Remembering LLM

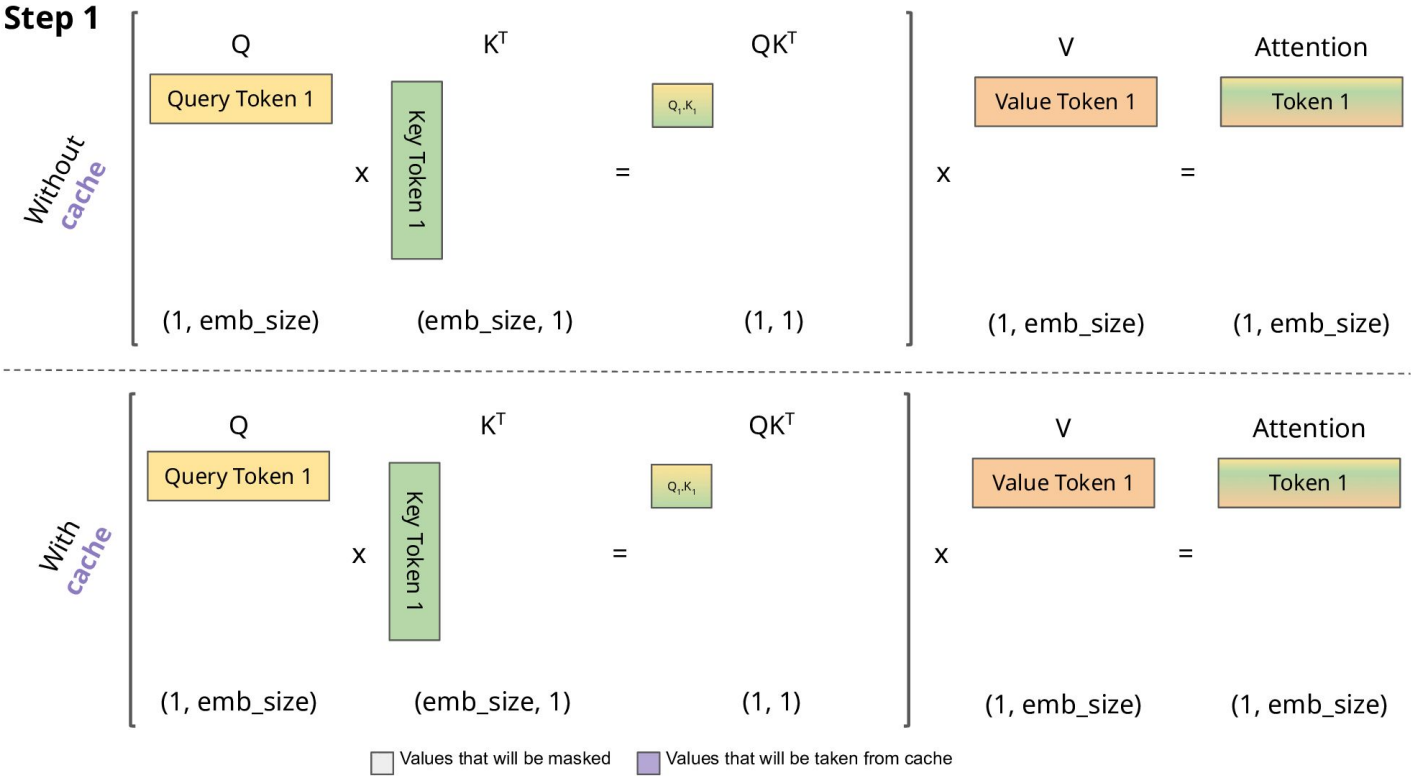
Kevin Tong, Tony Cui, Anish Ravichandran

Agenda

1. Introduction to StreamingLLM
2. Methodology
    a. K-Means clustering (K-means++)
    b. Capitalizing on the attention sink
    c. Maximal Distance Retrieval Augmented Generation
3. Evaluation & Datasets

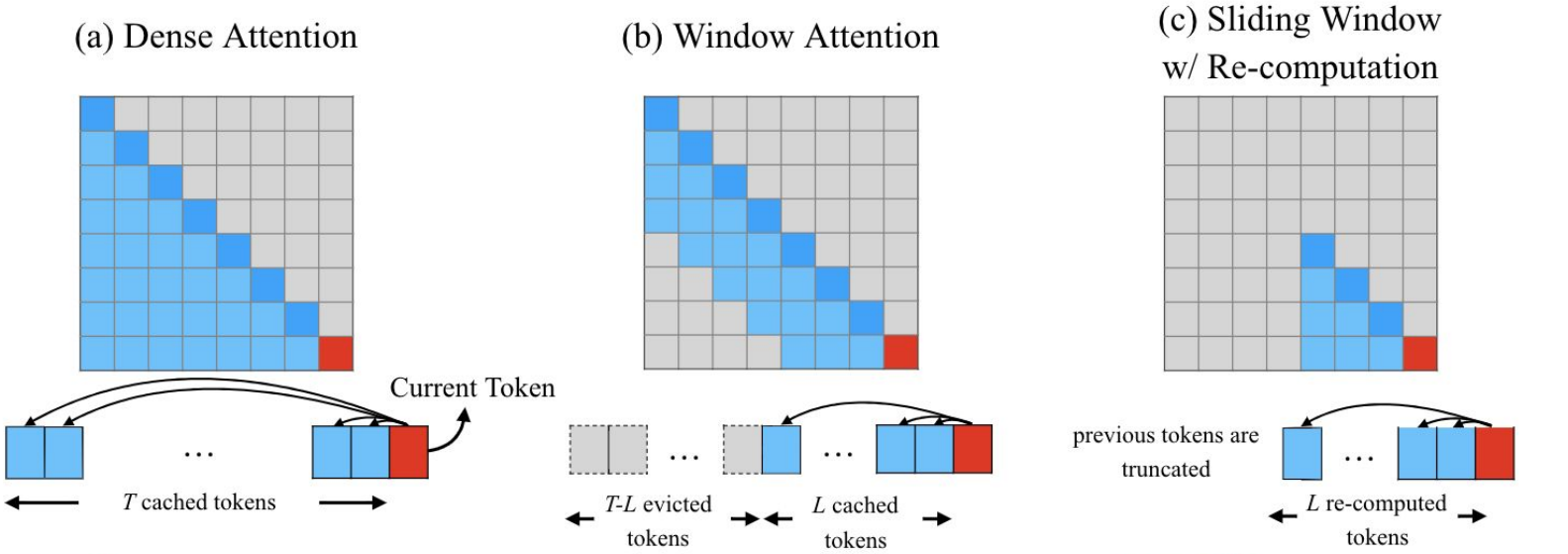
Introduction to StreamingLLM

KV Caching



GIF source: https://medium.com/@joaolages/kv-caching-explained-276520203249

Issues with Attention Methods



$O(T^2)$ ✗ **PPL:** 5641 ✗

Has poor efficiency and performance on long text.

$O(TL)$ ✓ **PPL:** 5158 ✗

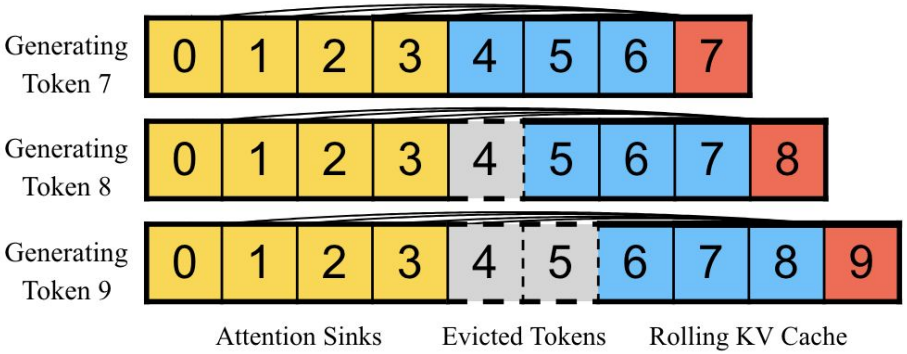
Breaks when initial tokens are evicted.

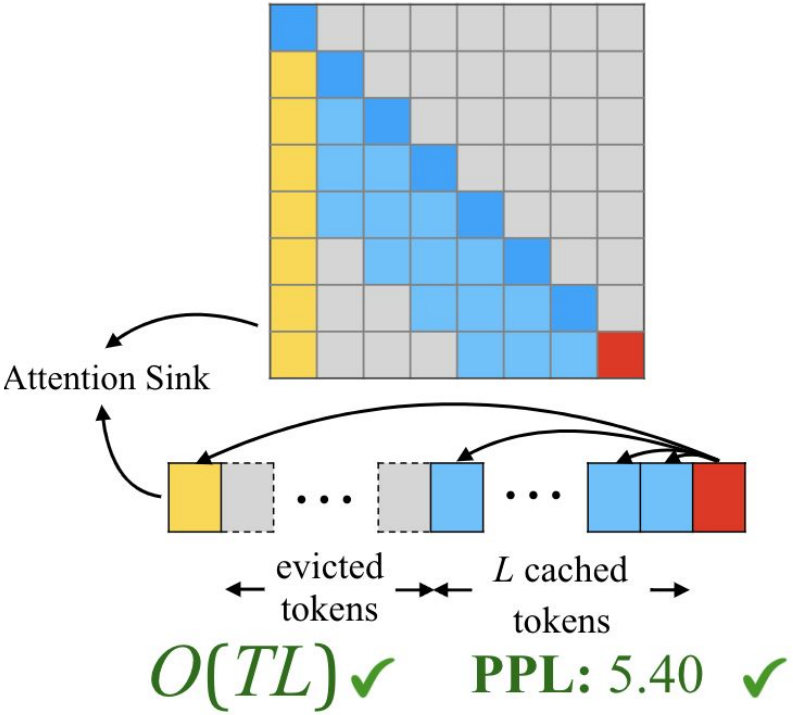
$O(TL^2)$ ✗ **PPL:** 5.43 ✓

Has to re-compute cache for each incoming token.

Source: Guangxuan Xiao, Han Lab

StreamingLLM with Attention Sinks





Can perform efficient and stable language modeling on long texts.

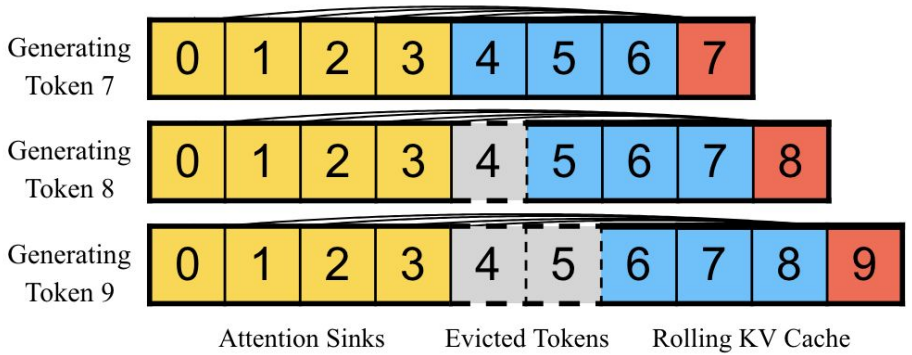
"StreamingLLM provides **up to 22.2x speedup** over the baseline, making LLMs for real-time streaming applications feasible."

Source: Guangxuan Xiao, Han Lab

Limitations

Even though we can generate text indefinitely, what if we evict tokens with relevant information?

Can we extend StreamingLLMs to
- Long Q&A
- Summarization



Source: Guangxuan Xiao, Han Lab

Methodology

Retrieval Augmented Generation

Idea: Retrieve information evicted from the KV Cache

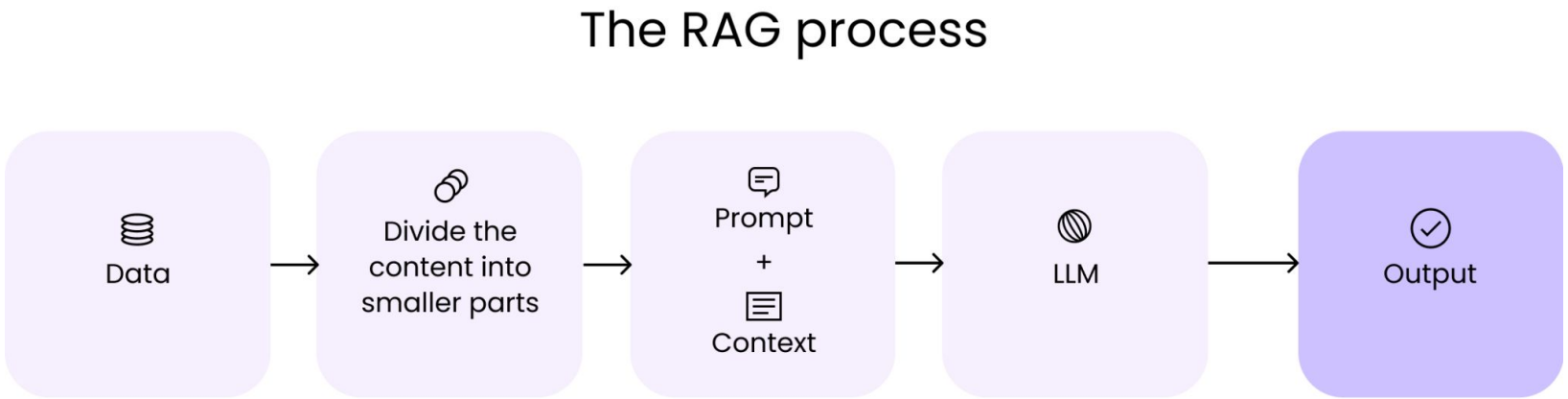


Image source: https://writer.com/blog/retrieval-augmented-generation-rag/

Retrieval Augmented Generation (Implementation)

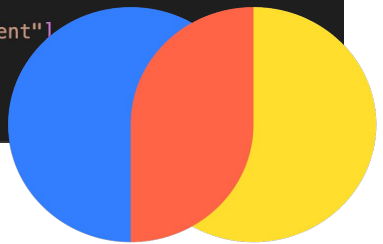
Embedding Model: all-MiniLM-L6-v2

- **Transforms strings into size 384 vectors**

Vector database:

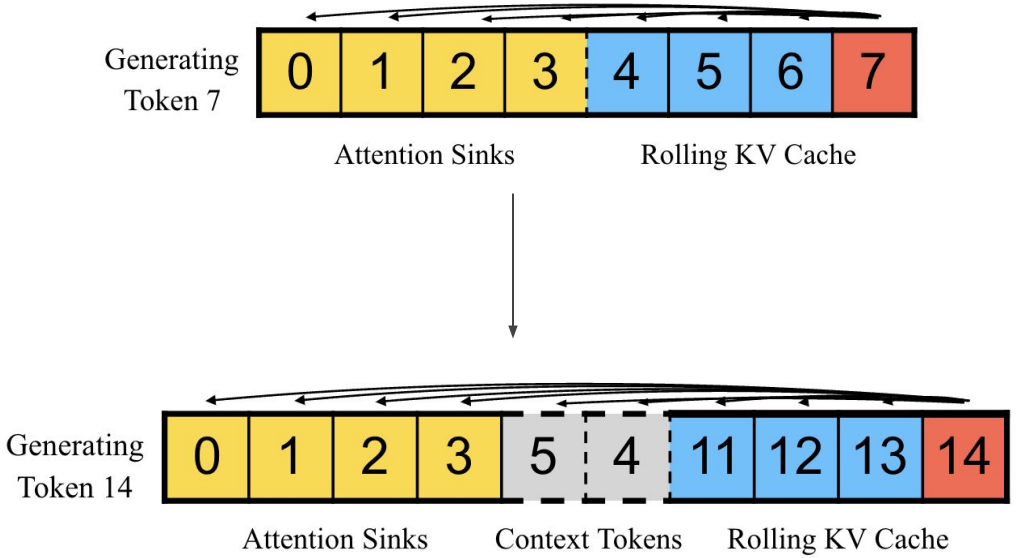
- **Chromadb**
- Try L2, Inner product, Cosine similarity

```python
import chromadb
client = chromadb.Client()

collection = client.create_collection(
    "sample_collection",
    embedding_function=all_MiniLM_L6_v2_embed_fn
)

collection.add(
    documents=["evicted token group 1", "evicted token group 2"],
    ids=["doc1", "doc2"],
)

results = collection.query(
    query_texts=["This is a query document"]
    n_results=2,
)

```



KV Cache Changes

- In order to accommodate the new context tokens, we add a section into the KV Cache between the attention sink and the sliding window



KV Cache Changes

- In order to accommodate the new context tokens, we add a section into the KV Cache between the attention sink and the sliding window

```
37    def __init__(
38        self,
39        start_size=4,
40        recent_size=512,
41        k_seq_dim=2,
42        v_seq_dim=2,
43        context_size=300,
44    ):
45        print(f"StartRecentKVCache: {start_size}, {recent_size}")
46        self.start_size = start_size
47        self.recent_size = recent_size
48        self.context_size = context_size
49        self.cache_size = start_size + recent_size
50        self.k_seq_dim = k_seq_dim
51        self.v_seq_dim = v_seq_dim
52        self.k_slice = DIM_TO_SLICE[k_seq_dim]
53        self.v_slice = DIM_TO_SLICE[v_seq_dim]
54        self.last_saved = 0
55        self.context_added = False
```

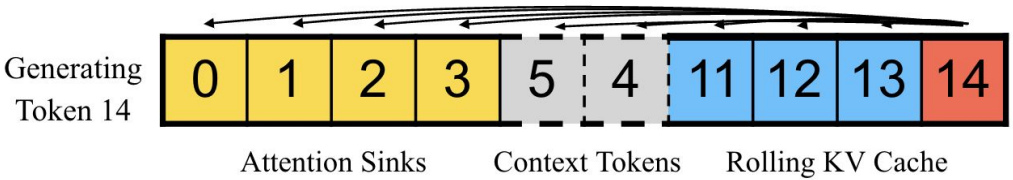
KV Cache Changes

- Store the tokens and evicted key value pairs to be used with the RAG

```
33
34    tokens = []
35    kvs = None
36
```

```
114     def add_context(self, past_key_values, context):
115         """
116         Add the context tokens to the past_key_values and update
117         the size of the context tokens within the kv cache
118         """
119         if past_key_values is None:
120             return None
121         assert(context[-1][-1].size(self.k_seq_dim) == self.context_size)
122         seq_len = past_key_values[0][0].size(self.k_seq_dim)
123         return [
124             [
125                 torch.cat(
126                     [
127                         self.k_slice(k, 0, self.start_size),
128                         k2,
129                         self.k_slice(k, self.start_size + self.context_size, seq_len)
130                     ],
131                     dim=self.k_seq_dim,
132                 ),
133                 context,
```

Attention Sink with k-clusters

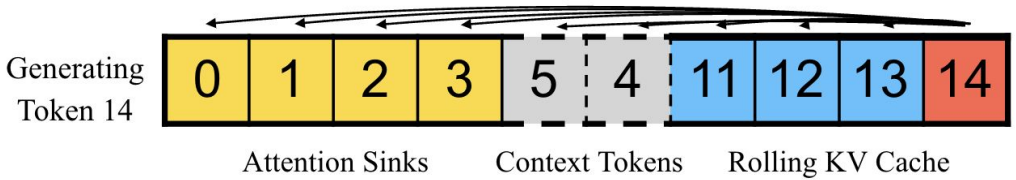


Llama-2-13B	PPL (↓)
0 + 1024 (Window)	5158.07
4 + 1020	5.40
4"\n"+1020	5.60

s

Source: Guangxuan Xiao, Han Lab

Attention Sink with k-clusters



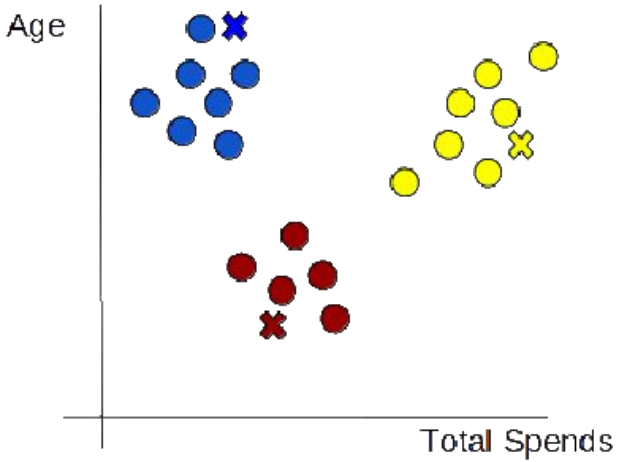
Idea: Could we capitalize on the attention sink?

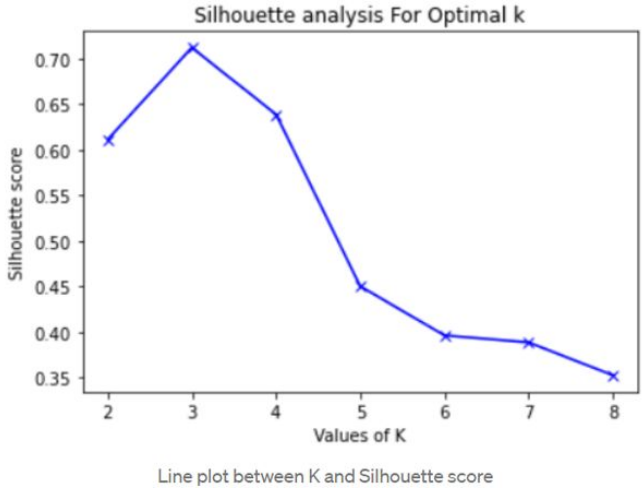
Llama-2-13B	PPL (↓)
0 + 1024 (Window)	5158.07
4 + 1020	5.40
4"\n"+1020	5.60

s

Source: Guangxuan Xiao, Han Lab

Attention Sink with k-clusters





s

Image source: https://www.analyticsvidhya.com/blog/2021/05/k-mean-getting-the-optimal-number-of-clusters/

Evaluation & Datasets

Evaluation & Datasets

Q&A	SQuAD_v2, TriviaQA; EM, F1
Summarization	Multi-news; ROUGE
Story Generation	WritingPrompts, Fandom Dev; BLEU

Source: https://paperswithcode.com/

Challenges & Future Work

Challenges & Future Work

- Challenges
  - Not enough compute on Colab
    -
  - Proper benchmarking and evaluation
- Future Work
  - More testing and iteration of methodology