

# SurfaceLogicKV

---

SurfaceLogicKV is a novel two-stage KV cache compression library for KV Cache compression.

## Install Packages

---

Please see requirements.txt.

## Two-stage method

---

### Surface Memorization and Logic Construction attention behavior Identification

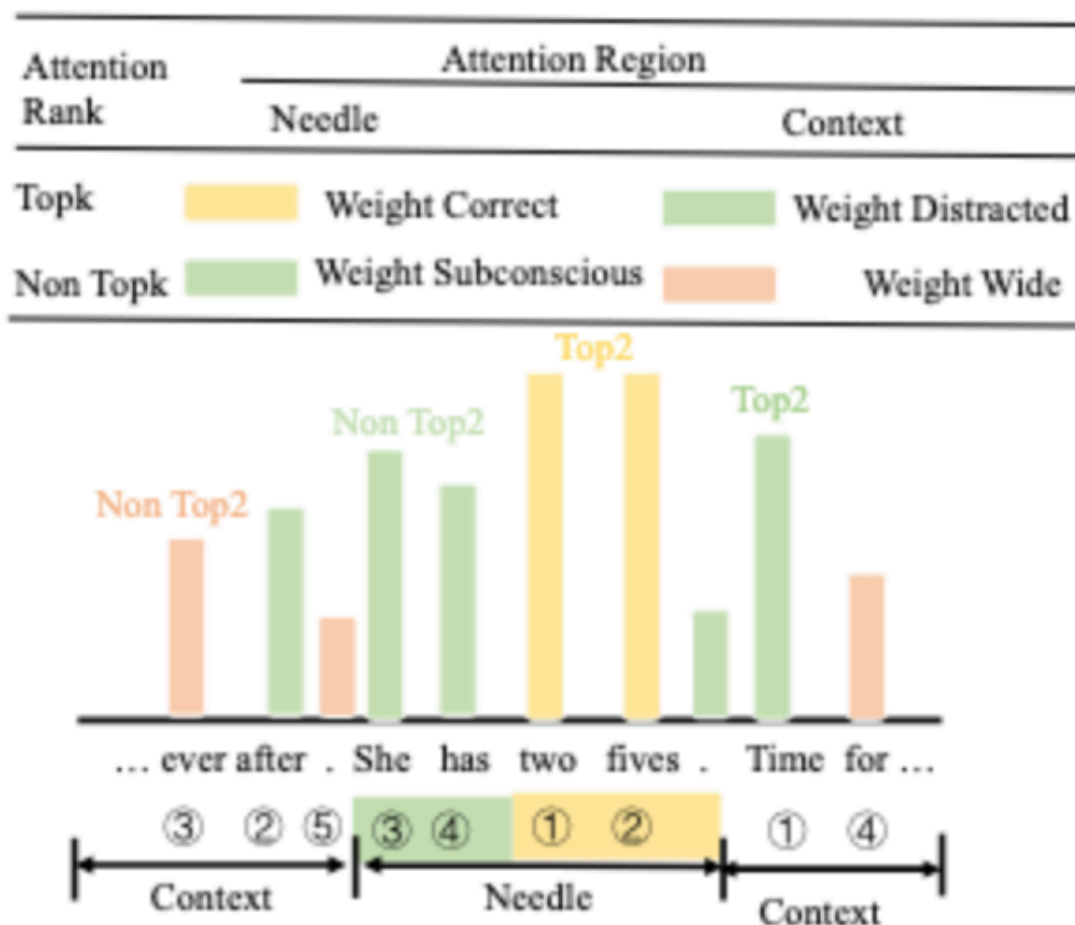
In the first stage, we identify the INFsc, SFsc and LGsc for different attention behavior using a Needle-in-a-Haystack test. This test helps us distinguish between:

- Human Correct - NIAH Weight Correct - Transformer Surface Memorization behavior defined by ourselves They can clearly focus on the correct answer area with copy-paste-reasoning.
- Human Ambiguous - NIAH Weight Distracted and Subconscious - Transformer Logic Construction behavior defined by ourselves Even when generally on the right track, they struggle to precisely pinpoint the answer due to a surrounding or long-distance related context. For example, when you hear “attention is all you need”, you relate it to Transformer.
- Human Wrong - NIAH Weight Wide They are easily misled by wide information in incorrect areas.

#	Question	Needle	Answer
1	What are good ways to spend time in campus?	The good ways to spend time in campus include relax and do nothing.	Relax and do nothing.
2	What habits are beneficial for health during Ph.D. career?	The beneficial habits during Ph.D. career contain exercise and healthy diet.	Exercise and healthy diet.
3	Which year did Tom start high school?	Tom was born in 2005. Tom started high school fifteen years after he was born.	Tom started high school in 2020.

Q: How much money does she have?

A: She has ten.



Attention Weight Distribution

Figure 1: Weight attention behavior confusion matrix with Needle-In-A-Haystack (NIAH) (Kamradt 2023). “Two” and “Five” also relate to time, like “two o’clock”.

Argument	Description
<code>--model_path</code>	Path to a local LLM checkpoint or the model identifier on Hugging Face Hub (e.g. <code>llama-3-8b-instruct</code> ).
<code>--haystack_dir</code>	Directory containing background text files ( <code>.txt</code> ). These files form the “haystack” corpus for the search task.
<code>--min_len</code>	Minimum context length (in tokens) to test.
<code>--max_len</code>	Maximum context length (in tokens) to test.
<code>--context_intervals</code>	Number of evenly spaced context-length intervals between <code>min_len</code> and <code>max_len</code> .
<code>--gpu</code>	GPU device ID to use (e.g. <code>0</code> , <code>1</code> ). Defaults to CPU if omitted.
<code>--top_k_infscore</code>	K value for “Top-K” selection when computing InfScore (the number of top-scoring tokens to consider).
<code>--depths</code>	Comma-separated list of percentage depths (e.g. <code>0.1,0.5,0.9</code> ) at which to insert the “needle” into the haystack.

```
1 python structure_head_InfScore.py
```

## Collaborative Layer-Head KV Cache Budget Allocation

During the prefill stage, we allocate the KV cache budget for each head based on the inter-layer aggregation and intra-layer distribution of INFsc.

By considering both layer-level and head-level information, DiliLazyKV dynamically assigns larger KV cache budgets to heads that contribute more significantly to inference performance and smaller budgets to less critical heads.

```
1 max_capacity_prompts=128
2
3 devices=(0 1)
4 head_choices=('reason')
5 betas=(1.351)
6 counter=0
7 for((i=0;i<${#head_choices[@]};i++));do
8     for((j=0;j<${#betas[@]};j++));do # This loop will also run once as
betas has 1 element
9         device=${devices[counter]}
```

```

10     head_choice=${head_choices[i]}
11     beta=${betas[j]}
12     temp=1
13     nohup bash head_base.sh \
14         $device \
15         ReasonKV \
16         ${max_capacity_prompts} \
17         flash_attention_2 \
18         meta-llama/Meta-Llama-3-8B-Instruct \
19         $head_choice \
20         $beta \
21         $temp >
    ./longbench_logs/llama3_ReasoniKV_${head_choice}_base${max_capacity_promp
s}_beta${beta}_temp${temp}.txt 2>&1 &
22     ((counter+=1))
23 done
24 done

```

## Key Features and Benefits

- Two-Stage Compression: Separates head importance identification from budget allocation for a more principled approach.
- Different Attention Behaviors Distinction
- Collaborative Layer-Head Allocation: Considers both layer-level and head-level importance for fine-grained budget management.
- Balances Compression and Performance: Achieves significant KV cache reduction while maintaining high inference accuracy.
- Enhanced Robustness: Demonstrates stable performance across different tasks and length.
- Beneficial for Real-World Deployment: Enables efficient deployment of large language models with reduced memory footprint and bandwidth requirements.

## Acknowledgement

Thanks [HeadKV-R2](#) for providing open-source code and data, which has been invaluable to the development of SurfaceLogicKV.