Extending Context-Awareness in StreamingLLM with Retrieval-Augmented Generation

Miguel S. Moreira [1], Luc Dao [2], Jaehyun Sim [2], Christopher McNally [1]

[1] Department of Electrical Engineering & Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, USA.
[2] System Design & Management, Massachusetts Institute of Technology, Cambridge, MA 02139, USA.

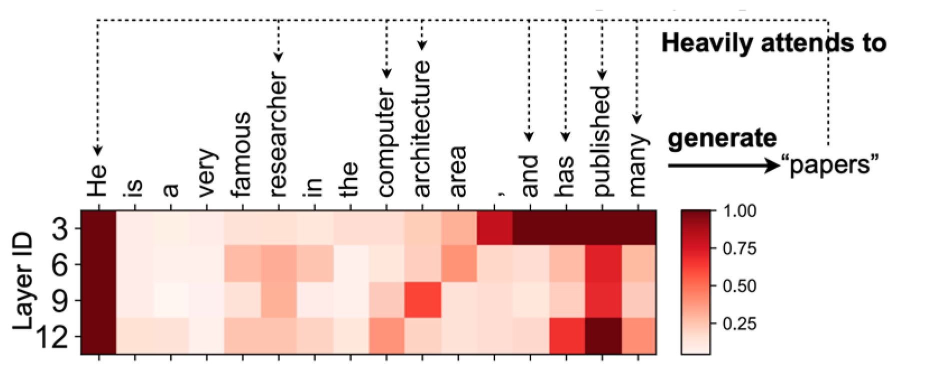
Hardware, AI and Neural-nets

StreamingLLM and Attention Sinks [1]

The "Attention Sink" Phenomenon

Tokens that disproportionately attract attention, irrespective of the context or semantic importance.

- Naturally emerge from softmax, which forces attention to sum to one, favoring initial tokens due to autoregressive nature.
- Can be explicitly added, allowing model to learn their use as implicit memories for (globally relevant) context

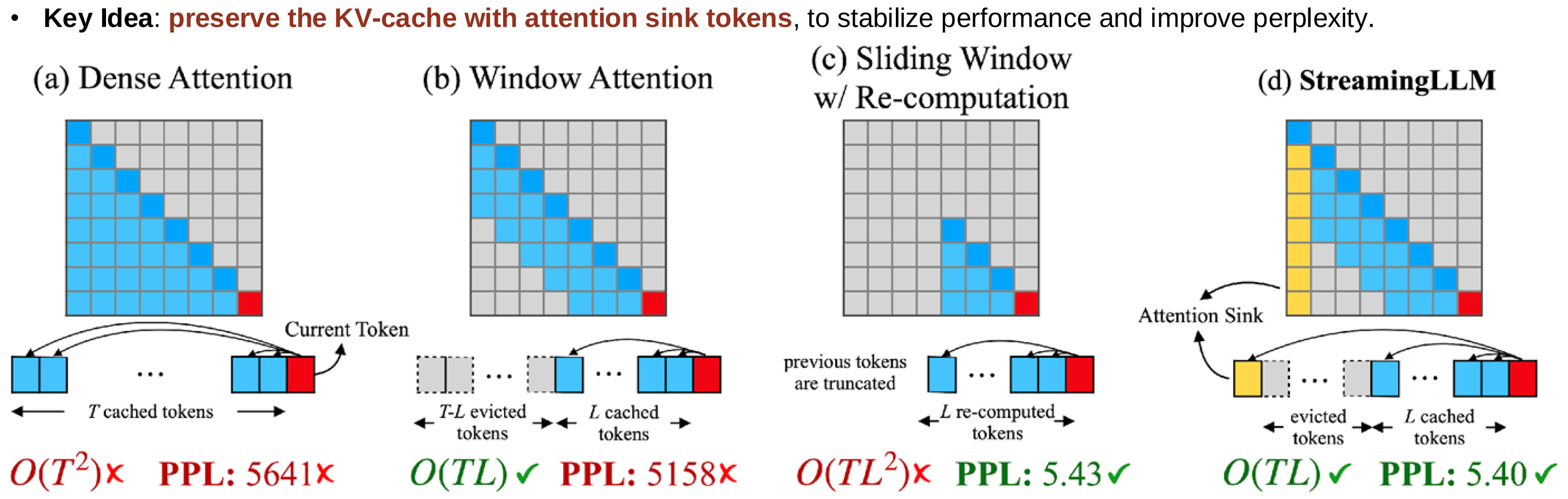


Llama-2-13B	PPL (↓)
0 + 1024 (Window)	5158.07
4 + 1020	5.40
4"\n"+1020	5.60

The importance of initial tokens arise from the autoregressive nature of the model, since first tokens are allowed to interact with every other token (**global aggregation via attention**).

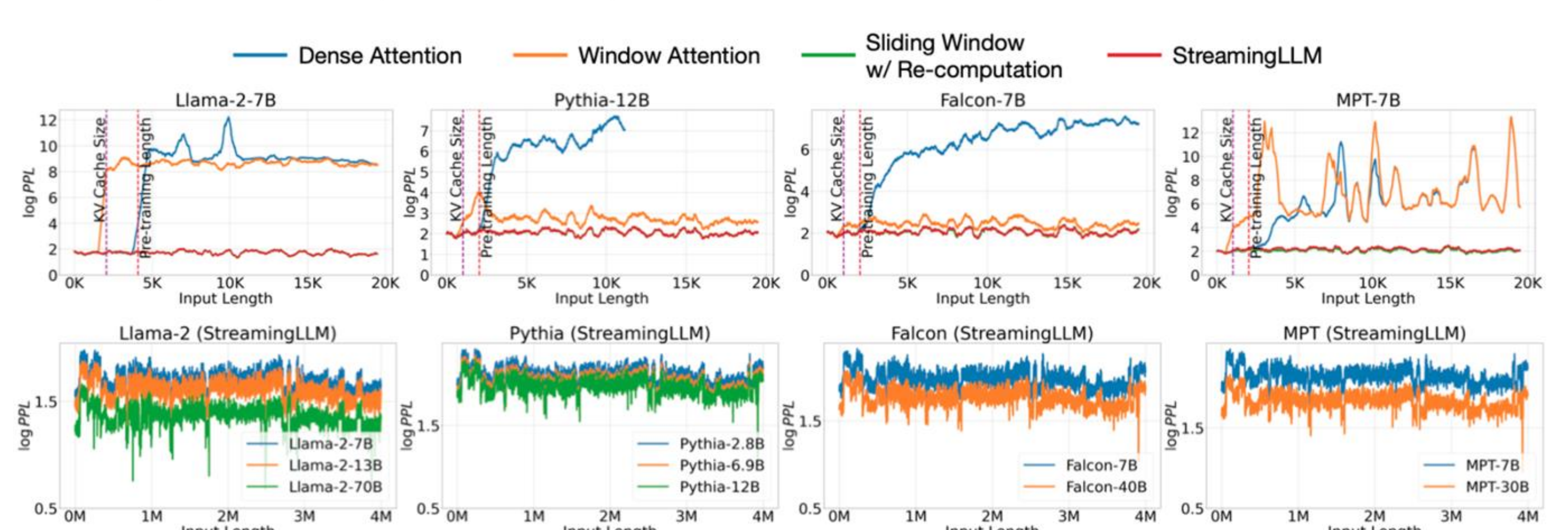
StreamingLLM: Using Attention Sinks for Infinite Streams

- **Objective**: Enable LLMs trained with a finite attention window to handle infinite text inputs without additional training.
- **Key Idea**: **preserve the KV-cache with attention sink tokens**, to stabilize performance and improve perplexity.



StreamingLLM shows **stable performance**; perplexity close to sliding window with re-computation baseline.

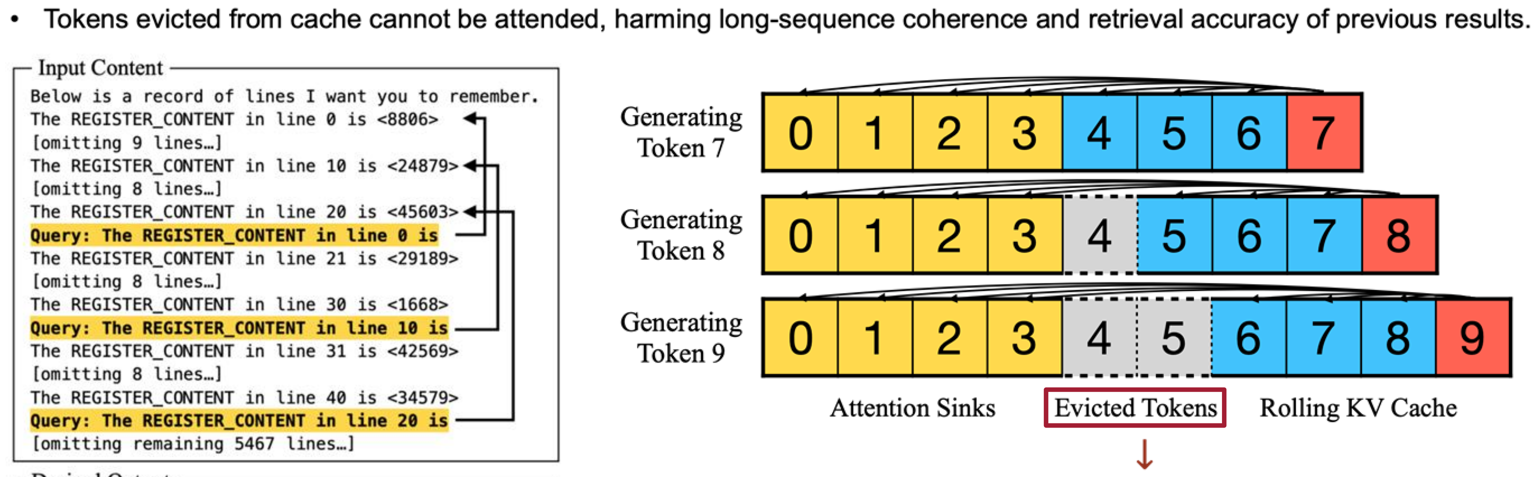
- Dense attention fails beyond pre-training attention window size.
- Window attention fails after input exceeds cache size (initial tokens evicted).
- Sliding window with re-computation incurs significant overhead (long sequences) due to re-computation for incoming tokens.
- StreamingLLM has perplexity close to the sliding window with re-computation baseline but provides up to **22.x speedup**.



Challenge of StreamingLLM

StreamingLLM **does not** give us infinite context

- Non-stop chatting ≠ Infinite context
- Tokens evicted from cache cannot be attended, harming long-sequence coherence and retrieval accuracy of previous results.

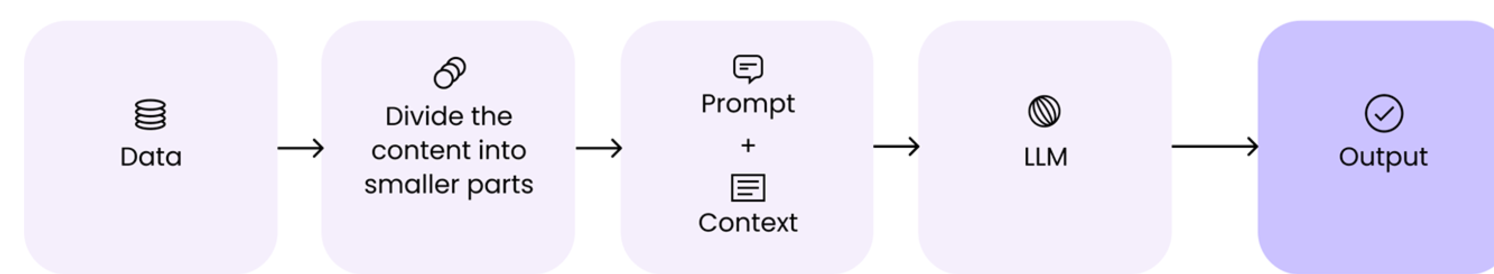


Tokens evicted from KV cache **cannot be attended to** anymore.

Retrieval Augmented Generation

Retrieval Augmented Generation (RAG) enhances language models by integrating an external retriever to dynamically fetch and embed relevant information during inference.

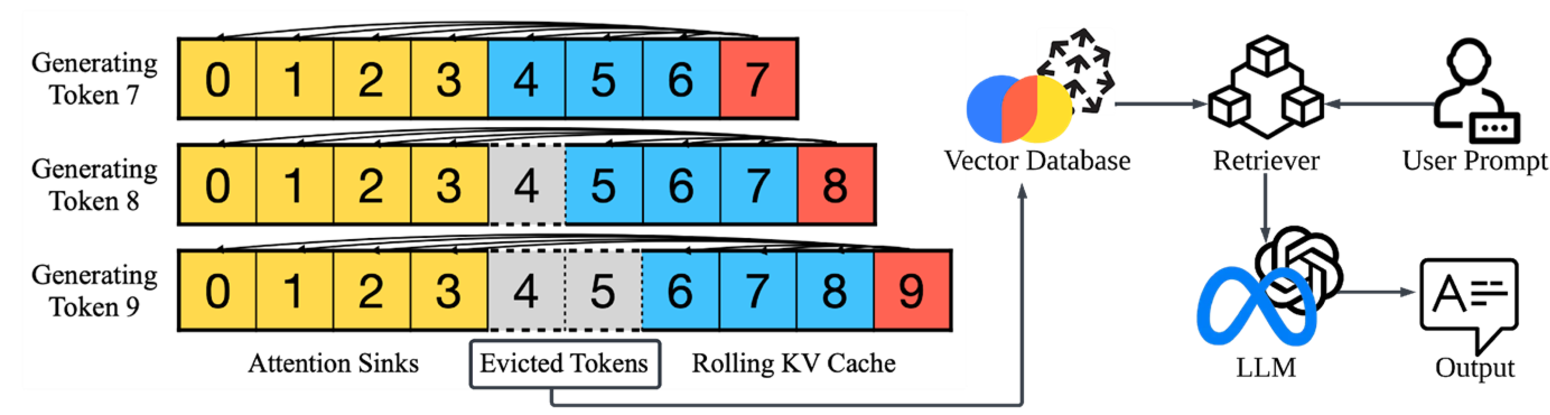
- **Objective**: Extend the effective context length with no architectural changes and minimum overhead during inference.
- **Key Idea**: The retriever identifies past information that is contextually relevant, information which is then fused with the prompt provided as an input to the model, allowing it to better inform the next token prediction.



Extending Context Awareness in StreamingLLM with RAG

StreamingLLM with RAG

- Tokens evicted from StreamingLLM are stored in an **external vector database** as embeddings.
- Evicted information can be retrieved on demand, **augmenting the current input prompt**.
- This approach **avoids architectural changes** by leveraging retrieval as an extension to existing inference workflows.



Importance of Explicit Retrieval for Long Context

- Attention Sinks in StreamingLLM should help capture **semantic** relationships, ensuring broad thematic alignment.
- However, **finite memory** should lead to **capacity saturation**, where **details in evicted context are permanently lost**, leading to degradation over long-context sequences and discrepancies in precise details.
- RAG enables the integration of external databases with vastly greater scale to **augment the effective context length** of a model, allowing greater coherence over extended sequences and **accuracy in precise details** from prior interactions.

Implementation

Implementation Logic

```
# Load modules
DEFINE LLM_MODEL, TOKENIZER, RETRIEVER

# Initialize KV Cache
kv_cache = KVCacheWithStreamingLLM()

# Process each prompt
for each prompt:
    # Merge the prompt with context query result
    new_prompt = RETRIEVER.query(prompt) + prompt

    # Generate texts with LLM Model
    output = LLM_MODEL.generate(new_prompt)

    # Obtain evicted context from KV Cache
    evicted_context = kv_cache.evicted()

    # Store new output and evicted context
    RETRIEVER.store(output)
    RETRIEVER.store(evicted_context)

    # Return the generated text
    return output
```

Setup for StreamingLLM

Model: meta-llama/Llama-2-7b-chat-hf

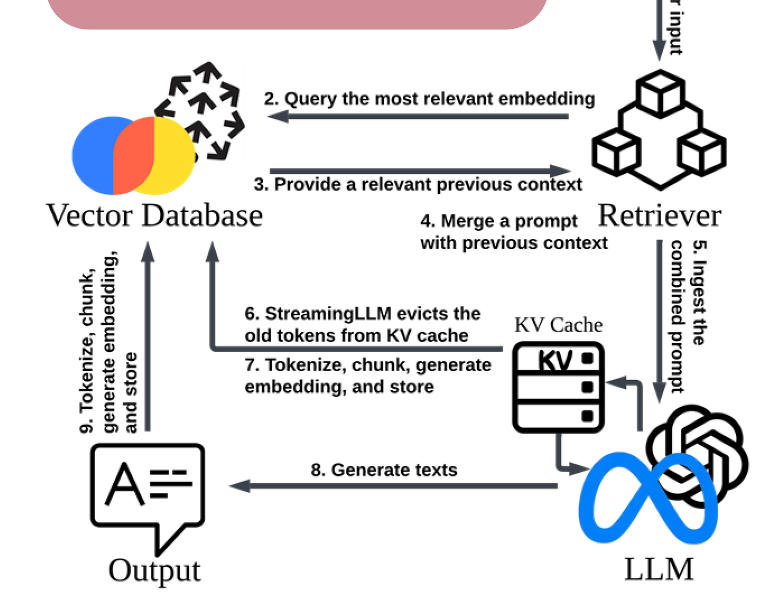
Start Size: 4 tokens

Recent Size: 4096 tokens

Workflow

Setup for RAG

- Vector Database: ChromaDB
- Embedding Model: all-MiniLM-L6-v2
- Embedding Space: 384 dimensions
- Chunk Size: 100 tokens



Results and Benchmarks

Benchmarks Considerations

Accuracy Functions and Metrics

	Cosine Similarity	Direct Keyword Matching
Definition	Measures the cosine of the angle between two vectors in embedding space, capturing overall **semantic** similarity.	Focuses on explicit overlap of key terms or concepts, **prioritizing direct matches** as a metric for accuracy.
Strengths	- Easy to implement and computationally efficient. - Effectively measures thematic or semantic connections.	- Captures direct relevance to the query for retrieval tasks. - Ensures explicit alignment with **specific key terms** or concepts in the input.
Weaknesses	Can rank irrelevant results highly due to loose semantic ties, even if they lack key query information.	May fail to capture nuanced or broader semantic relationships.

Given the focus on precise retrieval implicit in RAG techniques, we ascertained that **Direct Keyword Matching** was the most faithful metric with which to determine accuracy in our model, but we have complemented it with cosine similarity below.

Context Augmentation Strategy

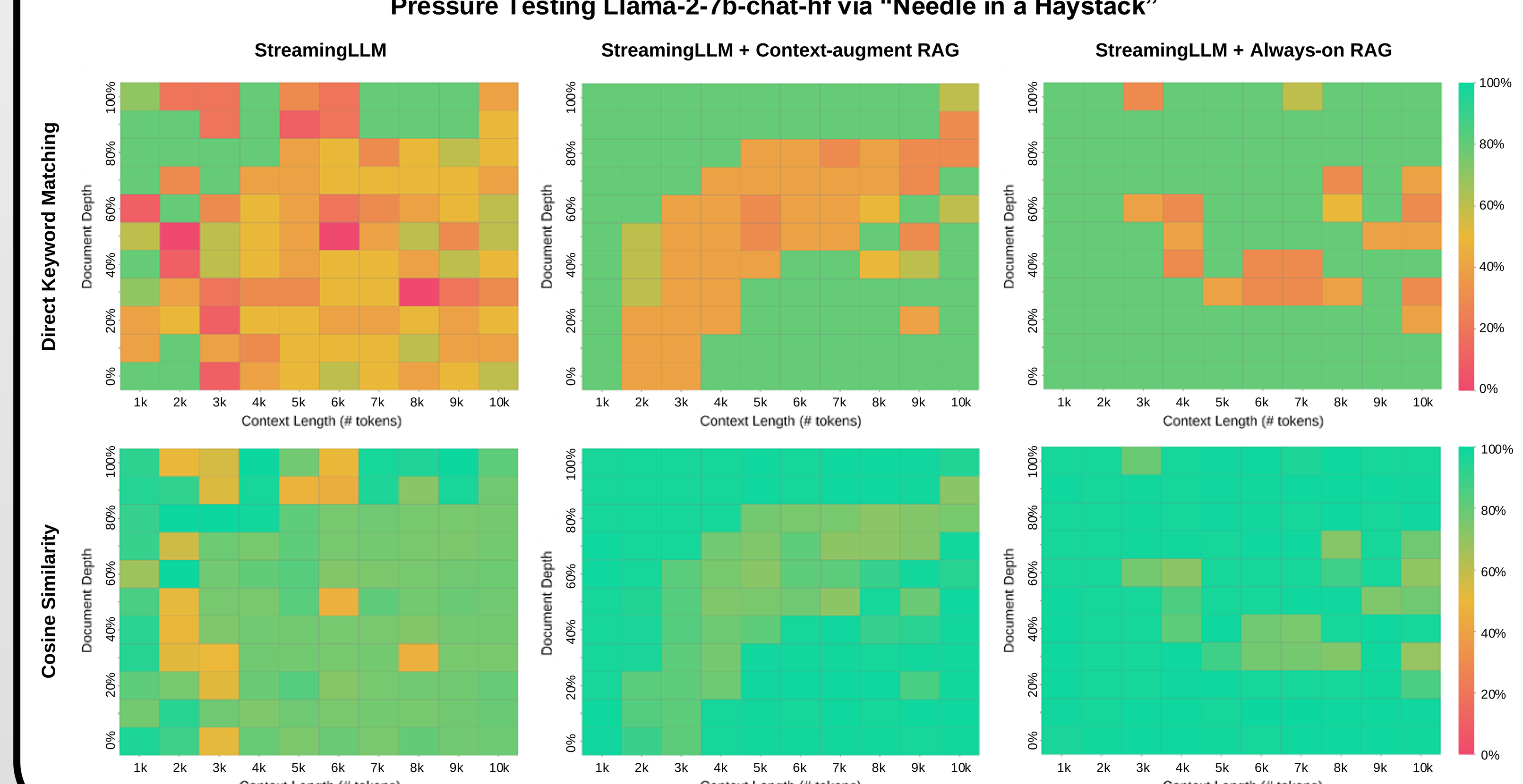
	Always-on RAG	Context-augmented RAG
Definition	Continuously stores all prompts into the vector database, regardless of their presence in the KV cache.	Stores prompts in the vector database only when they are evicted from the KV cache.
Strengths	- Ensures all prompts are available in the vector database, reducing risk of missing relevant data. - Simpler implementation as it doesn't require cache monitoring.	- Optimizes storage and retrieval by avoiding redundancy with KV cache. - Reduces vector database size, improving retrieval efficiency and latency.
Weaknesses	Can lead to storage bloat due to redundant data being stored or potentially unnecessary inference overhead.	Requires careful cache eviction monitoring to ensure no relevant prompts are lost.

We choose **context-augmented RAG** to ensures that only essential prompts are stored, minimizing duplication while reducing latency during retrieval, but have tested both, for completeness.

Needle In A Haystack [2]

- Evaluates LLMs' ability to accurately retrieve specific embedded information from large text contexts.
- Highlights potential weaknesses in models' long-context processing.

Pressure Testing Llama-2-7b-chat-hf via "Needle in a Haystack"



Conclusion

Combining StreamingLLM with RAG

- Enhanced the model's **context-awareness** and **retrieval capabilities** using Retrieval-Augmented Generation (RAG).
- Implementation supports **additional performance boosts** by integrating large databases of specialized knowledge.
- Within the broader context of deep learning theory, expanding context lengths is **key to bridging the training-inference gap**, enabling enhanced **in-context learning capabilities**.

RAG offers a scalable solution to overcome hardware memory limitations and extend model capabilities.

Contributions

Retriever with Vector DB was implemented by LD with support from CM. Model integration and testing was done by MSM with support from JS. Benchmarking was done by MSM and CM. Presentation materials were prepared by JS and MSM.

References

[1] Xiao, Guangxuan Xiao, et al. "Efficient streaming models with attention sinks." *arXiv:2309.17453* (2023).
[2] Gregory Kamradt. "LLMTEST Needle In A Haystack." *github.com/gkamradt/LLMTest_NeedleInAHaystack* (2023).