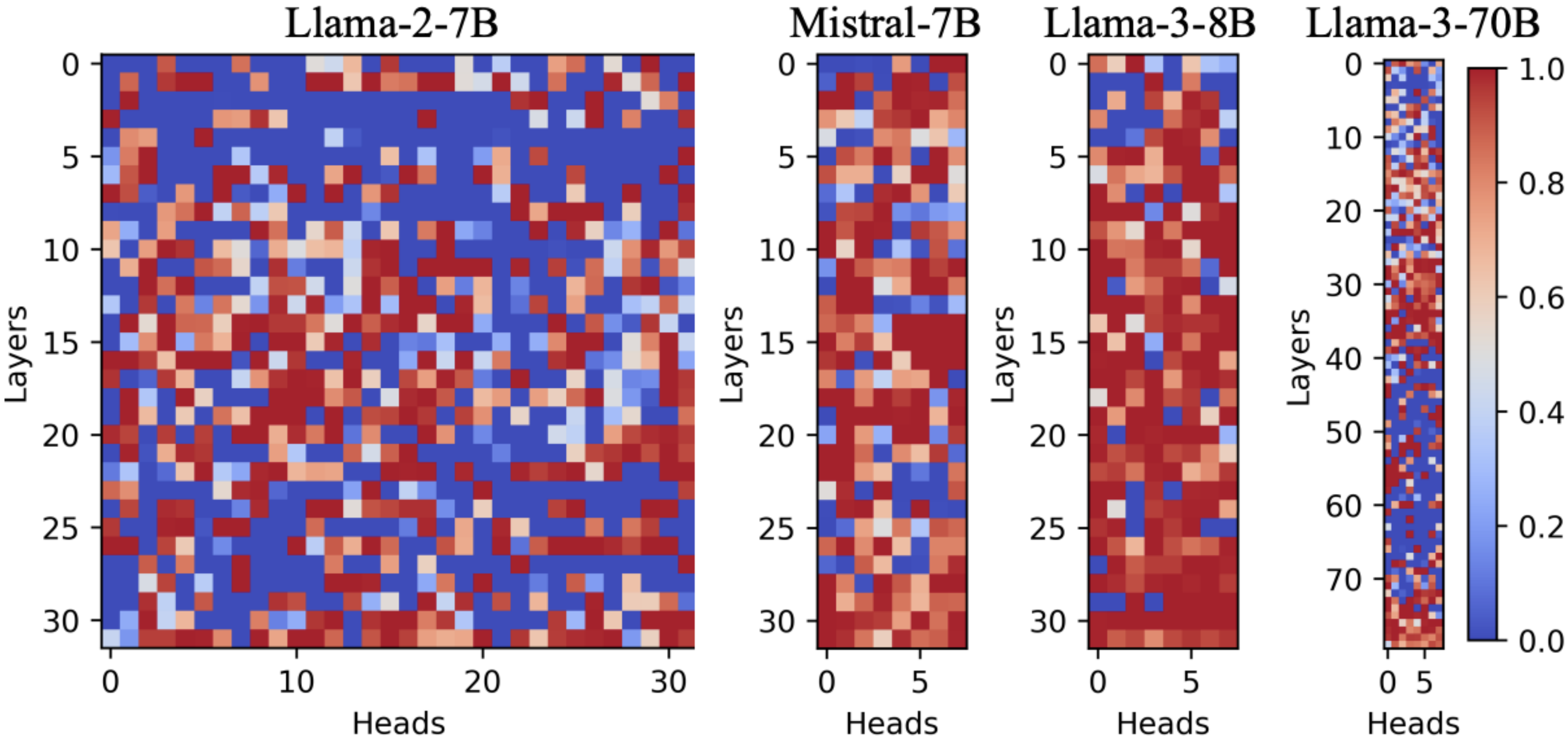
…

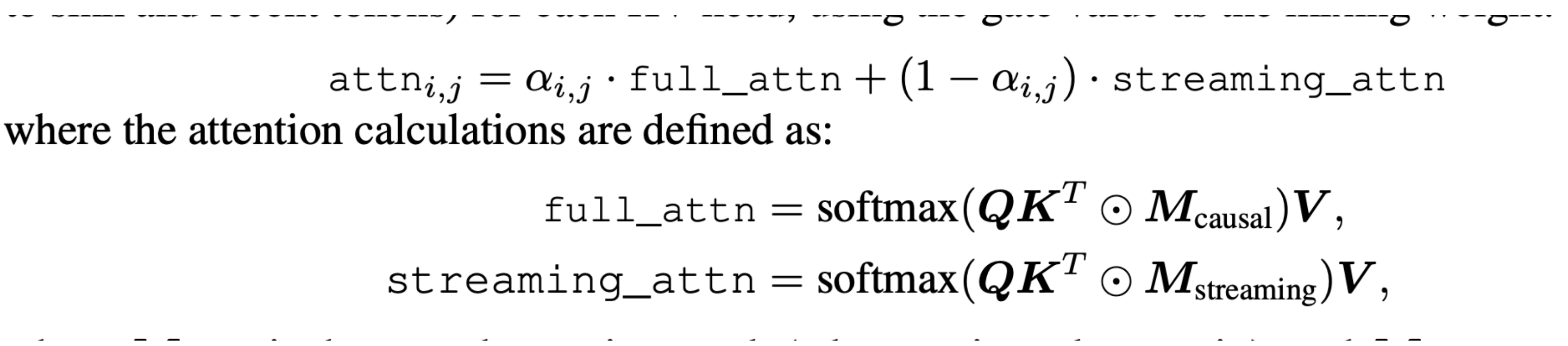
Based on the content of the book, what are the passkeys to the vault?

First Passkey: **lima zulu … golf papa**

…

Tenth Passkey: **xray echo … mike kilo**


Llama-2-7B
Mistral-7B
Llama-3-8B
Llama-3-70B
Layers
Heads
0
5
10
15
20
25
30
0
10
20
30
0
5
0
5
0
10
20
30
40
50
60
70
0
5
1.0
0.8
0.6
0.4
0.2
0.0

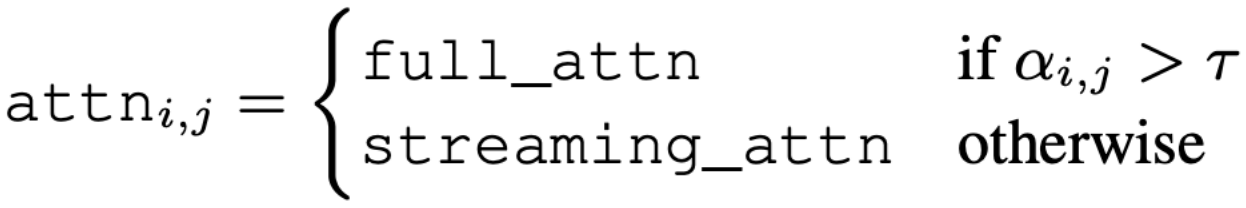


$$\texttt{attn}_{i,j} = \alpha_{i,j} \cdot \texttt{full\_attn} + (1 - \alpha_{i,j}) \cdot \texttt{streaming\_attn}$$

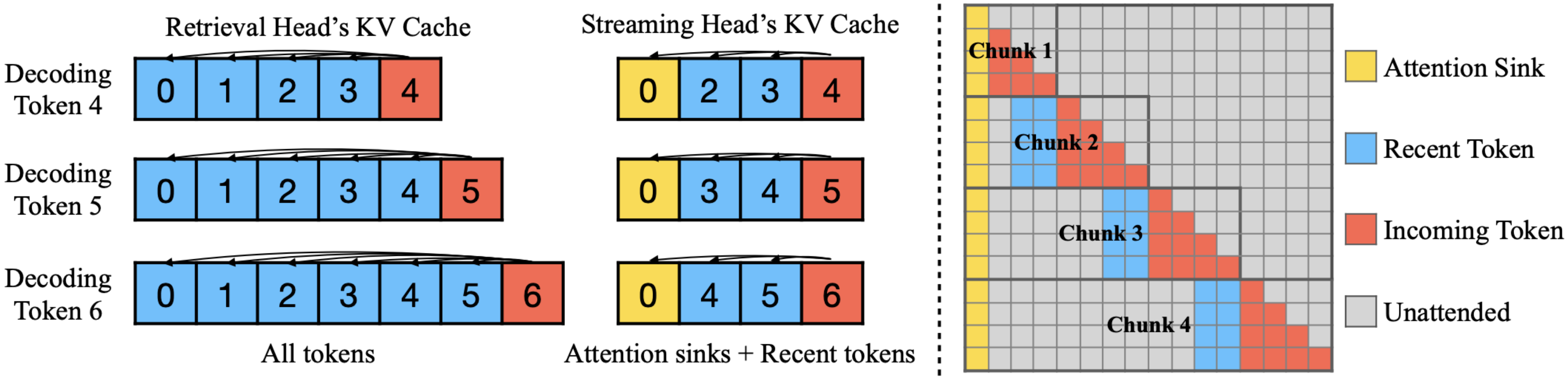
where the attention calculations are defined as:

$$\texttt{full\_attn} = \text{softmax}(\boldsymbol{Q}\boldsymbol{K}^T \odot \boldsymbol{M}_{\text{causal}})\boldsymbol{V},$$

$$\texttt{streaming\_attn} = \text{softmax}(\boldsymbol{Q}\boldsymbol{K}^T \odot \boldsymbol{M}_{\text{streaming}})\boldsymbol{V},$$



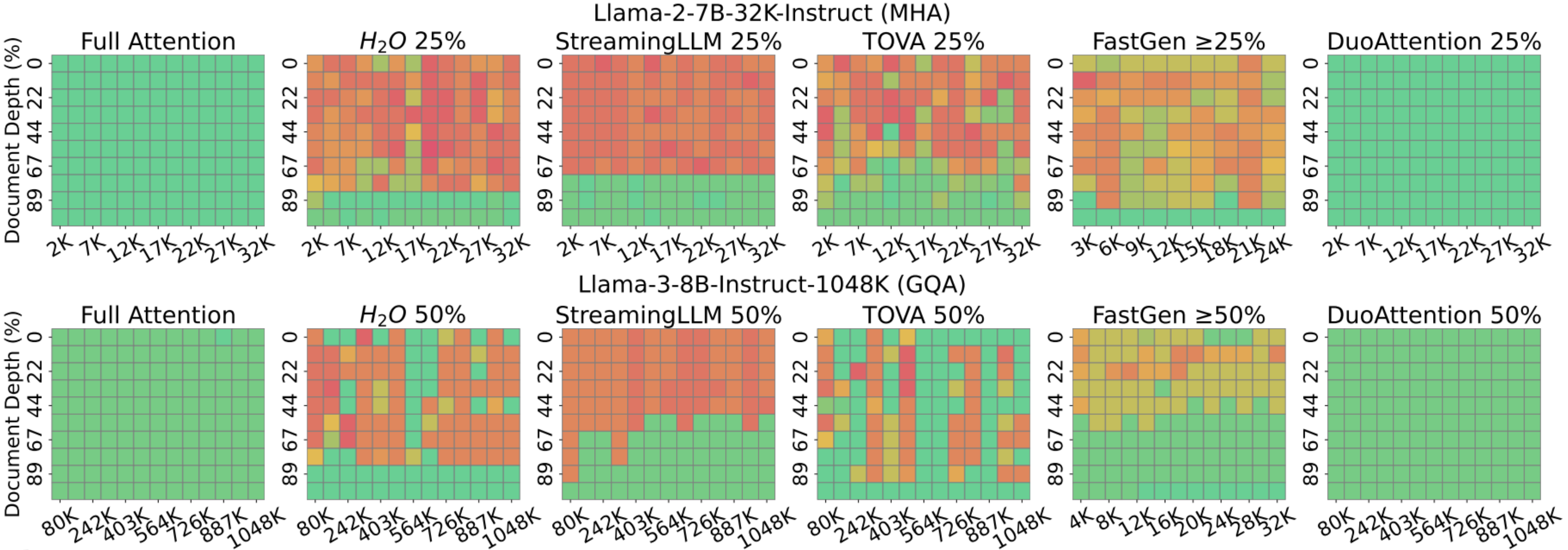
$$\texttt{attn}_{i,j} = \begin{cases} \texttt{full\_attn} & \text{if } \alpha_{i,j} > \tau \\ \texttt{streaming\_attn} & \text{otherwise} \end{cases}$$

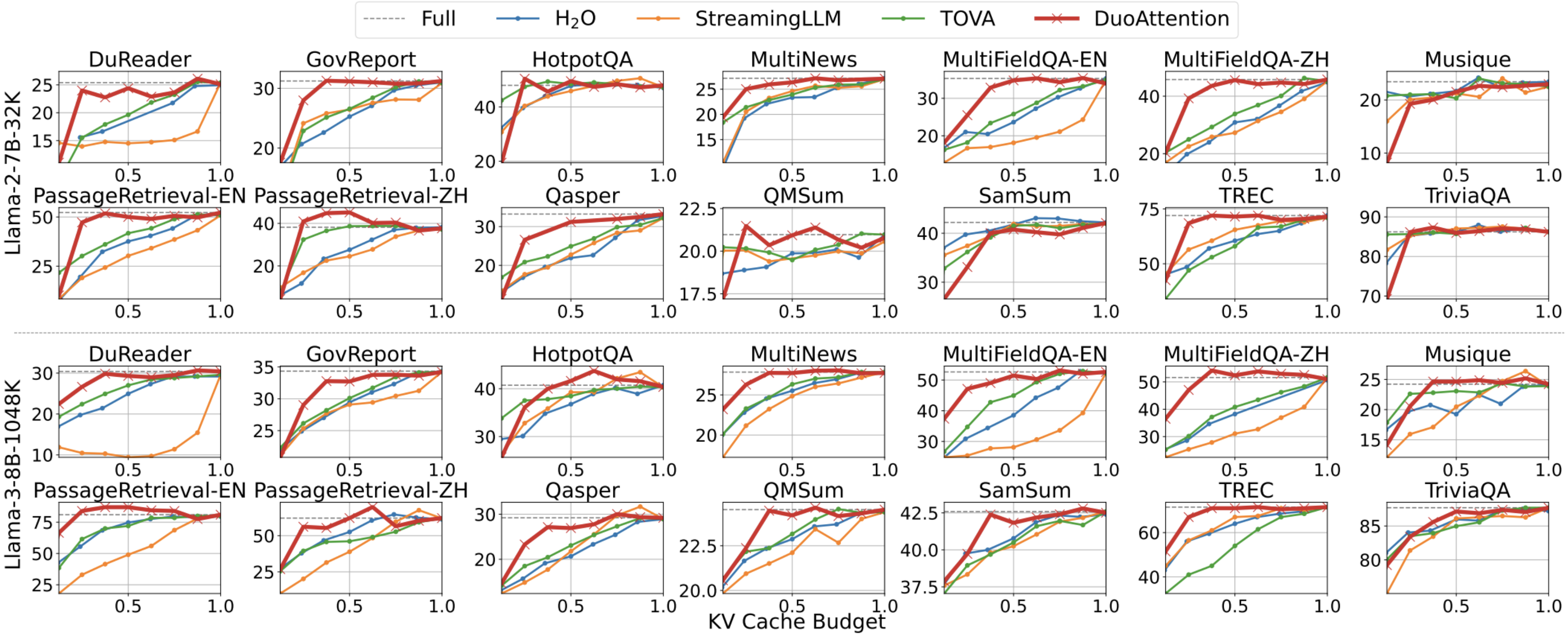

Retrieval Head's KV Cache
Streaming Head's KV Cache
Decoding Token 4
0 1 2 3 4
0 2 3 4
Decoding Token 5
0 1 2 3 4 5
0 3 4 5
Decoding Token 6
0 1 2 3 4 5 6
0 4 5 6
All tokens
Attention sinks + Recent tokens
Chunk 1
Chunk 2
Chunk 3
Chunk 4
Attention Sink
Recent Token
Incoming Token
Unattended

$$\mathcal{L}_{\text{distill}} = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=T-l+1}^{T} \left(\boldsymbol{H}_{\text{full}}^{(i)}[j] - \boldsymbol{H}_{\text{mixed}}^{(i)}[j] \right)^2$$

$$\mathcal{L}_{\text{reg}} = \sum_{i=1}^{\texttt{\#layers}} \sum_{j=1}^{\texttt{\#heads}} \left| \alpha_{i,j} \right|.$$

$$\mathcal{L} = \mathcal{L}_{\text{distill}} + \lambda \mathcal{L}_{\text{reg}}.$$


Llama-2-7B-32K-Instruct (MHA)
Full Attention
H_2O 25%
StreamingLLM 25%
TOVA 25%
FastGen ≥25%
DuoAttention 25%
Document Depth (%)
0
22
44
67
89
2K
7K
12K
17K
22K
27K
32K
3K
6K
9K
12K
15K
18K
21K
24K
Llama-3-8B-Instruct-1048K (GQA)
Full Attention
H_2O 50%
StreamingLLM 50%
TOVA 50%
FastGen ≥50%
DuoAttention 50%
80K
242K
403K
564K
726K
887K
1048K
4K
8K
12K
16K
20K
24K
28K
32K


Full
H$_2$O
StreamingLLM
TOVA
DuoAttention
Llama-2-7B-32K
DuReader
15
20
25
GovReport
20
30
HotpotQA
20
40
MultiNews
15
20
25
MultiFieldQA-EN
20
30
MultiFieldQA-ZH
20
40
Musique
10
20
PassageRetrieval-EN
25
50
PassageRetrieval-ZH
20
40
Qasper
20
30
QMSum
17.5
20.0
22.5
SamSum
30
40
TREC
50
75
TriviaQA
70
80
90
Llama-3-8B-1048K
DuReader
10
20
30
GovReport
25
30
35
HotpotQA
30
40
MultiNews
20
25
MultiFieldQA-EN
30
40
50
MultiFieldQA-ZH
30
40
50
Musique
15
20
25
PassageRetrieval-EN
25
50
75
PassageRetrieval-ZH
25
50
Qasper
20
30
QMSum
20.0
22.5
SamSum
37.5
40.0
42.5
TREC
40
60
TriviaQA
80
85
0.5
1.0
KV Cache Budget

LoongRL: Reinforcement Learning for Advanced Reasoning over Long Contexts

Siyuan Wang[*1,2] **Gaokai Zhang**[*1,3] **Li Lyna Zhang**[1‡]
Ning Shang[1] **Fan Yang**[1] **Dongyao Chen**[2] **Mao Yang**[1]
[1]Microsoft Research Asia [2]Shanghai Jiao Tong University [3]Carnegie Mellon University

Motivation and Background

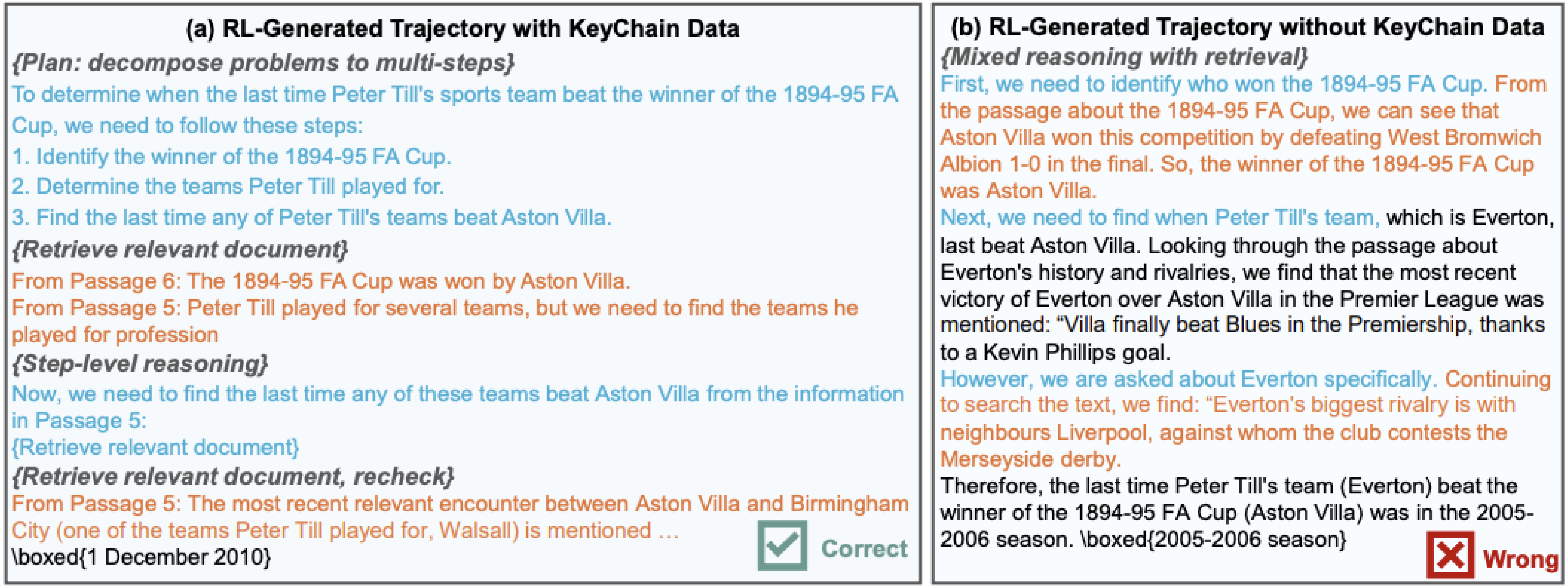
- Most RL reasoning advances focus on short-context tasks; long-context multi-hop reasoning remains underexplored
- Long-context systems often retrieve evidence but fail to execute robust reasoning chains
- High-difficulty long-context RL data is scarce

Problem Definition

- Goal: learn reliable multi-step reasoning patterns over long contexts, not only retrieval
- Challenge 1: full 128K RL rollouts are prohibitively expensive
- Challenge 2: with outcome-only rewards, avoid reward hacking and learn generalizable reasoning behavior

Proposed Solution: How and Why

- How-1: KeyChain synthesis converts short multi-hop QA into hard long-context tasks using UUID chains and distractors
- How-2: solving requires locating the true question, chain-tracing retrieval, multi-step reasoning, and re-checking
- How-3: GRPO-based multi-stage curriculum training introduces harder KeyChain data progressively
- Why: high-entropy non-semantic UUID chains force explicit stepwise reasoning instead of shortcut pattern matching



(a) RL-Generated Trajectory with KeyChain Data

{Plan: decompose problems to multi-steps}

To determine when the last time Peter Till's sports team beat the winner of the 1894-95 FA Cup, we need to follow these steps:

1. Identify the winner of the 1894-95 FA Cup.
2. Determine the teams Peter Till played for.
3. Find the last time any of Peter Till's teams beat Aston Villa.

{Retrieve relevant document}

From Passage 6: The 1894-95 FA Cup was won by Aston Villa.

From Passage 5: Peter Till played for several teams, but we need to find the teams he played for profession

{Step-level reasoning}

Now, we need to find the last time any of these teams beat Aston Villa from the information in Passage 5:

{Retrieve relevant document}

{Retrieve relevant document, recheck}

From Passage 5: The most recent relevant encounter between Aston Villa and Birmingham City (one of the teams Peter Till played for, Walsall) is mentioned …

\boxed{1 December 2010}

☑ Correct

(b) RL-Generated Trajectory without KeyChain Data

{Mixed reasoning with retrieval}

First, we need to identify who won the 1894-95 FA Cup. From the passage about the 1894-95 FA Cup, we can see that Aston Villa won this competition by defeating West Bromwich Albion 1-0 in the final. So, the winner of the 1894-95 FA Cup was Aston Villa.

Next, we need to find when Peter Till's team, which is Everton, last beat Aston Villa. Looking through the passage about Everton's history and rivalries, we find that the most recent victory of Everton over Aston Villa in the Premier League was mentioned: "Villa finally beat Blues in the Premiership, thanks to a Kevin Phillips goal.

However, we are asked about Everton specifically. Continuing to search the text, we find: "Everton's biggest rivalry is with neighbours Liverpool, against whom the club contests the Merseyside derby.

Therefore, the last time Peter Till's team (Everton) beat the winner of the 1894-95 FA Cup (Aston Villa) was in the 2005-2006 season. \boxed{2005-2006 season}

☒ Wrong

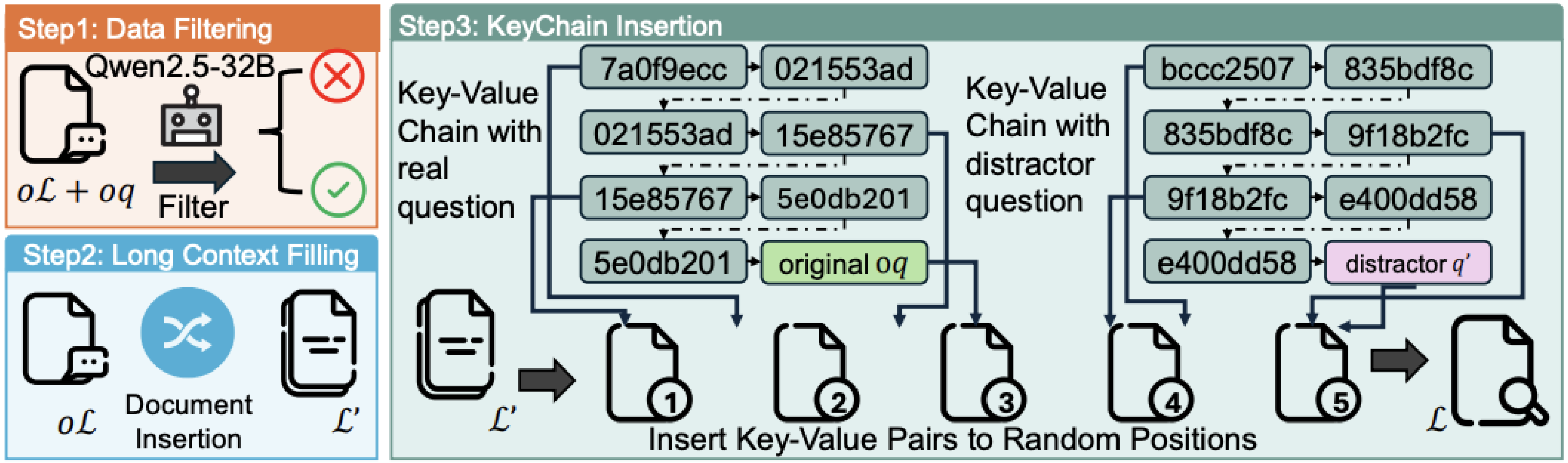


Figure 2: Overview of our KeyChain data construction.

Example of KeyChain-augmented long-context question

```
Please read the following text.
<Document 0>
<original text> {"UUIDB-n":  "distracting question"} <original text>
<Document 1>
{"UUIDA-1":  "UUIDA-2"}
<Document 2>
{"UUIDB-1":  "UUIDB-2"}
...
{"UUIDA-n":  "correct question"}
...
In the context above, there is one correct question to answer.
The correct question can only be found by following the correct
consecutive chain of key:value pairs encoded with UUID strings
(e.g., f81d4fae-7dec-11d0-a765-00a0c91e6bf6), starting from
"starting UUIDA-1".
Find the correct question first, then answer it.
```

$$J_{\mathrm{GRPO}}(\theta) = \mathbb{E}_{(\mathcal{L},q,a)\sim\mathcal{D},\,\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\mathrm{old}}}(\cdot|q)} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\Big(\min\left[\rho_{i,t}(\theta)A_{i,t}, \mathrm{clip}(\rho_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon)A_{i,t}\right] \quad -\beta D_{\mathrm{KL}}(\pi_\theta\|\pi_{\mathrm{ref}})\Big)\right] \tag{1}$$

$$\rho_{i,t}(\theta) \;=\; \frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\mathrm{old}}}(o_{i,t}|q,o_{i,<t})}.$$

$$A_{i,t} = \frac{r_i - \mathrm{mean}(\{r_1, r_2, \cdots, r_G\})}{\mathrm{std}(\{r_1, r_2, \cdots, r_G\})}$$

$$r_i = \begin{cases} 1 & \text{if } \{a \subseteq y_{\mathrm{ans}} \;\vee\; y_{\mathrm{ans}} \subseteq a\}, \\ 0 & \text{otherwise.} \end{cases}$$

Table 1: Data recipe for long-context RL training.

Dataset	**Description**	**# Size**	**Length range**	Difficulty
HotpotQA-KeyChain	KeyChain-augmented HotpotQA	2,500	16,272–20,670	Hard
MuSiQue-KeyChain	KeyChain-augmented MuSiQue	2,500	16,495–20,623	Hard
2WikiMultiHopQA-KeyChain	KeyChain-augmented 2WikiMultiHopQA	2,500	14,911–20,576	Hard
HotpotQA	Standard multi-hop QA	2,500	12,058–16,279	Medium
MuSiQue	Standard multi-hop QA	2,500	12,562–16,283	Medium
2WikiMultiHopQA	Standard multi-hop QA	2,500	10,727–16,274	Medium
Book RULER (Multi-key)	Long-context retrieval (20 keys, 1 value)	512	12,038–17,387	Easy
Book RULER (Multi-value)	Long-context retrieval (1 key, 20 values)	512	11,648–17,840	Hard
Math Choice	Multiple-choice math problems	2,500	40–425	Easy
DAPO Math	Mathematical reasoning	2,500	65–1,014	Hard