CS 498: Machine Learning System Spring 2026

Minjia Zhang

The Grainger College of Engineering

Today

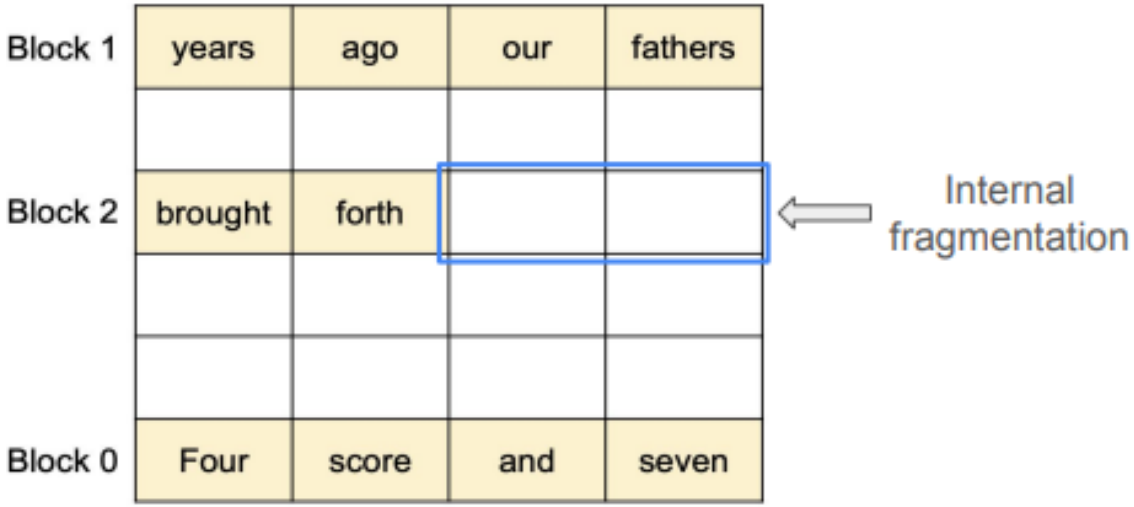
DL Inference

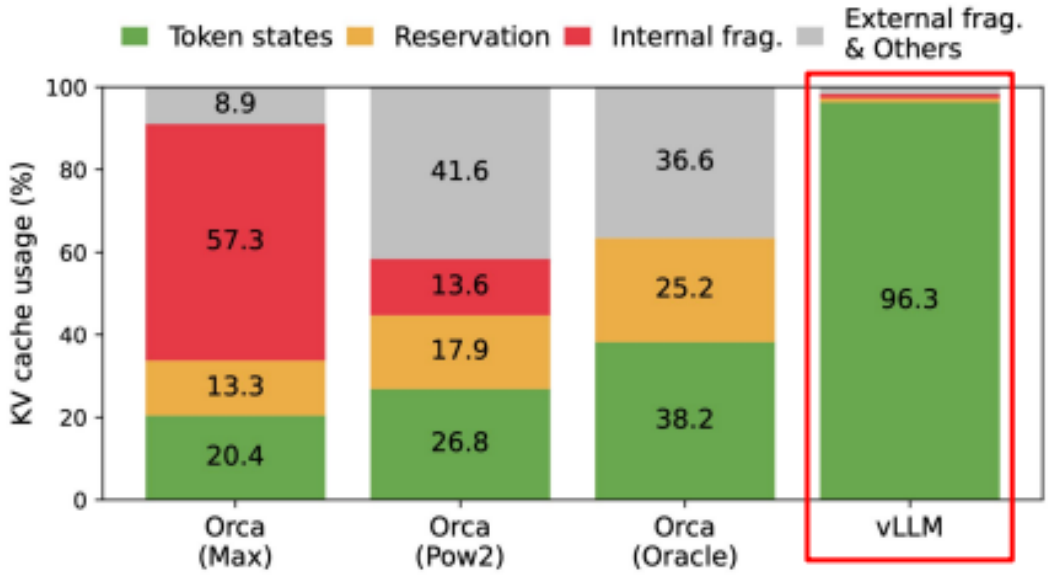
- H2O
- Attention Sink

Objective: Explain the motivation and techniques of adaptive KV cache, using H2O and Attention Sink as case studies

Memory Efficiency of PagedAttention

- Minimal internal fragmentation
  - # of wasted tokens per sequence < block size
- No external fragmentation

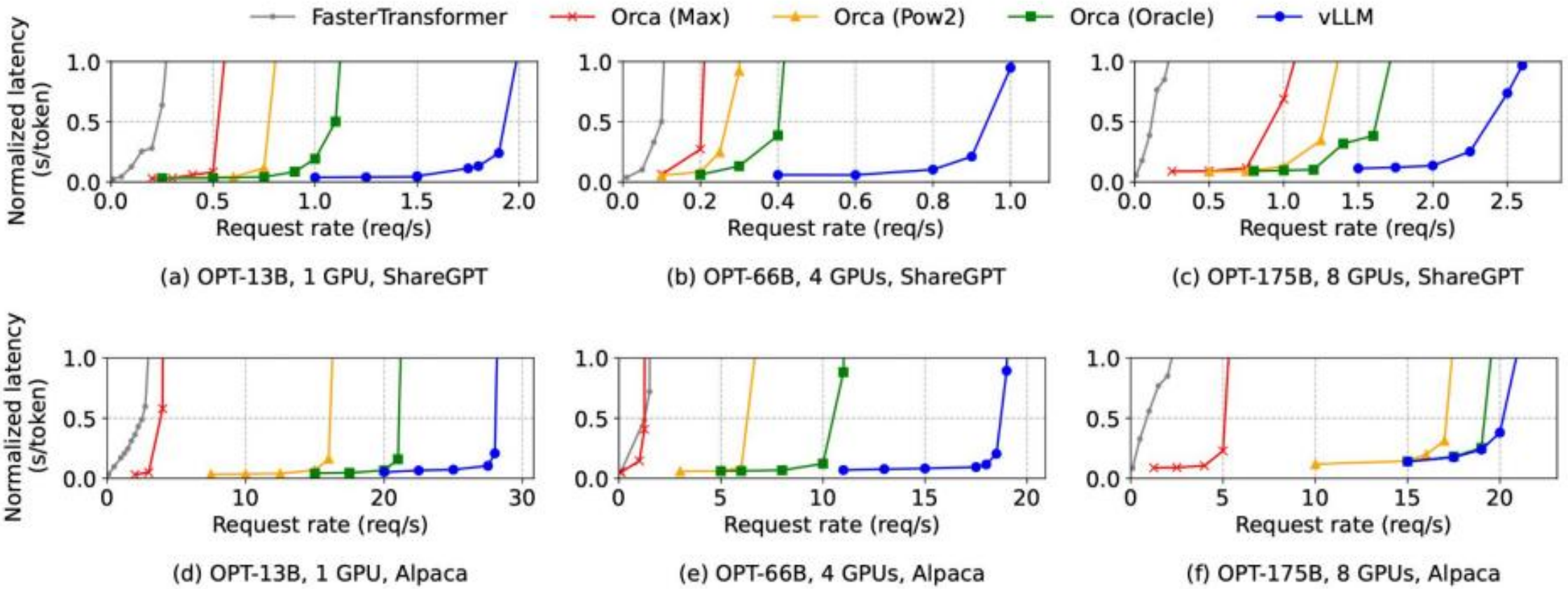




- With PagedAttention, wasted KV cache space is < 4% (3-5x improved memory utilization)

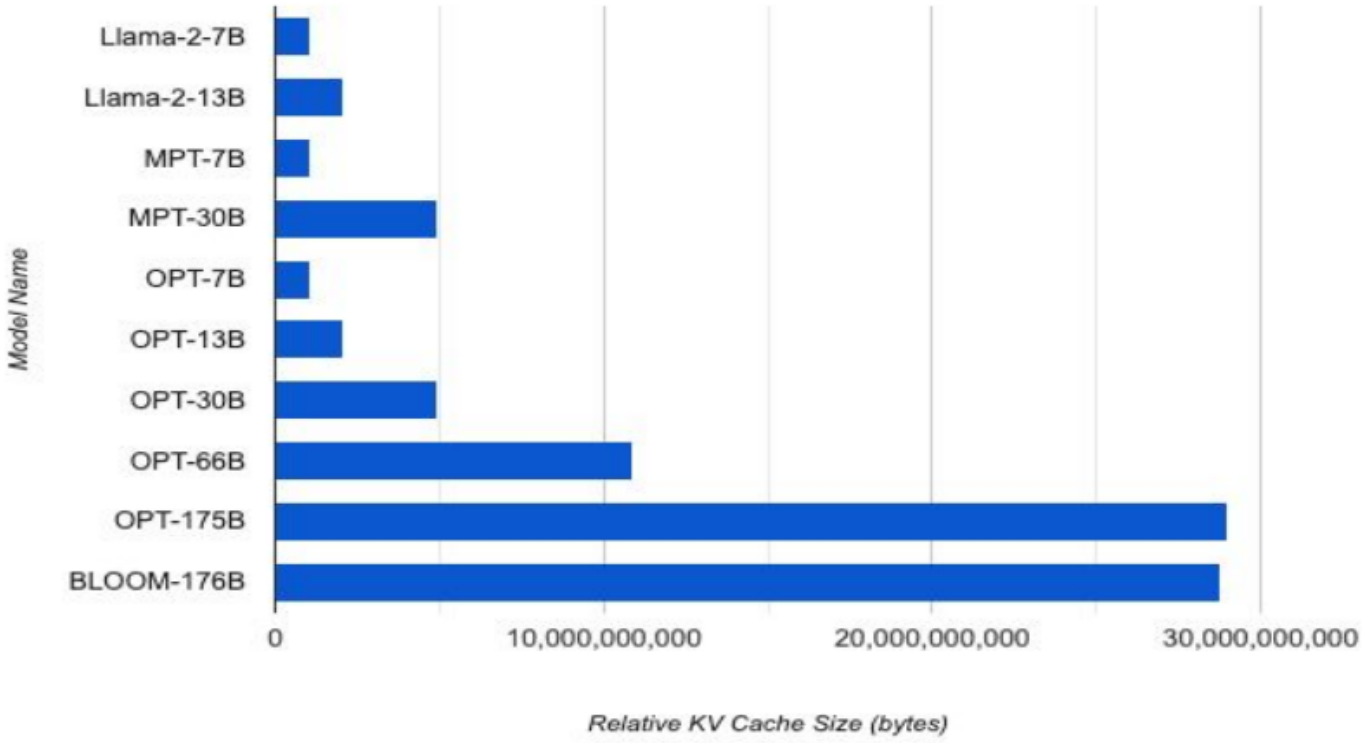
vLLM Performance with Basic Generation

- one sample per request on three models and two datasets



KV Cache Grows Fast

Relative KV cache size for different models



KV cache size: (2 x B x L x S x d x P) bytes

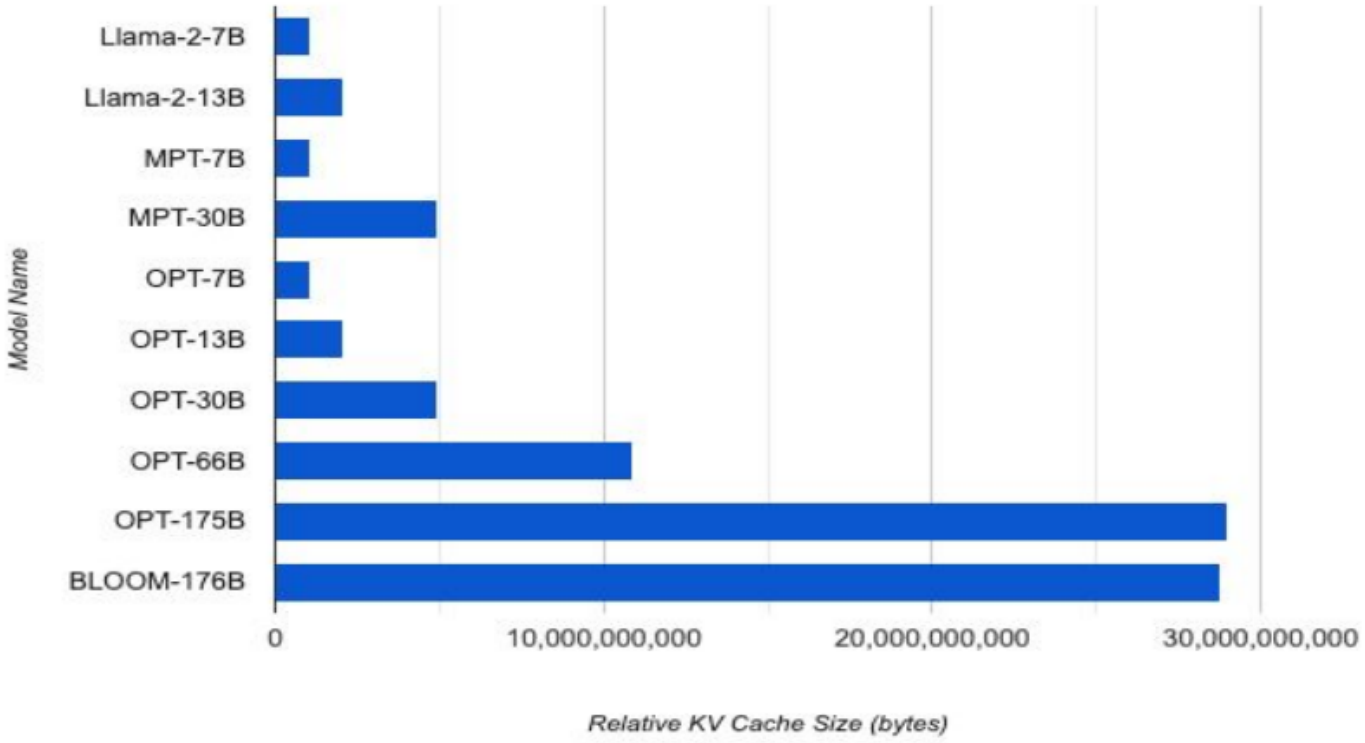
where:
B = batch size
L = # of layers
S = sequence length
D = hidden dimension
P = precision

LLaMA-2-7B with a batch size of 64 and sequence length of 1024 requires a cache size of 32GB

NVIDIA A100 GPU has ~ 40 GB - 80 GB

KV Cache Grows Fast

Relative KV cache size for different models



KV cache size: (2 x B x L x S x d x P) bytes

where:
B = batch size
L = # of layers
S = sequence length
D = hidden dimension
P = precision

LLaMA-2-7B with a batch size of 64 and sequence length of 1024 requires a cache size of 32GB

NVIDIA A100 GPU has ~ 40 GB - 80 GB

Question: KV cache is still quite expensive. Can we further reduce its size?

Motivation for Adaptive Caching

- Full KV cache with paged attention is memory-efficient, but not content-aware.

- Can we evict or compress KV pairs that are unlikely to influence future predictions?

Challenges with Cache Eviction

- Ideal Cache?

Challenges with Cache Eviction

- Ideal Cache?
  - i. Small cache size to reduce memory footprint
  - ii. Low miss rate
  - iii. Low-cost eviction policy

Challenges with Cache Eviction

- Ideal KV Cache?
  - i. Small cache size to reduce memory footprint
  - ii. Low miss rate
  - iii. Low-cost eviction policy

Challenges with Cache Eviction

- Ideal KV Cache

  i. Small cache size to reduce memory footprint

  - Challenge: Whether the size of KV cache can be restricted?

  ii. Low miss rate

  iii. Low-cost eviction policy

Challenges with Cache Eviction

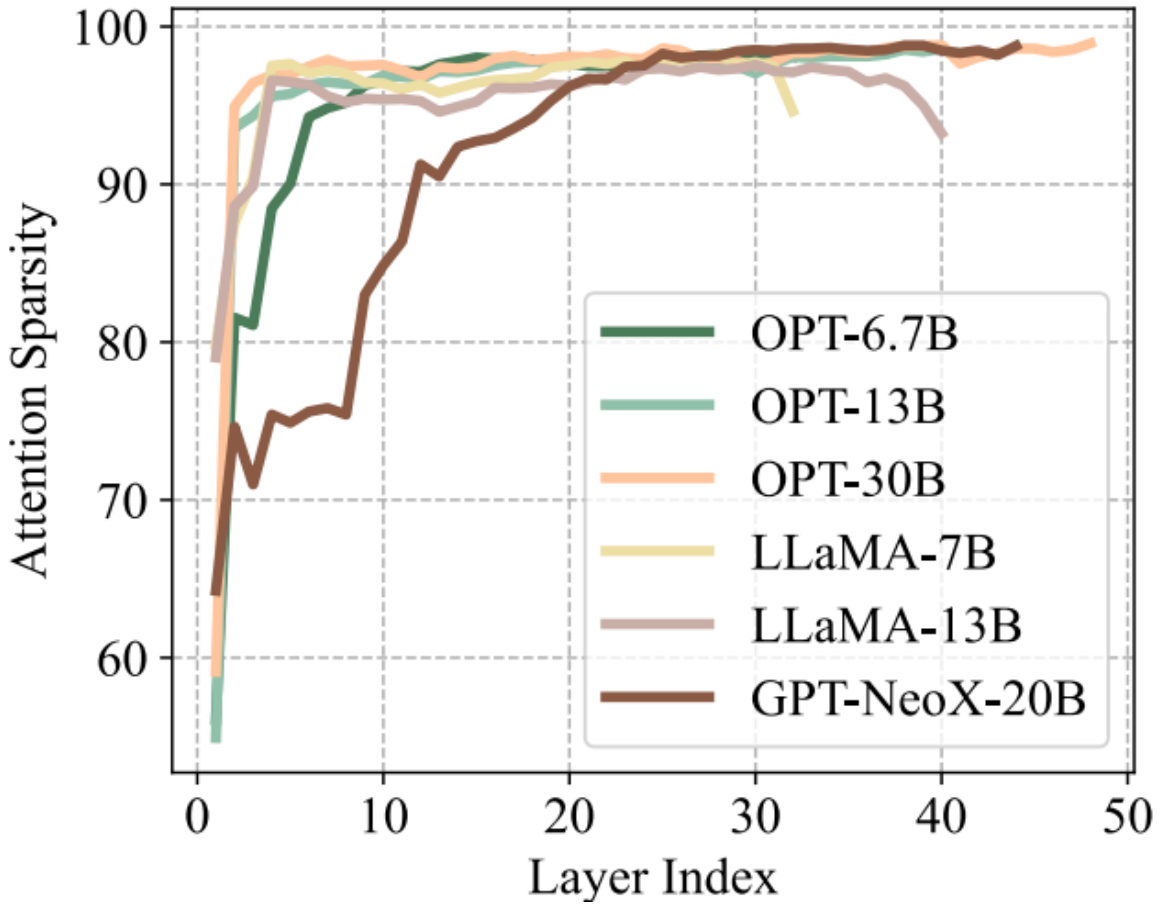
- Ideal KV Cache
  - i. Small cache size to reduce memory footprint
    - Challenge: Whether the size of KV cache can be restricted?
  - ii. Low miss rate
    - Challenge: Evicted KV embeddings can not be accessed in the future. How to maintain the performance and long-content generation accuracy?
  - iii. Low-cost eviction policy

Challenges with Cache Eviction

- Ideal KV Cache
  - i. Small cache size to reduce memory footprint
    - Challenge: Whether the size of KV cache can be restricted?
  - ii. Low miss rate
    - Challenge: Evicted KV embeddings can not be accessed in the future. How to maintain the performance and long-content generation accuracy?
  - iii. Low-cost eviction policy
    - Challenge: Combination problem, brute force policy might lead to high latency

Observation 1

- Sparsity for small cache size

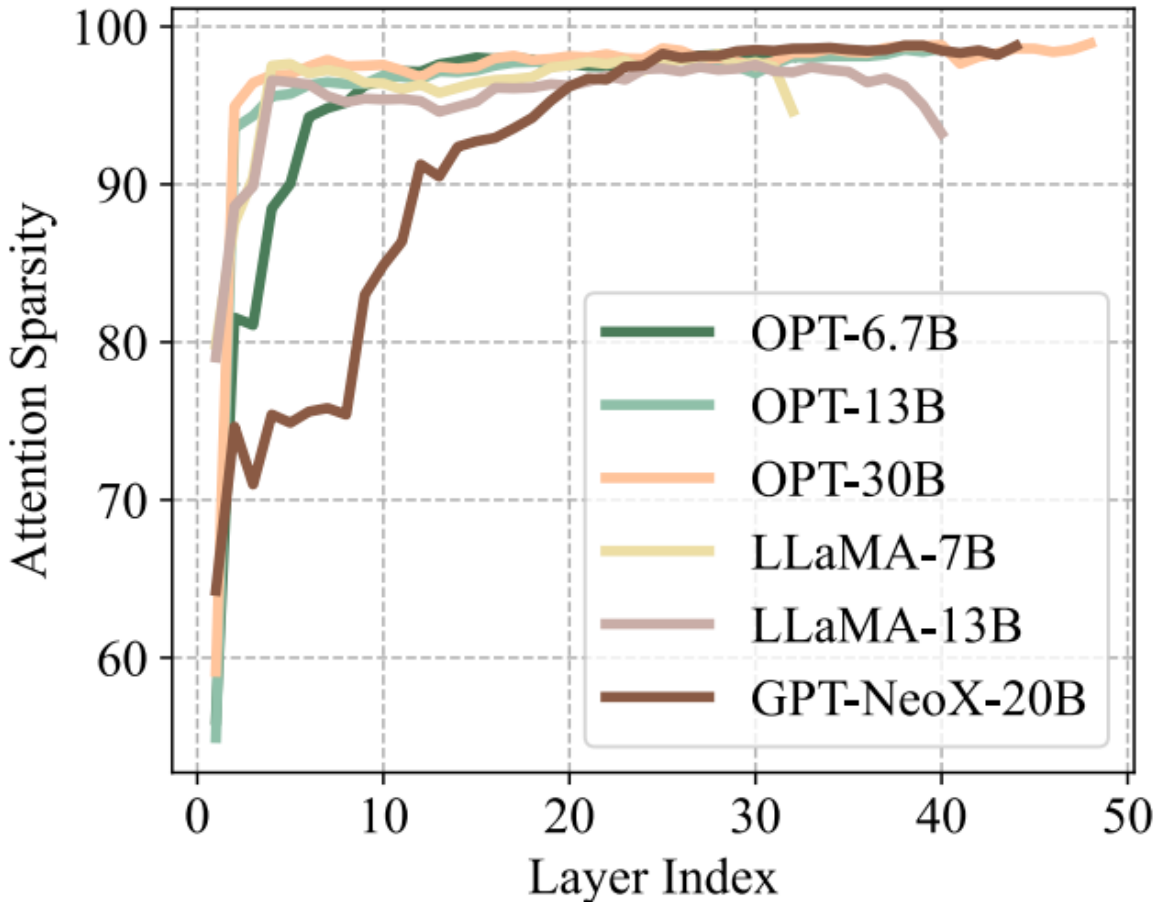


- Attention Sparsity

  - Threshold = 1% x Maximum attention score from softmax (QK^T)

Observation 1

- Sparsity for small cache size



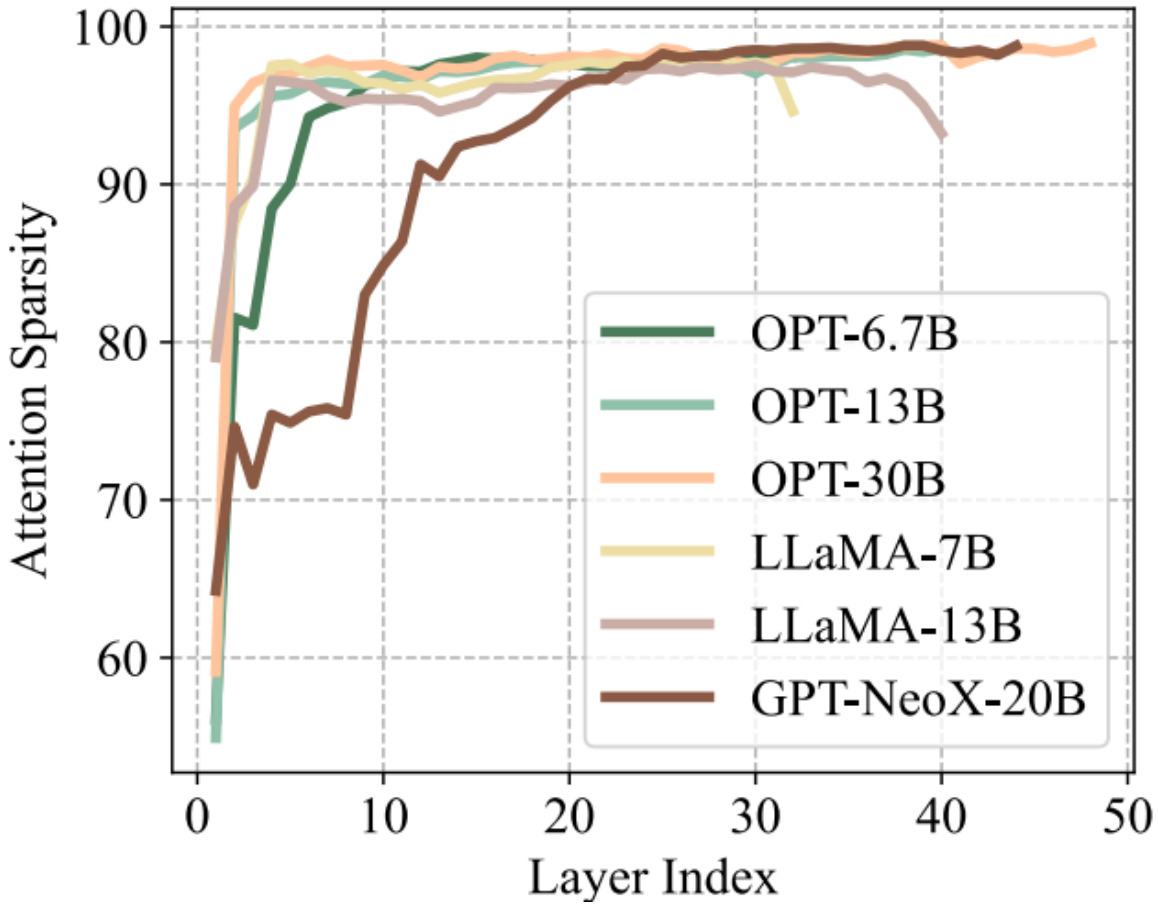
- Attention Sparsity

  - Threshold = 1% x Maximum attention score from softmax (QK^T)

- Sparsity over 95% in almost all layers

Observation 1

- Sparsity for small cache size



- Attention Sparsity

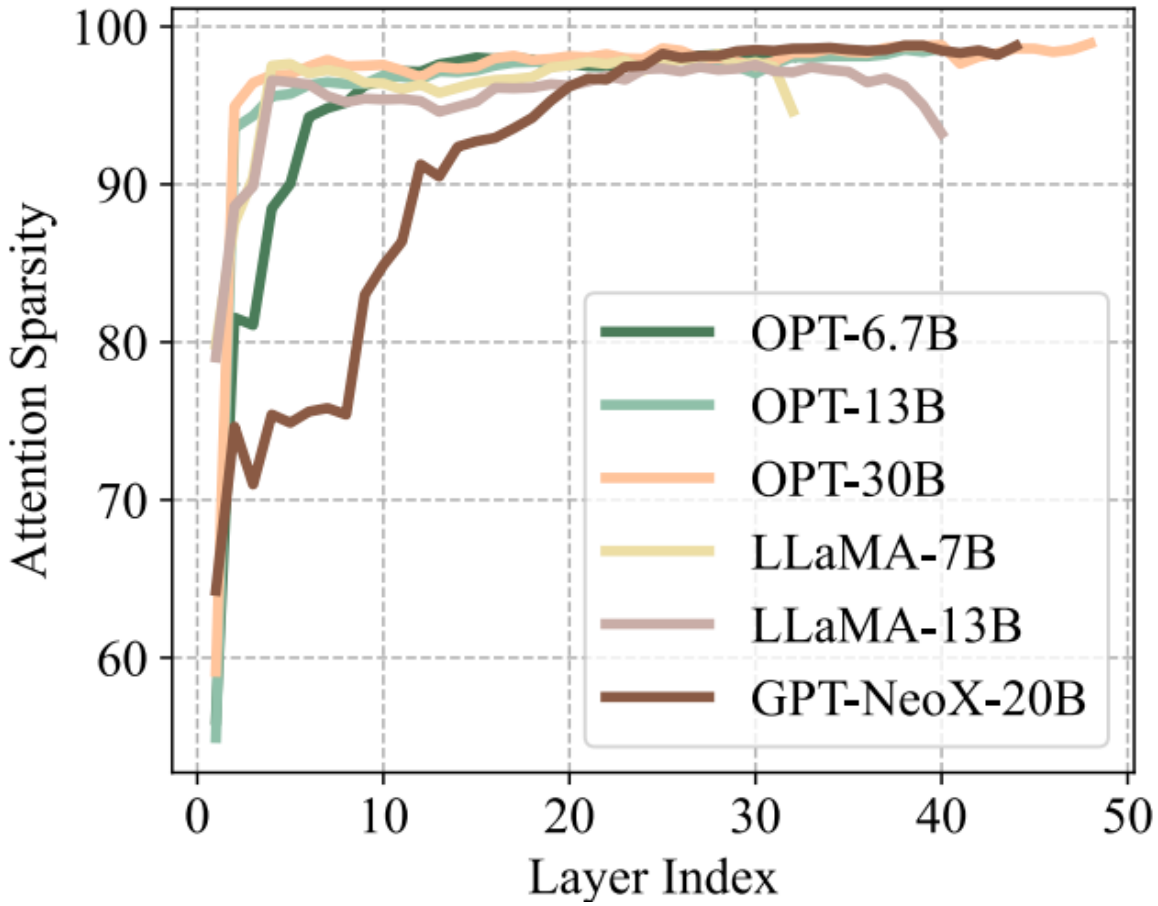
  - Threshold = 1% x Maximum attention score from softmax (QK^T)

- Sparsity over 95% in almost all layers

- Access to all previous key, value is unnecessary for generating the next token

Observation 1

- Sparsity for small cache size



- Attention Sparsity

  - Threshold = 1% x Maximum attention score from softmax (QK^T)

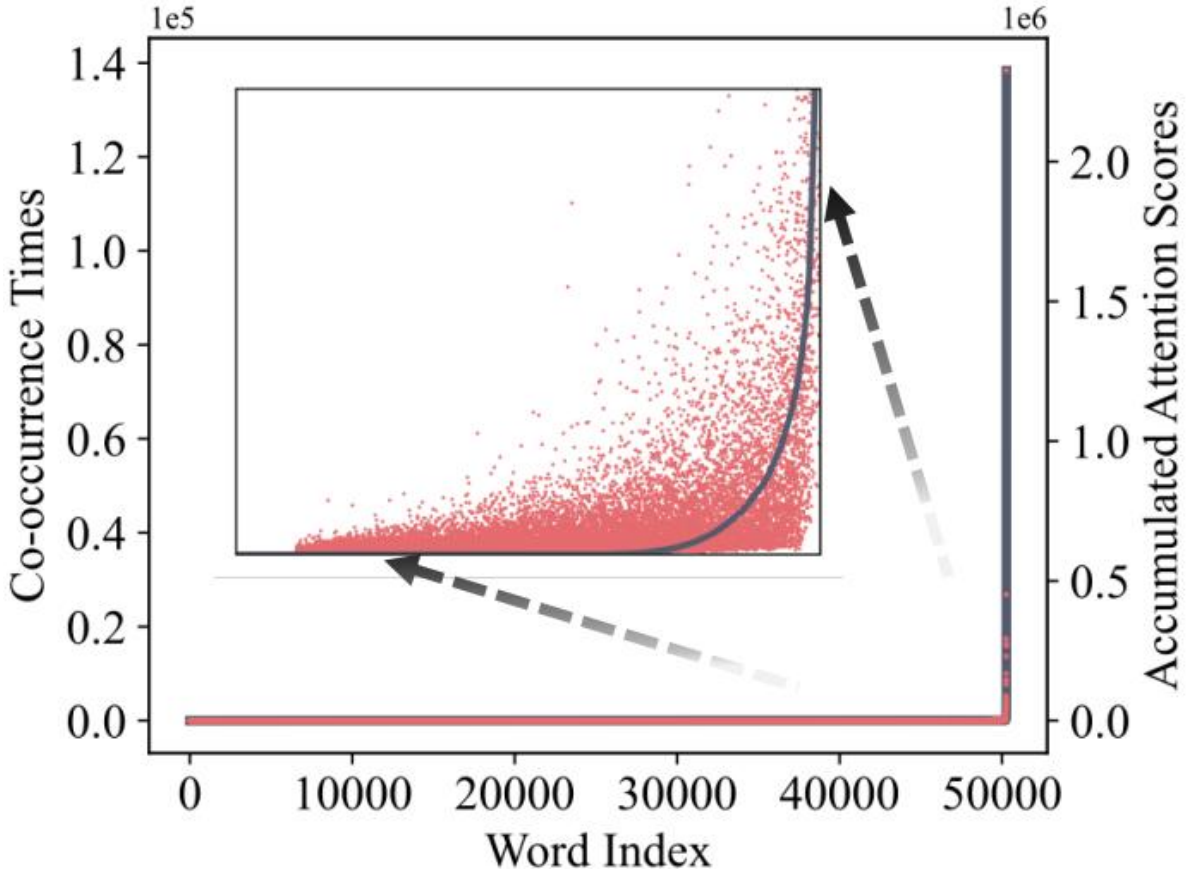
- Sparsity over 95% in almost all layers

- Access to all previous key, value is unnecessary for generating the next token

Takeaway 1: Perhaps a small cache size is possible. But which tokens should stay in cache?

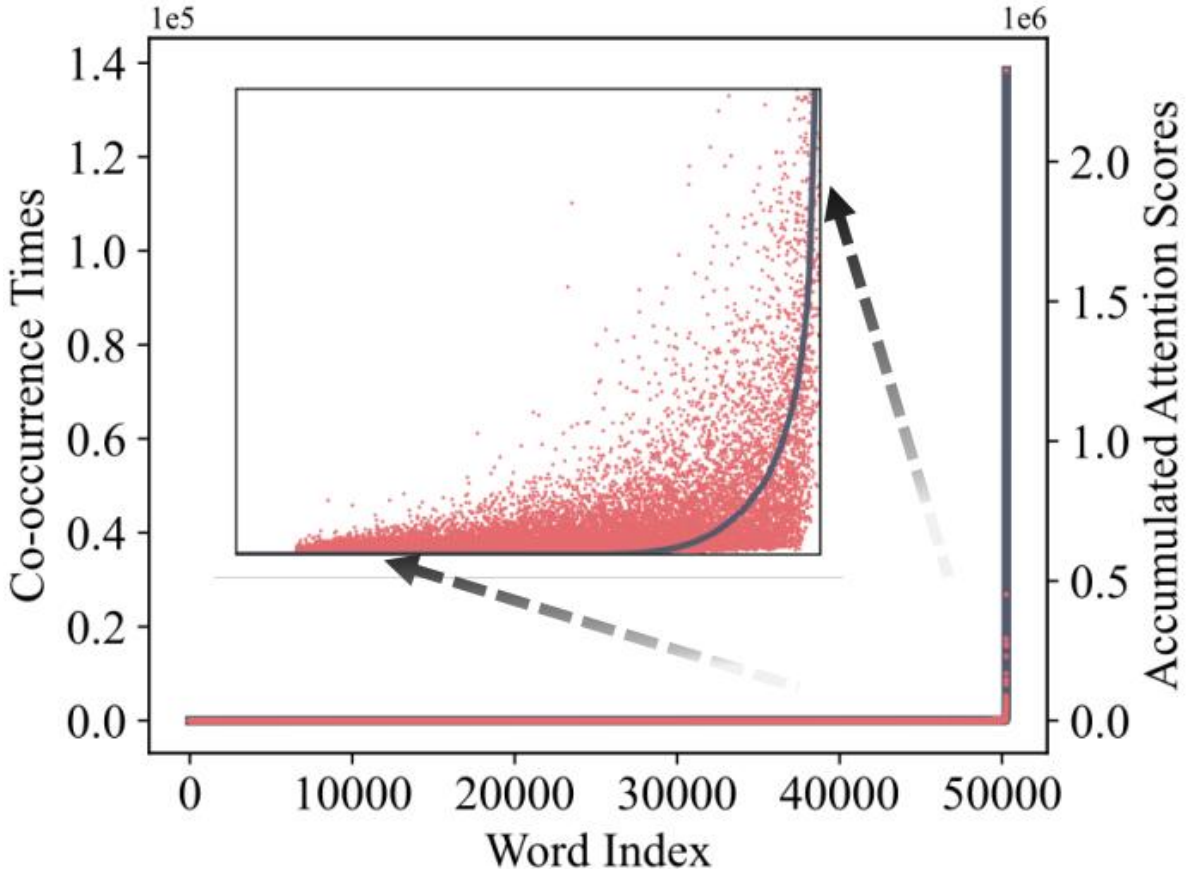
Observation 2

- Heavy-Hitters for Low Miss Rate



Observation 2

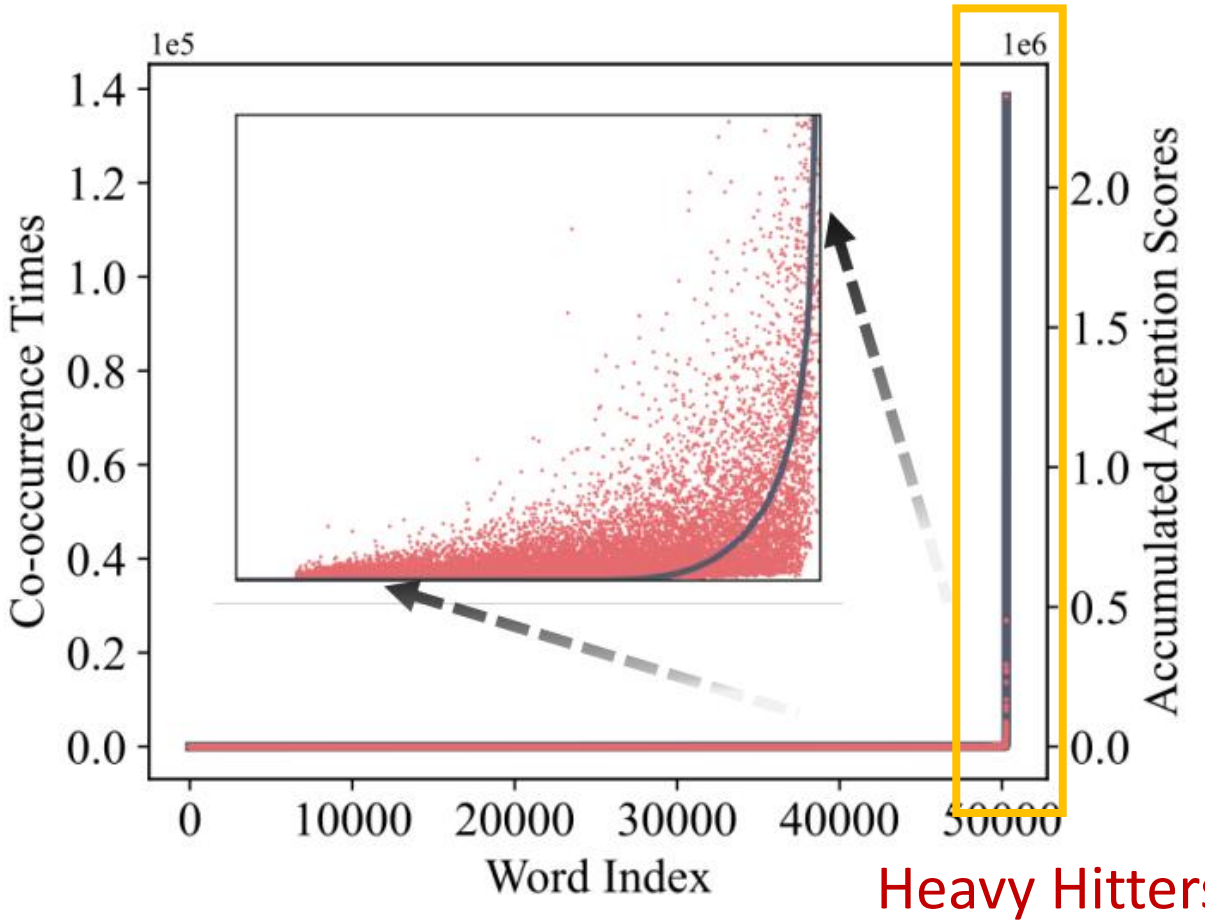
- Heavy-Hitters for Low Miss Rate



- Power-law distribution
  - Accumulated attention score (red)
  - A small set of tokens are critical during generation
  - High correlation with co-occurrences (gray) in the data

Observation 2

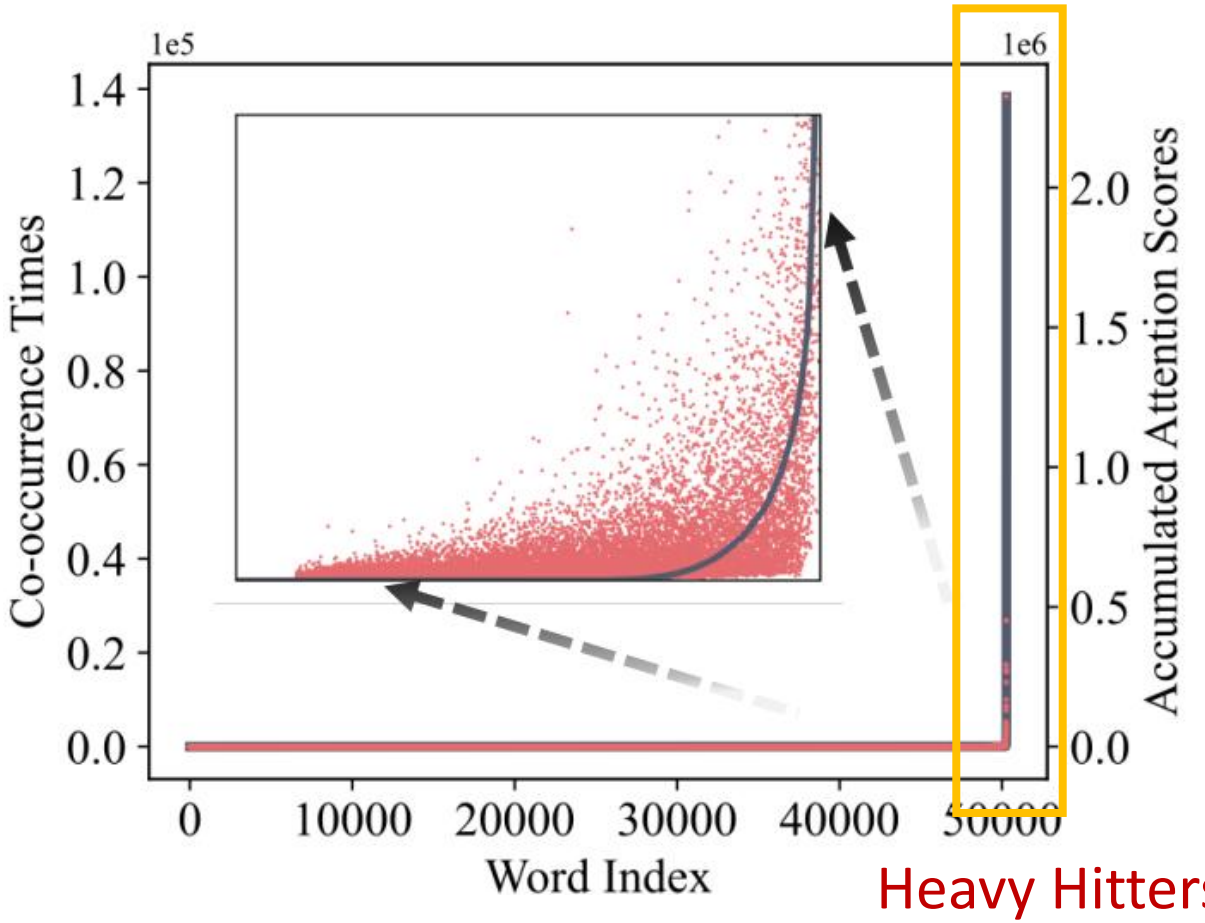
- Heavy-Hitters for Low Miss Rate



- Power-law distribution
  - Accumulated attention score (red)
  - A small set of tokens are critical during generation
  - High correlation with co-occurrences (gray) in the data

Observation 2

- Heavy-Hitters for Low Miss Rate

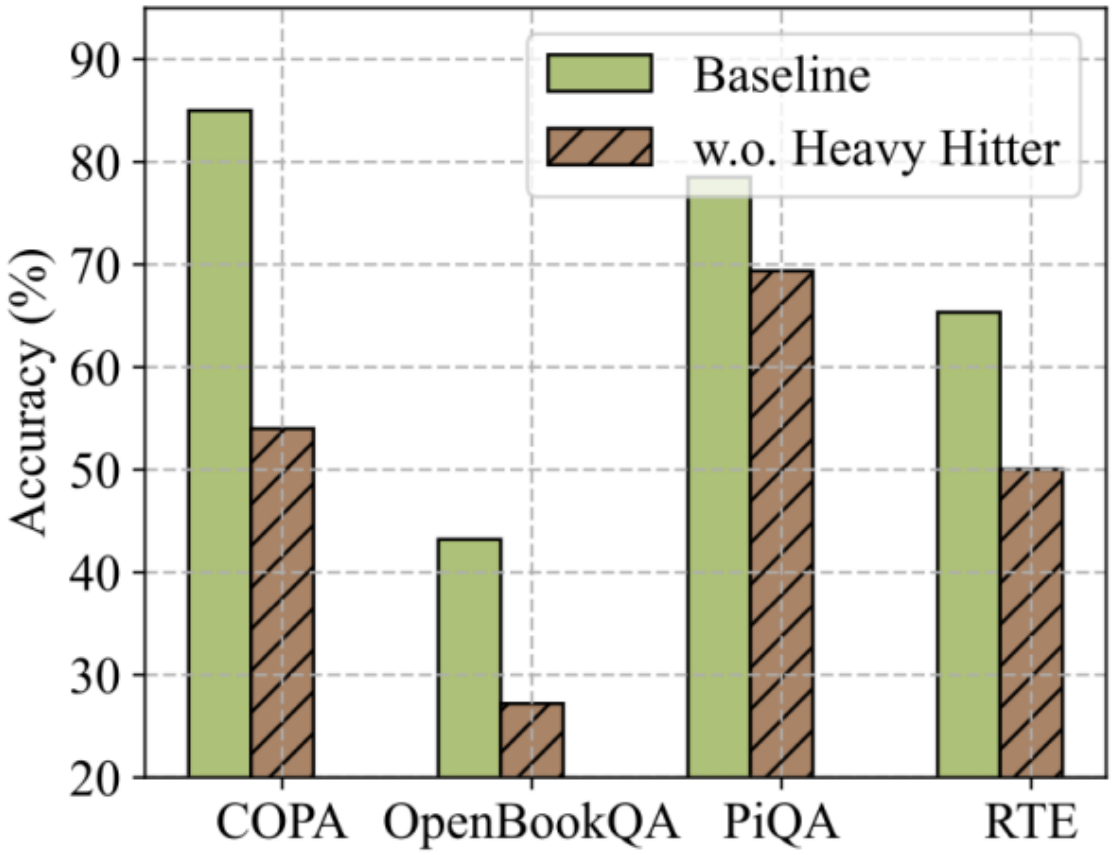


- Power-law distribution
  - Accumulated attention score (red)
  - A small set of tokens are critical during generation
  - High correlation with co-occurrences (gray) in the data

Takeaway 2: Aggregated attention score might be a good signal for determining tokens to retain in KV cache

Observation 3

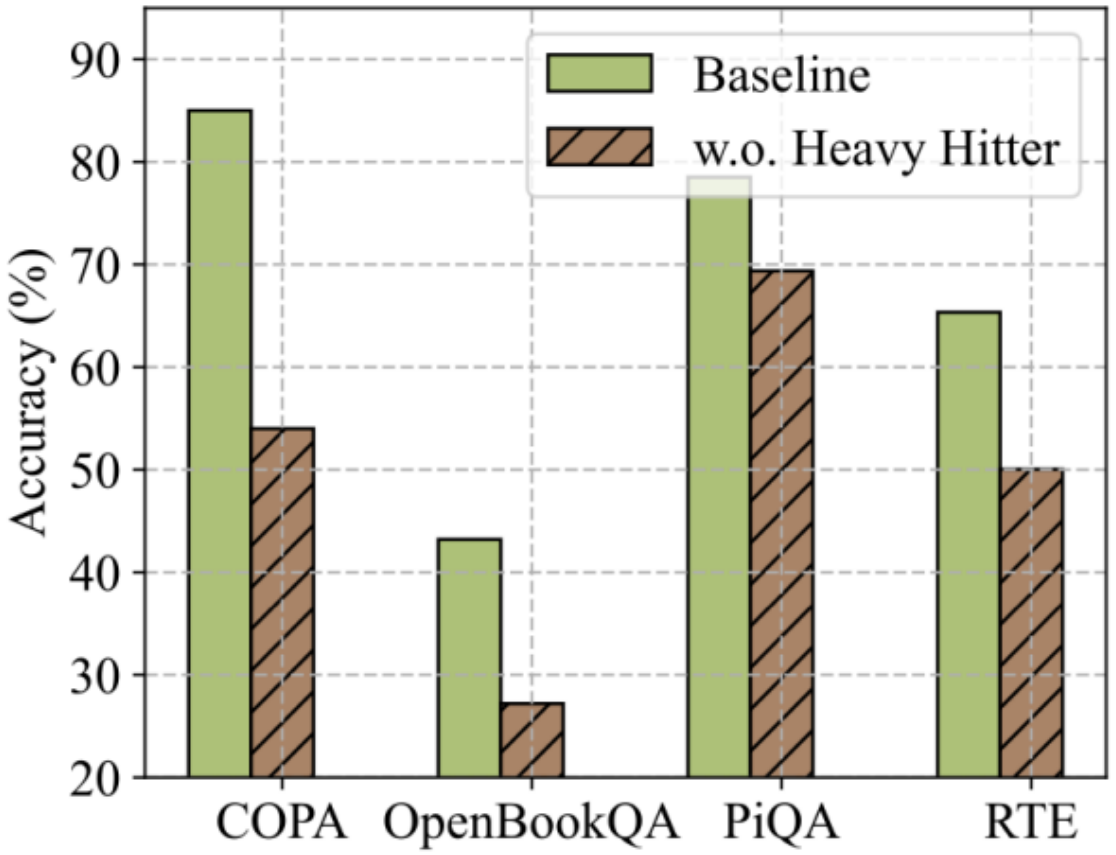
- A small portion of tokens contribute most to attention score



- Evicting Heavy-Hitters destroy the performance of LLMs

Observation 3

- A small portion of tokens contribute most to attention score



- Evicting Heavy-Hitters destroy the performance of LLMs

Takeaway 3: Heavy-hitters are important for LLM inference accuracy

Problem Formulation

- We don’t do know all heavy-hitters before hand

- Because we don’t have the attention scores of the entire sequence before (LLM generation is autoregressive)

Problem Formulation

- We don’t do know all heavy-hitters before hand
- Because we don’t have the attention scores of the entire sequence before (LLM generation is autoregressive)
- Formulate KV cache eviction as a dynamic submodular problem and use a greedy policy for cache eviction

Problem Formulation

- We don’t do know all heavy-hitters before hand
- Because we don’t have the attention scores of the entire sequence before (LLM generation is autoregressive)
- Formulate KV cache eviction as a dynamic submodular problem and use a greedy policy for cache eviction

An optimization problem: the goal is to select a subset of elements (from a large set) that satisfies certain constraints (e.g., size) while maximizing a submodular objective function (e.g., benefit)

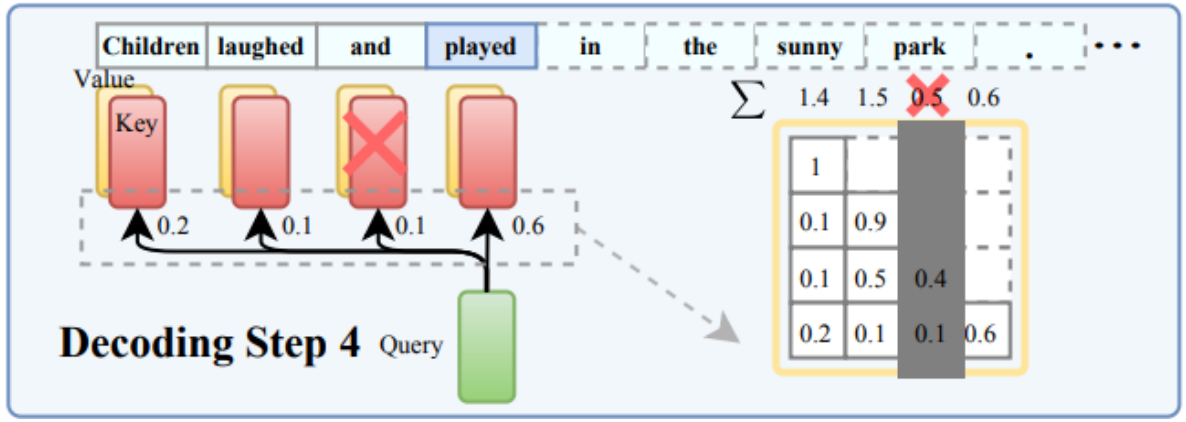
“Submodular Function Maximization”, Andreas Krause and Daniel Golovin

Heavy-Hitter Oracle

- Greedy Algorithm for Low-Cost Policy
  - Keep top-K tokens with highest aggregated attention scores

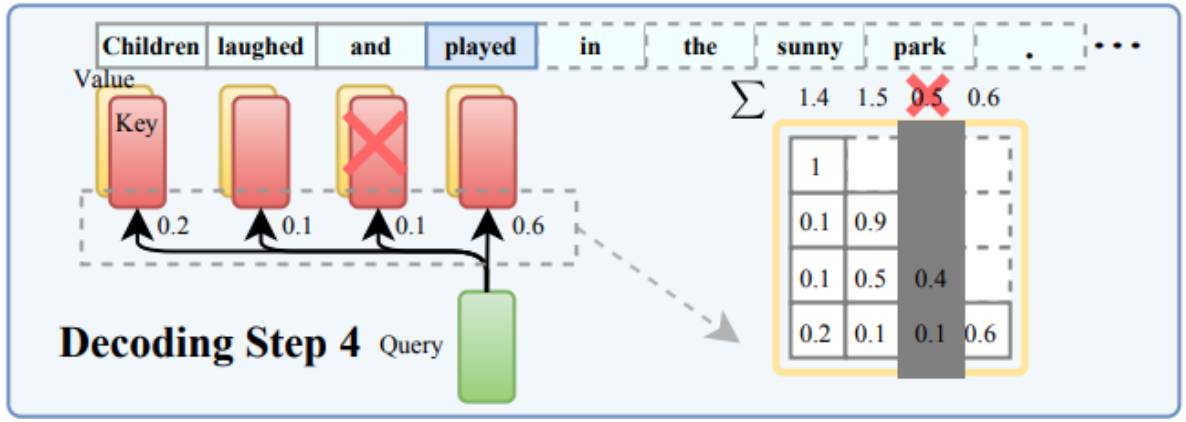
Heavy-Hitter Oracle

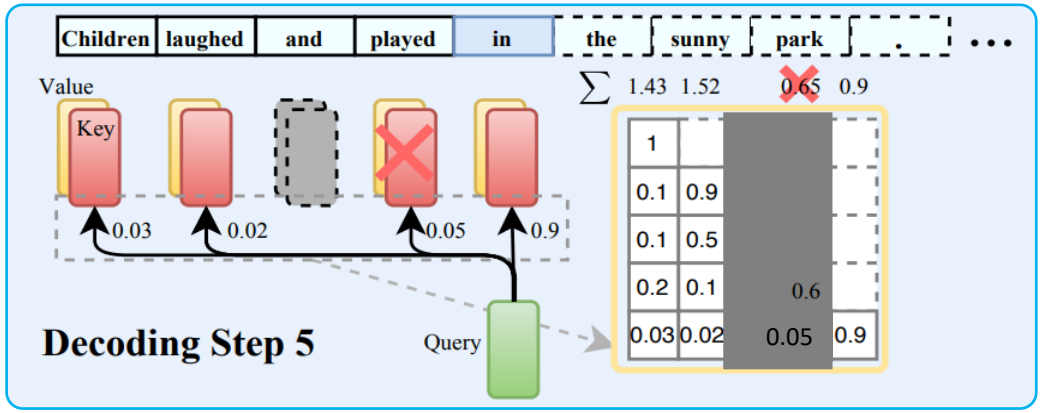
- Greedy Algorithm for Low-Cost Policy
  - Keep top-K tokens with highest aggregated attention scores



Heavy-Hitter Oracle

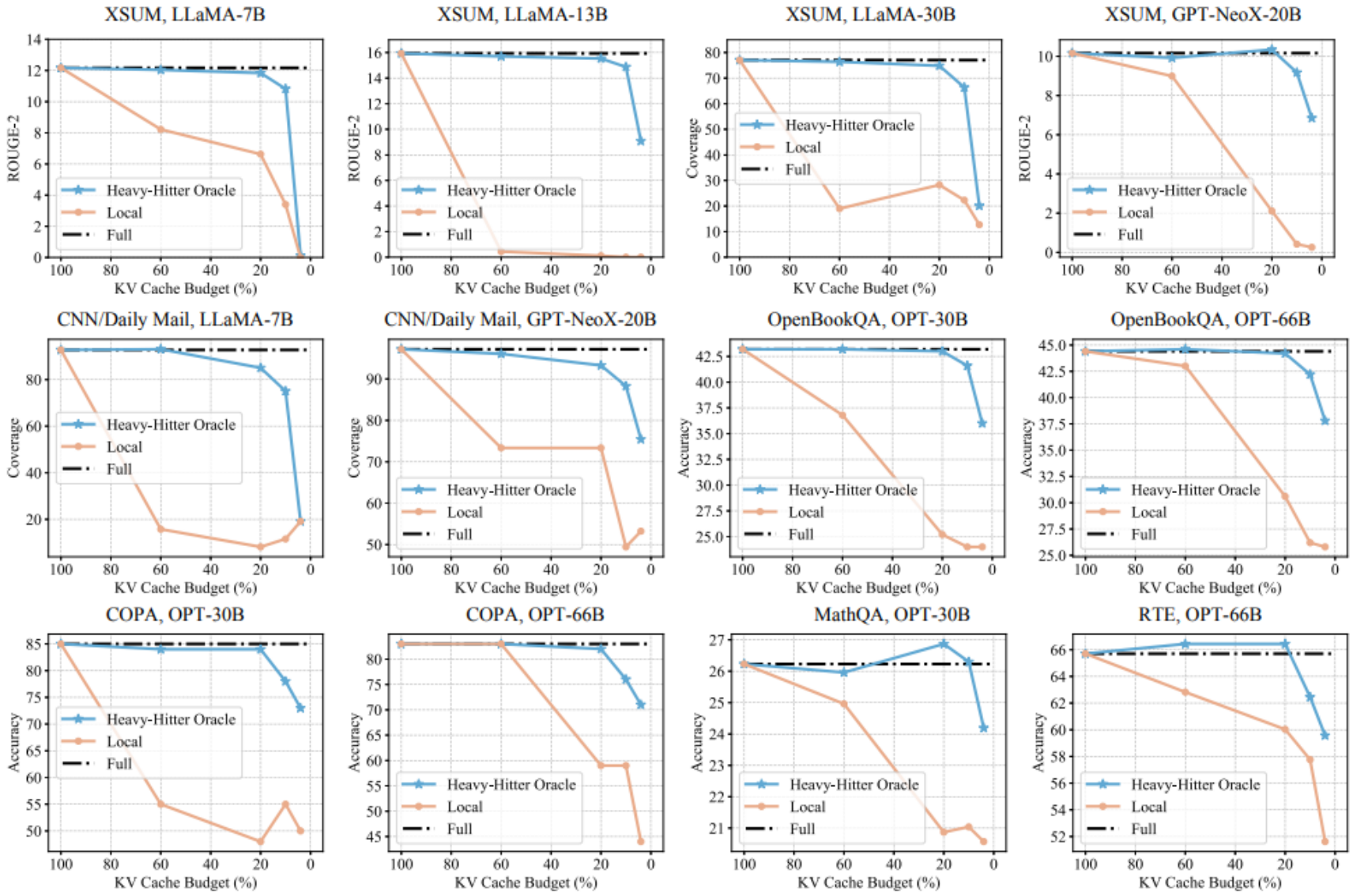
- Greedy Algorithm for Low-Cost Policy
  - Keep top-K tokens with highest aggregated attention scores





KV Cache Reduction

1. Experiments on OPT, LLAMA, GPT-NeoX-20B
2. 8 tasks samples from lm-eval-harness
3. Baseline: full KV cache, “local” window
4. Results
   - Reduce KV cache by 5-10x
   - Comparable Performance



High-Throughput Generative Inference

Seq. length	Model size	Batch size	Metric	FlexGen	H_2O (20%)
7000+1024	30B	1	latency (s)	57.0	50.4
5000+5000	13B	4	latency (s)	214.2	155.4
2048+2048	6.7B	24	latency (s)	99.5	53.5
2048+2048	6.7B	24	throughput (token/s)	494.1	918.9
2048+2048	6.7B	64	throughput (token/s)	OOM	1161.0

Limitations of H2O

Limitations of H2O

- Not compatible with FlashAttention
- Lack support to PagedAttention
- Assume attention score = token importance
- Evicted tokens are irreversible
- Unknown performance on other tasks, e.g., reasoning heavy tasks

Attention Sink

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

Attention Sink

Key observation

- Most of the attention score is held in the first few “sink” tokens
  - Preserve these tokens and reuse the KV cache for last L tokens
  - O(L) computation without much loss in performance

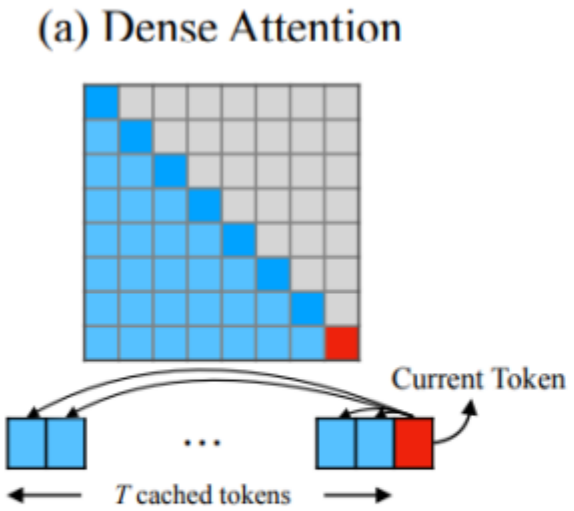
EFFICIENT STREAMING LANGUAGE MODELS WITH ATTENTION SINKS

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

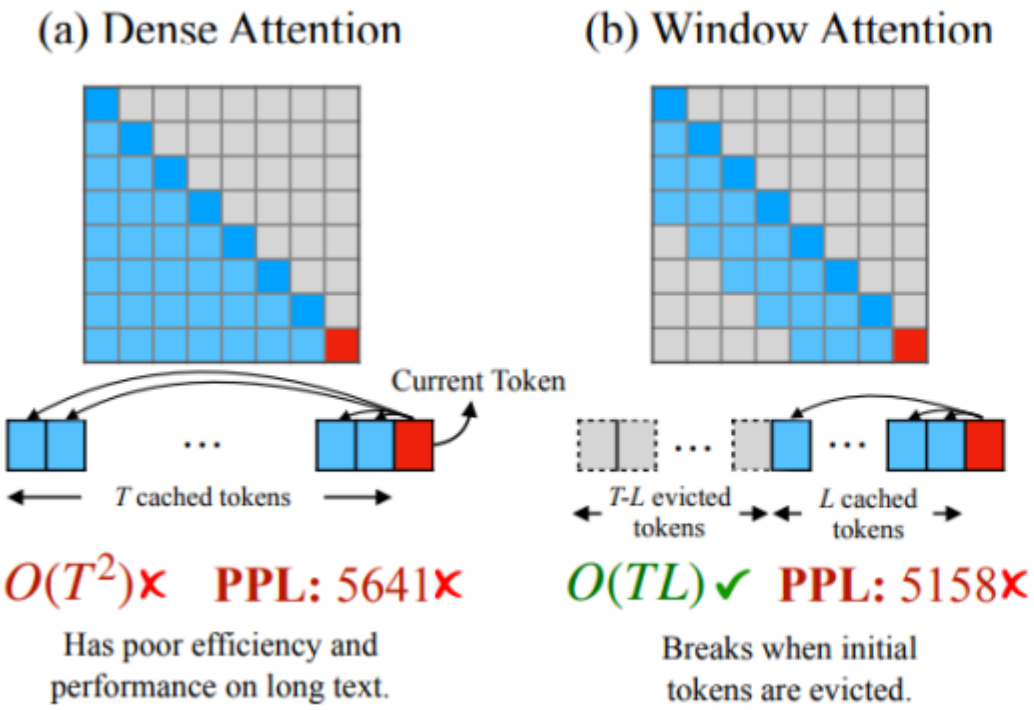
Attention Sink



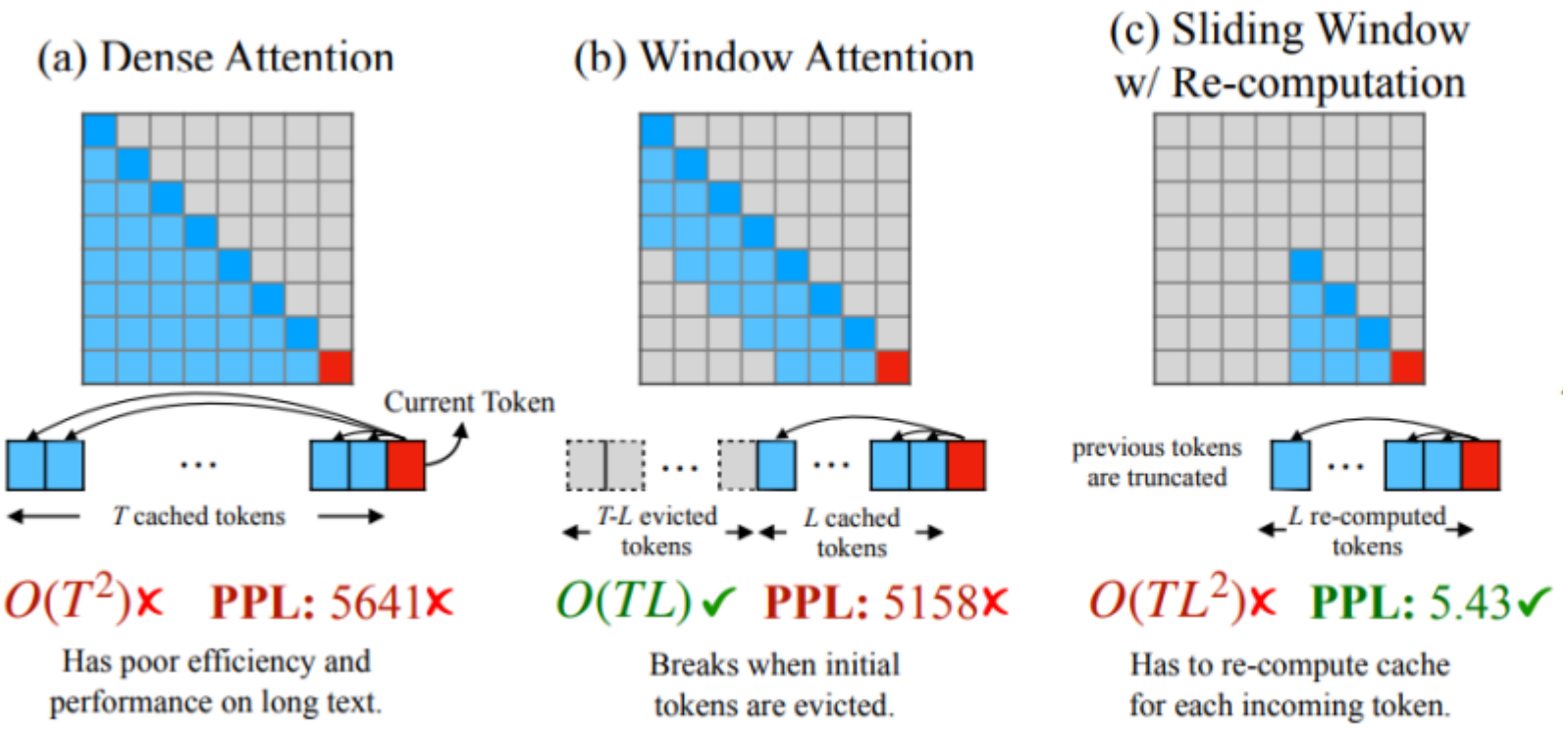
$O(T^2)$✗ **PPL:** 5641✗

Has poor efficiency and performance on long text.

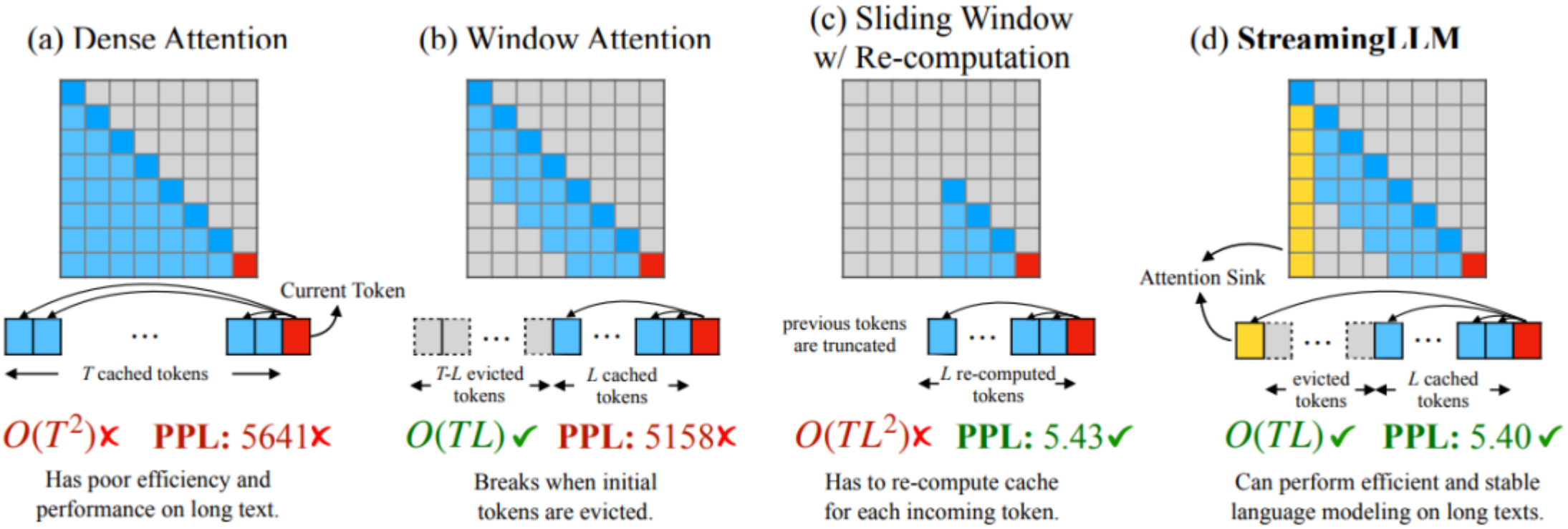
Attention Sink



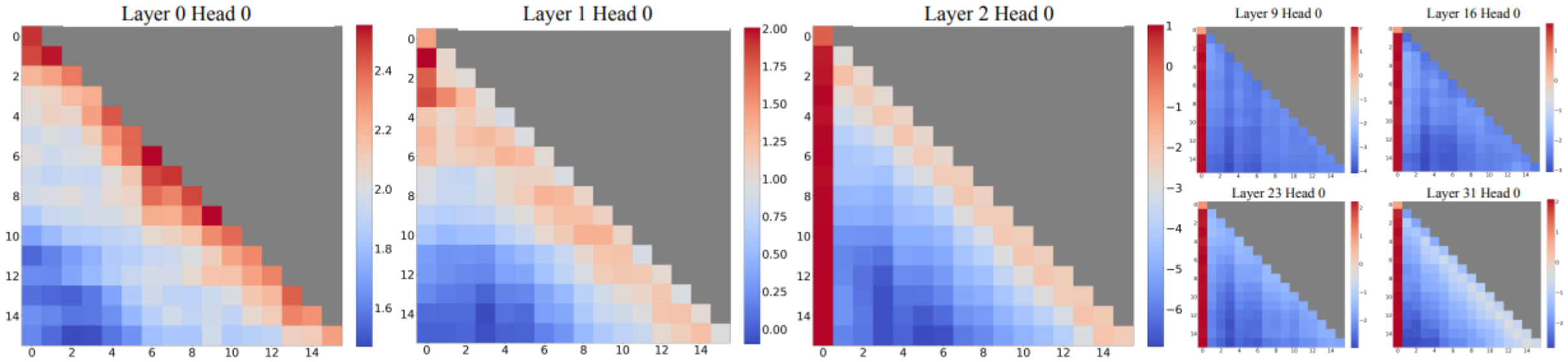
Attention Sink



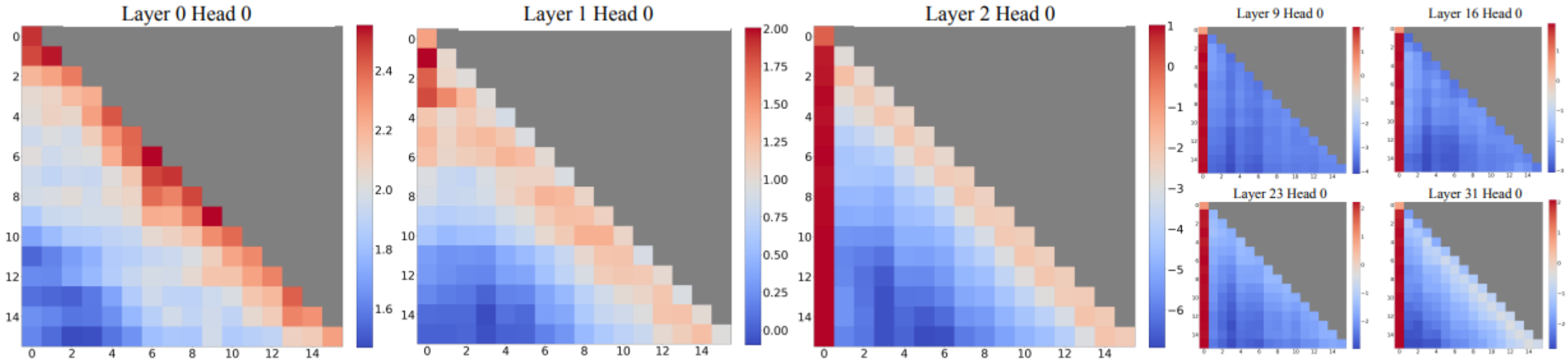
Attention Sink



Empirical Observation of Attention Sink

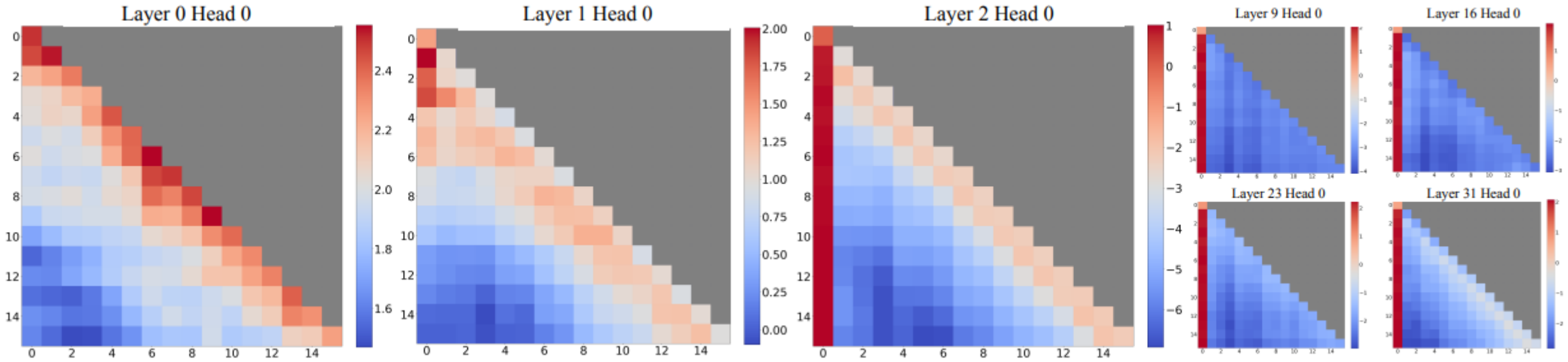


Empirical Observation of Attention Sink



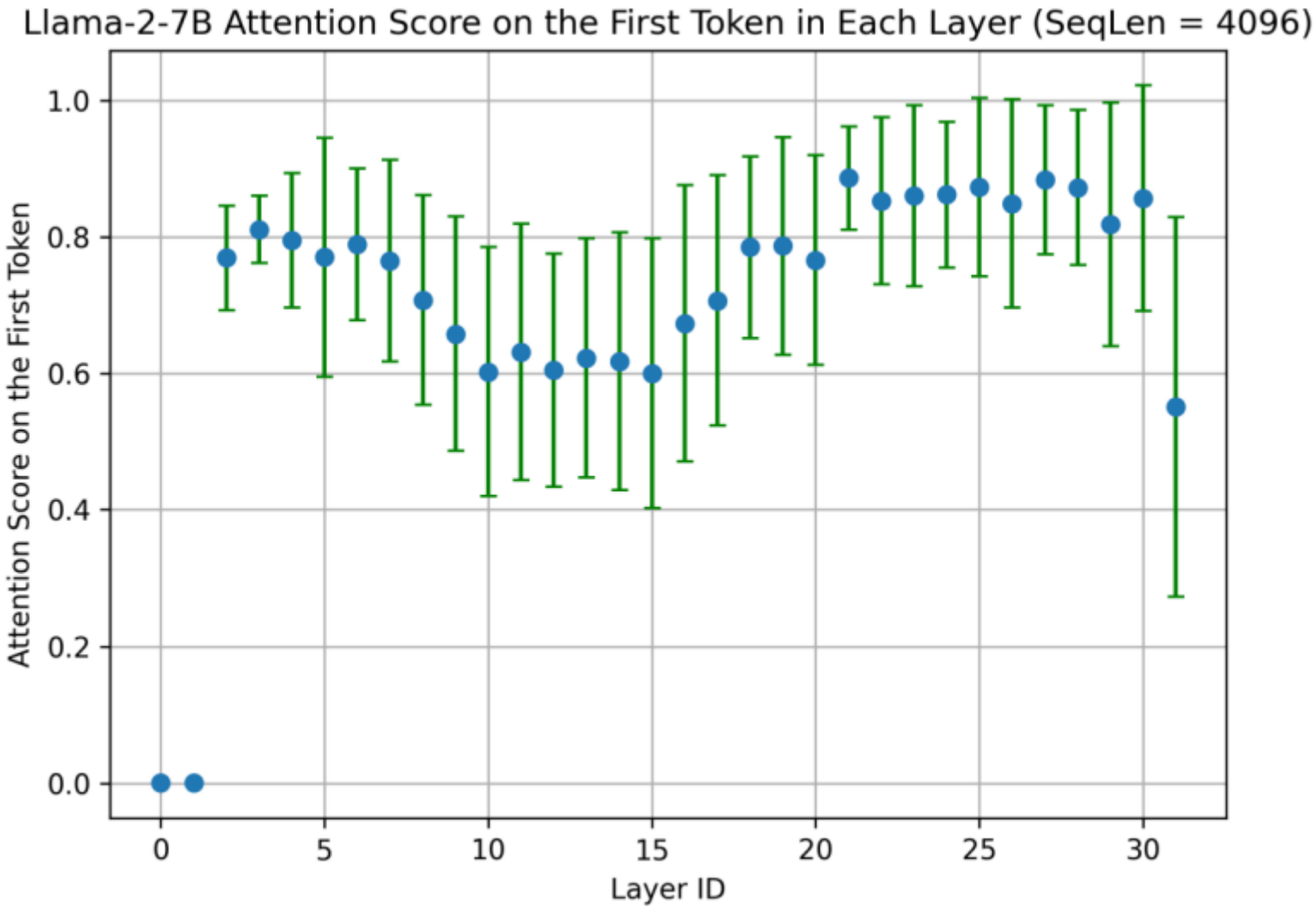
1. Attention maps in the first two layers (layers 0 and 1) exhibit the "local" pattern

Empirical Observation of Attention Sink

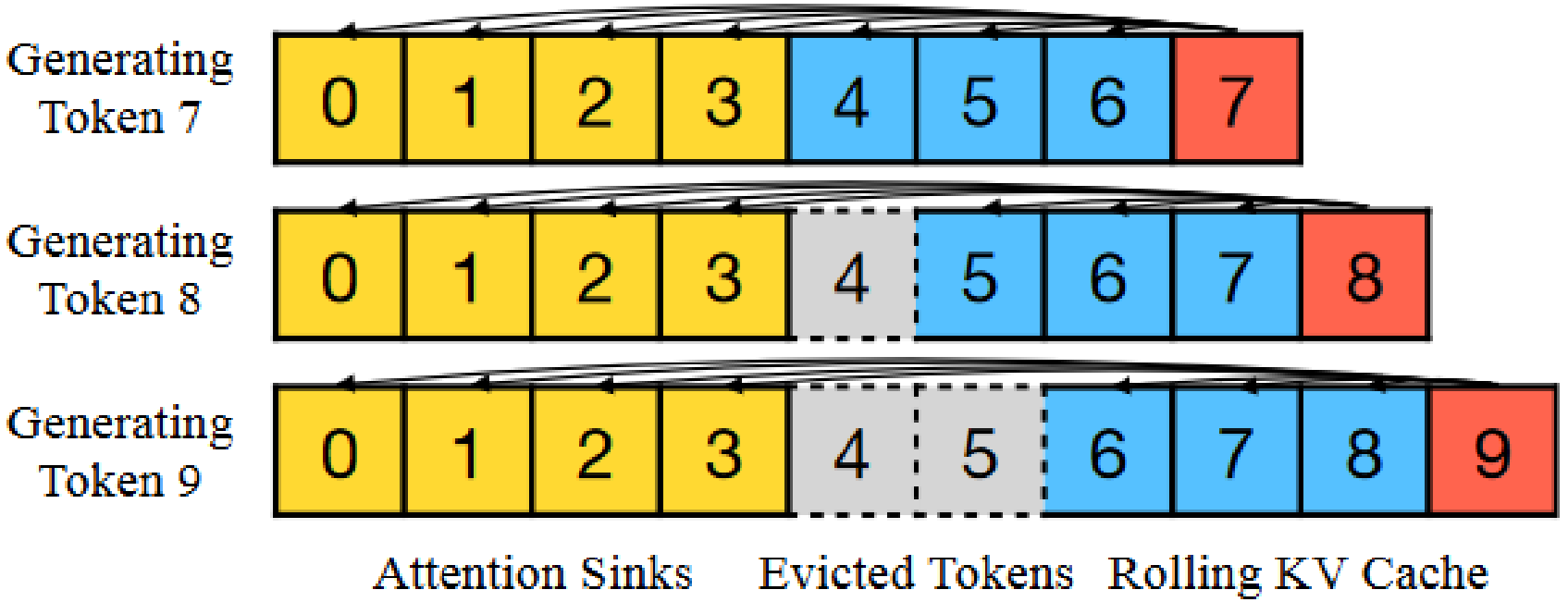


1. Attention maps in the first two layers (layers 0 and 1) exhibit the "local" pattern
2. Beyond the bottom two layers, the model heavily attends to the initial token across all layers and heads

Attention Scores on Longer Sequences

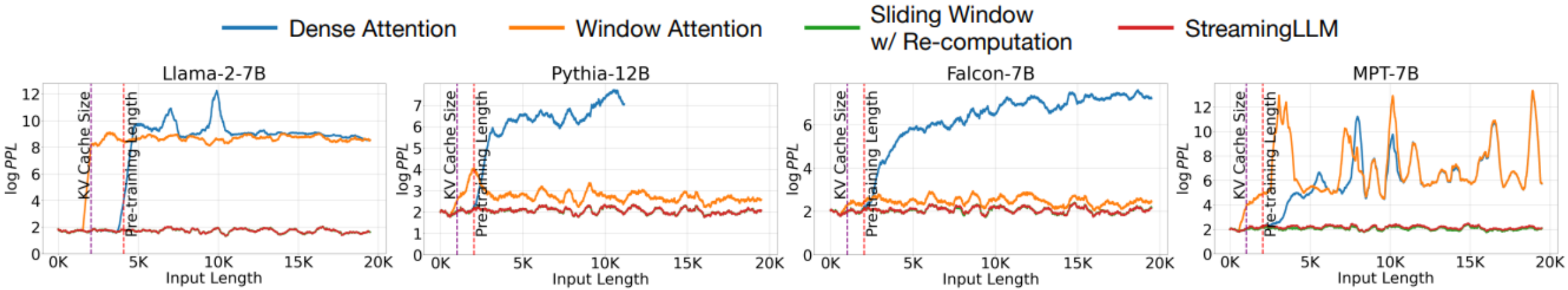


StreamingLLM: Attention Sink + Window Attention



1. Always keep the initial few tokens, i.e., the attention sink tokens
2. Rolling window

Attention Sink – Perplexity Experiments



- Dense attention fails once the input length surpasses the pre-training attention window size
- Window attention collapses once the input length exceeds the cache size
- StreamingLLM demonstrates stable performance

How Many Attention Sinks Do We Need?

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

Which is More Important? Semantic or Position?

Table 1: Window attention has poor performance on long text. The perplexity is restored when we reintroduce the initial four tokens alongside the recent 1020 tokens (4+1020). Substituting the original four initial tokens with linebreak tokens "\n" (4"\n"+1020) achieves comparable perplexity restoration. Cache config x+y denotes adding x initial tokens with y recent tokens. Perplexities are measured on the first book (65K tokens) in the PG19 test set.

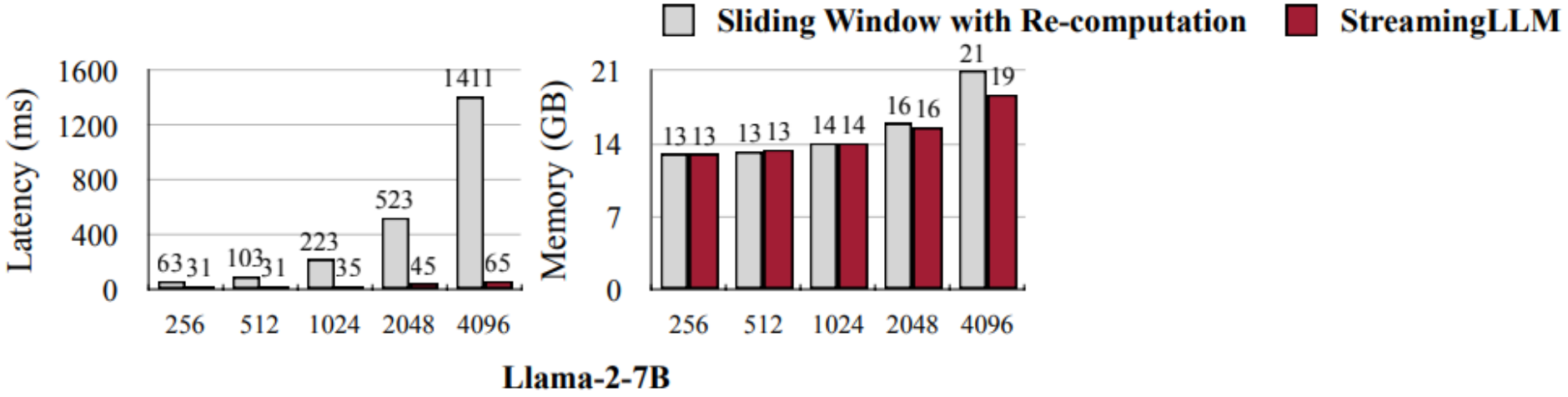
Llama-2-13B	PPL (↓)
0 + 1024 (Window)	5158.07
4 + 1020	5.40
4"\n"+1020	5.60

Which is More Important? Semantic or Position?

Table 1: Window attention has poor performance on long text. The perplexity is restored when we reintroduce the initial four tokens alongside the recent 1020 tokens (4+1020). Substituting the original four initial tokens with linebreak tokens "\n" (4"\n"+1020) achieves comparable perplexity restoration. Cache config x+y denotes adding x initial tokens with y recent tokens. Perplexities are measured on the first book (65K tokens) in the PG19 test set.

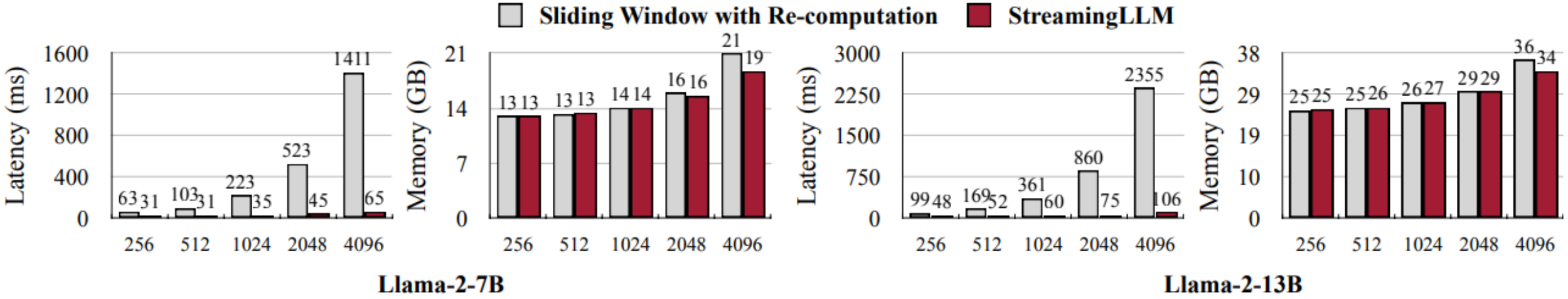
Llama-2-13B	PPL (↓)
0 + 1024 (Window)	5158.07
4 + 1020	5.40
4"\n"+1020	5.60

Efficiency Experiment Results



22.2x speedup per token generation latency while retaining a memory footprint similar to recomputation baseline

Efficiency Experiment Results



22.2x speedup per token generation latency while retaining a memory footprint similar to recomputation baseline

Limitations

Limitations

- Hard to interpret
- Limited evaluation scope
- May underperform on out-of-distribution inputs

Questions?