SemanticKV: A Semantics-Driven KV Cache Eviction Framework for Efficient Long Prompt LLM Inference

Meng Huang
National Elite Institute of Engineering
Northwestern Polytechnical University
Xi'an, China
huangmeng@mail.nwpu.edu.cn

Baolong Wang
Rapid-Fire Weapon Research Institute
Xi'an Kunlun Industry (Group) Co., Ltd
Xi'an, China
732032773@qq.com

Xiao Zhang*
School of Computer Science
Northwestern Polytechnical University
Xi'an, China
zhangxiao@nwpu.edu.cn

Shujie Han*
School of Computer Science
Northwestern Polytechnical University
Xi'an, China
shujiehan@nwpu.edu.cn

***Abstract*—Key-Value (KV) cache is essential for accelerating attention computation in Large Language Models (LLMs). However, as the cache grows linearly with input length, long-prompt inference imposes substantial GPU memory and latency overheads. Existing KV cache eviction methods typically focus on the decoding phase and rely on heuristic rules, which often lead to suboptimal and unstable performance. To address these limitations, we propose SEMANTICKV, a general and plug-and-play framework for long-prompt KV cache eviction. SEMANTICKV identifies semantically active tokens based on the L2-norm of their query vectors, estimates their importance through multi-head attention and retains critical tokens via a head-wise grouped selection strategy. This design preserves semantic integrity across attention heads while significantly reducing redundant memory usage. Extensive experiments demonstrate that SEMANTICKV achieves up to 5x KV cache reduction while maintaining over 95% of the original model performance. For sequences exceeding 100K tokens, SEMANTICKV improves throughput by 2.4x and reduces peak memory consumption by 29%. These results establish SEMANTICKV as a practical and stable solution for efficient long-prompt LLM inference under constrained GPU resources.**

I. INTRODUCTION

With the rapid development of artificial intelligence, large language models (LLMs) have achieved revolutionary breakthroughs in natural language processing. Mainstream models like GPT-5 [12], Claude-4.5 [5], and Gemini-2.5-Pro [7] have shown significant advancements in long-prompt tasks. This improvement in long-prompt capabilities opens up new possibilities for complex tasks such as multi-turn dialogue, long-document analysis, and codebase understanding.

However, efficient long-prompt inference remains challenging due to inherent limitations of the Transformer architecture. A primary limitation stems from the linear growth of the KV cache with input length, which consumes substantial GPU memory and increases inference latency. As a result, the computational cost of each decoding step scales with the number of historical tokens, while the expanding cache size restricts batch processing capacity and increases deployment expenses.

During the inference process, Transformers-based models [17] typically consist of two phases: the prefill phase and the decoding phase. The prefill phase processes the entire input prompt and generates initial KV cache, while the decoding phase uses this cache to autoregressively generate output tokens. Currently, most existing KV cache eviction methods focus primarily on the decoding phase, overlooking the critical need for prompt compression during the prefill phase. This oversight is particularly problematic in practical applications such as conversational agents and AI assistants, where input prompts like multi-turn dialogues and extensive codebases often far exceed the generated responses in length, making the prefill phase a critical memory bottleneck. Therefore, how to achieve effective KV cache compression while maintaining output quality given long and potentially redundant inputs remains an open challenge.

Through analysis of pre-trained LLMs, the repetitive attention pattern has been observed [10], where tokens attend mainly to a few key tokens, following a power-law distribution. Based on this observation, we propose SEMANTICKV, a semantics-driven KV cache framework for long-prompt LLM inference. It consists of two techniques, i.e., semantic query selection via L2-norm and head-wise structured retention. Our evaluation shows that with only 20% of the original KV cache, SEMANTICKV significantly speeds up inference and reduces GPU memory consumption, while delivering consistent performance on long-prompt tasks. Our main contributions are as follows:

- Token semantic activity is quantified via the L2-norm of query vectors, measured independently across attention heads to preserve head-specific semantics.
- A three-stage retention framework is designed to maintain contextual continuity and semantic integrity, including start, middle, and recent regions.
- SEMANTICKV is a plug-and-play method that requires no additional training. It can reduce memory usage by about 25% and accelerate decoding by up to 2.3x, effectively lowering deployment costs for long-prompt.

II. BACKGROUND AND RELATED WORK

A. Generative LLM Inference

Generative Large Language Model (LLM) inference typically consists of two distinct phases: the *prefill* phase and the *decoding* phase. In the prefill phase, the model tokenizes the entire input prompt and processes it through a stack of Transformer blocks [17]. Each block generates *query*, *key*, and *value* (QKV) matrices and applies multi-head self-attention, where multiple heads attend to different semantic subspaces to

* Xiao Zhang and Shujie Han are the corresponding authors.

capture diverse contextual relationships. Attention computation is based on the scaled dot-product mechanism: for each head, the similarity between queries and keys is measured by the dot product QK^T, scaled by $\sqrt{d}$ (where d is the head dimension). The resulting scores, referred to as *attention logits*, are normalized via a softmax function to produce *attention weights*, which are then used to aggregate values into context-aware representations. The outputs are subsequently passed through a feed-forward network (FFN). In contrast, the decoding phase proceeds autoregressively: at each step, the model generates a single token by updating the QKV matrices, recomputing attention over the growing context, and applying the FFN in every block. While multi-head attention enriches semantic representation, it also amplifies computational and memory demands during decoding, especially for long sequences.

To address these efficiency challenges, modern LLMs employ the *KV cache*, which stores previously computed key and value vectors to avoid redundant computation. Instead of recalculating attention for all past tokens, including those from the input prompt, the cache allows new tokens to reuse prior keys and values, reducing per-step attention cost from linear to constant with respect to sequence length. This optimization enables fast and scalable autoregressive generation. However, the cache size grows linearly with sequence length, creating a memory bottleneck for long prompts.

Recent work explores *KV cache eviction* as a solution, selectively discarding less influential tokens while preserving those most critical for maintaining semantic coherence [1], [16]. Token importance is often measured by the *accumulated attention score* [20], which aggregates attention weights assigned to a token's key by subsequent queries. Tokens with higher scores exert greater influence on future predictions, and their KV pairs are prioritized in the cache. In this work, our SEMANTICKV combines multi-head attention for semantic diversity with intelligent cache management for efficiency so as to achieve a balance between representational richness and computational scalability.

B. Related Work

Our work targets efficient long-prompt LLM inference by addressing a critical challenge: managing the GPU memory footprint of the KV cache during both prefill and decoding phases. We categorize the existing approaches into two broad categories: KV cache eviction and other memory optimizations.

KV cache eviction. To mitigate the memory overhead of KV cache during long-prompt inference, recent studies propose eviction-based approaches that selectively prune less important tokens. The sliding window method [14] retains only the most recent tokens, discarding earlier context. H2O [20] prioritizes historical tokens with high accumulated attention scores, while Scissorhands [10] prunes unimportant tokens and redundant heads in stages. SepLLM [3] preserves delimiters summarizing text, and SnapKV [9] combines local snapshots with representative global tokens. Despite their effectiveness, these methods often discard intermediate contextual information, which can degrade coherence.

The most closely related works are NACL [4] and StreamingLLM [18], both leveraging KV cache eviction to accelerate inference. NACL scores tokens using attention aggregated from proxy tokens and then samples from this distribution, introducing non-determinism. StreamingLLM retains initial tokens alongside a sliding window of recent tokens for infinite-length streaming generation, but relies on fixed positional heuristics. SEMANTICKV overcomes these limitations with a semantics-driven eviction policy that combines L2-norm and head-wise scoring to preserve tokens most critical for maintaining coherence across long contexts.

Other memory optimizations. Beyond eviction, existing work explores model-level modifications and system-level optimizations to reduce the KV cache size. Model-level methods alter attention mechanisms; for example, Grouped-Query Attention (GQA) [2] shares keys and values across head groups, significantly shrinking the cache but requiring model redesign or retraining. System-level approaches apply compression or offloading. FlexGen [15] uses 4-bit quantization with CPU offloading, while HyperGen [8] partitions model parameters and prefill computations to optimize GPU memory usage in resource-constrained environments. KIVI [11] further reduces memory via 2-bit quantization with minimal accuracy loss. Although these strategies alleviate memory pressure, they do not address the attention complexity inherent to long sequences.

III. OBSERVATION

Computing accumulated attention scores for all tokens is the most accurate way to measure token importance for KV cache eviction. However, this requires evaluating attention for every query and key pair in all heads, which incurs prohibitive cost during long-prompt inference. In this section, we present the key insight that motivates the design of SEMANTICKV.

The core question is whether we can identify a small subset of tokens whose queries are sufficiently informative to approximate full attention behavior, thereby reducing the cost of computing QK^T. In multi-head attention, QK^T measures the similarity between queries and keys. Intuitively, a query that encodes rich semantic information tends to produce high similarity with multiple keys. This insight suggests an optimization: instead of using all queries, we can select a subset of semantically rich queries to reduce the computational overhead of QK^T.

To operationalize this idea, we need a metric to quantify semantic richness. A query vector with larger component values carries a stronger signal, making it more influential in attention computation, which we refer to as *activation*. Since each attention head captures different aspects of semantics, a token whose query vectors exhibit strong activation across heads is likely to convey more semantic information. We quantify this activation using the L2-norm of the query vector (formal definitions presented in Section IV-A): a higher L2-norm implies greater magnitude and richer semantics. Thus, L2-norm serves as an efficient proxy for identifying tokens whose queries are most informative for attention computation.

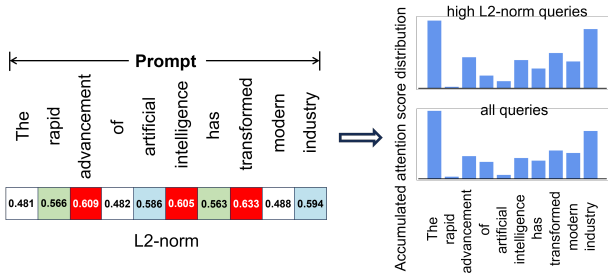


Fig. 1. Comparison of accumulated attention score distribution between selecting high L2-norm queries and all queries.

Fig 1 illustrates this concept. We compute the L2-norm for each token's query vector and select tokens with the highest norms (e.g., *transformed*, *advancement*, *intelligence*), which carry the core semantic content of the sentence. In this example, selecting the top three queries by L2-norm yields an accumulated attention score distribution similar to that obtained using all queries. This shows that semantically rich queries can effectively represent the sentence while reducing computational complexity from $O(N^2d)$ to $O(\lambda N^2 d)$, where N is the input sequence length and λ is the fraction of selected queries with $0 < \lambda \ll 1$. We integrate this observation into the design of SEMANTICKV in Section IV and evaluate its effectiveness in Section V.

IV. DESIGN

Building on the observation in Section III, we propose SEMANTICKV, a semantics-driven KV cache eviction framework. The design consists of two key components: semantic query selection and head-wise structured eviction (shown in Fig 2).

A. Semantic Query Selection via L2-Norm

During the prefill phase, each input token t generates a query vector $q_t \in \mathbb{R}^d$ and a key vector $k_t \in \mathbb{R}^d$ for every attention head. Our objective is to identify a compact set of tokens whose queries are semantically informative and can guide downstream importance estimation. To achieve this, we quantify each token's query activation, analyze its effect on attention computation, and then select tokens with the strongest activation as semantic representatives.

Query activation measurement. Let $q_t = (q_{t,1}, q_{t,2}, ..., q_{t,d})$ be the query vector of token t and $q_{t,j}$ denote the j-dimension of that vector. We measure the query activation of token t through the L2-norm of its query vector: $\|q_t\|_2 := \sqrt{\sum_{j=1}^{d}(q_{t,j})^2}$. A large L2-norm indicates that the query vector has a strong magnitude, suggesting that the token carries richer semantic information and is likely to influence the attention distribution more substantially.

Revisit of attention logits and effect of query activation on attention. Although attention logits and attention weights are standard components of the Transformer architecture [17], we restate their definitions here to clarify why the L2-norm serves as an effective metric for semantic query selection. The attention logit between the query vector of token t and the key vector of token i is defined as $\ell_{t,i} := \frac{q_t \cdot k_i}{\sqrt{d}}$. Expanding the dot product yields $q_t \cdot k_i = \|q_t\|_2 \|k_i\|_2 \cos\theta_{t,i}$, where $\theta_{t,i}$ is the angle between the query and key vectors. This expression makes clear that a larger query norm $\|q_t\|_2$ amplifies the attention logits for all keys, increasing the overall influence of token t in the sequence. The corresponding attention weights are obtained via softmax: $a_{t,i} := \text{softmax}(\ell_{t,i})$. Thus, queries with higher activation naturally dominate the resulting attention distribution, making them strong candidates for semantic query selection. By explicitly expanding the attention formulas, we highlight how query magnitude directly shapes attention scores, grounding our use of the L2-norm in the underlying mechanics of attention rather than treating it as a heuristic.

Semantic query selection. Leveraging the above insights, we identify semantically rich queries by selecting those with the highest L2-norms. For each head, we select the top-λN of queries with the largest L2-norms as semantic queries (e.g., λ=10%). These queries form the part of an observation set used in Section IV-B to estimate token importance while avoiding expensive full-query attention.

B. Head-wise Structured Retention Strategy

Based on the semantic query selection in Section IV-A, we describe how these queries are used to estimate the token importance and build a compressed KV cache. The goal of this stage is to determine which input tokens carry sufficient contextual information to merit retention during decoding.

Head-wise importance scoring. To account for the different roles that tokens play along the sequence, we divide the sequence into three disjoint regions: a *sink region* containing the initial tokens that encode global context, a *recent region* containing the final tokens that preserve local coherence, and a *middle region* containing all remaining tokens whose importance varies.

The middle region contains most tokens, but only a subset meaningfully contributes to future decoding. We define an *important token* as one whose *keys* consistently receive high attention weights from subsequent queries, indicating sustained influence on the model's attention distribution. As different attention heads specialize in distinct semantic subspaces [17], importance must be evaluated independently per head to preserve this diversity.

To estimate the importance of tokens in the middle region, we construct an observation set of queries (denoted by $\mathcal{O}$). This set consists of i) the semantic queries selected in Section IV-A and ii) the queries of recent tokens, which are naturally involved in computing attention over the middle region. For each token t in the middle region and each head h, we define its *importance* as the average attention weight its key receives from all queries in $\mathcal{O}$: $r_t^{(h)} = \frac{1}{|\mathcal{O}|}\sum_{j\in\mathcal{O}} a_{t,j}^{(h)}$, where $a_{t,j}^{(h)}$ denotes the attention weight assigned to the key of token t by the query of token j in head h. Intuitively, $r_t^{(h)}$ is large when the key of t repeatedly receives high attention by many informative queries, indicating that token t plays an important role in contextual reasoning.

Retention set after selecting important middle tokens. Based on the token importance, we select a small fraction of the most important middle tokens for each head (e.g., 20%). To preserve

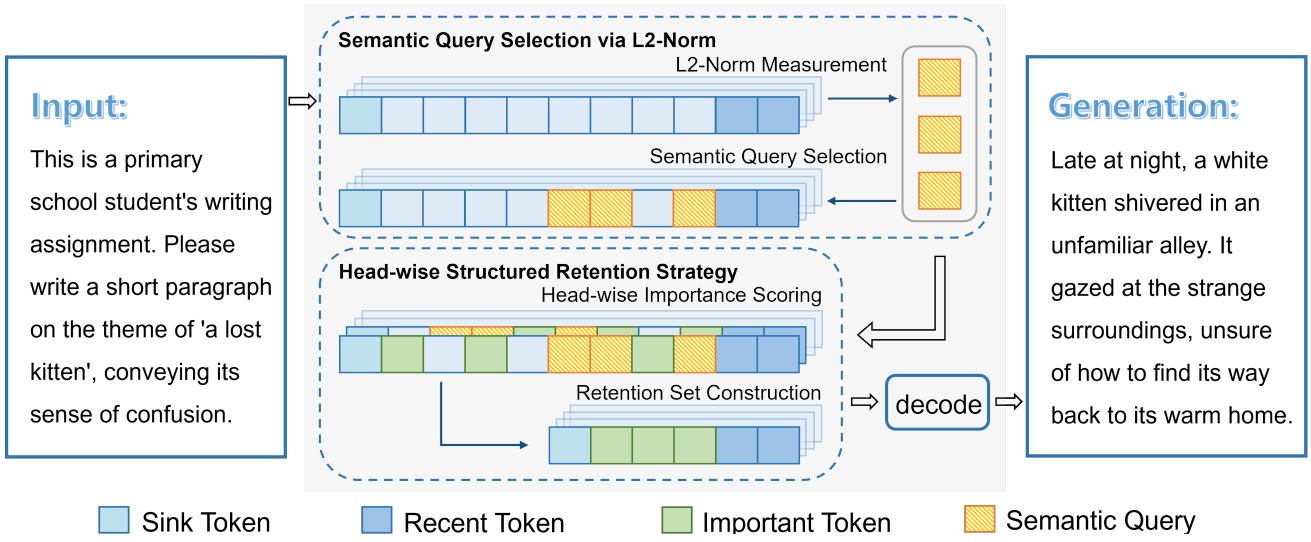


Fig. 2. Illustration of the framework of SEMANTICKV.

global and local context, we retain all sink tokens, all recent tokens, together with the selected important middle tokens across heads to perform the following LLM inference.

V. EVALUATION

A. Methodology

Implementation. We prototyped SEMANTICKV based on NACL [4] with 210 LoC of changes. We modify NACL's KV cache eviction in the prefill phase and realize the selection of semantic queries via the L2-norm based on PaddlePaddle [13]. We implement the head-wise structured retention strategy in the computation of attention by estimating of token importance, using the optimized FlashAttention-2 algorithm [6] .

Setup. We evaluate the SEMANTICKV eviction algorithm from two perspectives: (i) downstream task performance under constrained GPU memory and (ii) its impact on decoding efficiency and memory usage during long-sequence generation. Unless otherwise stated, the algorithm is configured with 256 sink tokens, a 1% recent-token ratio, and a 1% semantic-token ratio. We conduct experiments on three widely used decoder-only LLMs, i.e., LLaMA3.1-8B-Instruct, LLaMA3.2-3B and Mistral-7B, across ten tasks from the InfiniteBench benchmark [19]. These tasks span code reasoning, mathematical computation, long-document understanding, multi-turn dialogue, and information retrieval, with average input lengths exceeding 100K tokens. We measure task accuracy across varying memory budgets, along with decoding speed, throughput, and memory consumption. All experiments are performed on a local server equipped with four NVIDIA GeForce RTX 3090 GPUs.

Baselines. We compare our method with four baselines of KV cache eviction strategies: Full KV Cache, NACL [4], StreamingLLM [18], Scissorhands [10] and H2O [20].

B. Long-prompt Downstream Tasks

We evaluate SEMANTICKV on long-prompt reasoning tasks with input lengths of 32K, 64K, and 72K tokens, as shown in Table I. It consistently matches or exceeds the baselines while maintaining performance close to full cache. For example, at 64K, SEMANTICKV achieves 23.99% accuracy, comparable to full cache and higher than other methods by 1 to 6 points. Performance degradation across Q&A and reasoning tasks is minimal, while baselines like StreamingLLM and Scissorhands show larger drops up to 6.86 points. SEMANTICKV achieves an effective balance between memory efficiency and task performance for long-prompt LLM inference. As shown in Fig. 3, we further evaluate SEMANTICKV on En.MC and Retrieve.PassKey tasks using LLaMA3.2-3B and Mistral-7B, which are from two distinct families. We compare full cache, NACL, and SEMANTICKV in terms of task accuracy. Across both models and datasets, SEMANTICKV consistently maintains strong performance, surpassing NACL in accuracy while approaching full-cache results. These results demonstrate the generality and effectiveness of SEMANTICKV across different models and long-prompt tasks.

C. Inference Efficiency

To evaluate efficiency of SEMANTICKV, we analyze decoding speed, GPU memory usage, and KV cache compression before and after optimization.

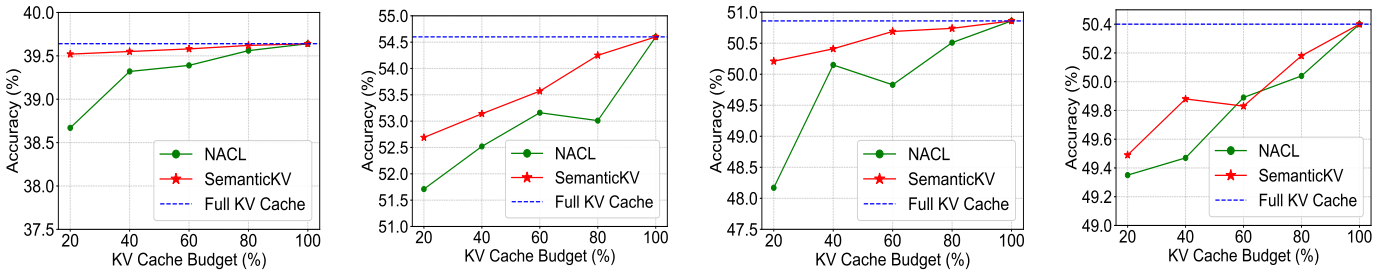
Decode performance. We evaluate decode performance of SEMANTICKV in terms of decoding latency (ms/token) and throughput (tokens/s) for one batch, reflecting real-time responsiveness and generation efficiency. Fig. 4(a) shows that the full cache baseline's latency grows roughly linearly with sequence length, exceeding 150 ms/token for 48K inputs. By utilizing only 20% of the KV cache, SEMANTICKV reduces latency to approximately 70 ms/token, achieving a 2.1x speedup. It maintains this stable latency even at a context length of 102K, whereas the full cache implementation runs out of memory beyond 48K.

Fig. 4(b) shows that the throughput of full cache decreases from 12.93 tokens/s at 24K input length to 5.71 tokens/s at

TABLE I.

ONLY 20% OF THE KV CACHE IS RETAINED. THE INPUT SEQUENCE LENGTHS ARE 23K, 64K, AND 72K, AND EVALUATIONS ON LONG-TEXT TASKS ARE CONDUCTED USING THE LLAMA3.1-8B-INSTRUCT MODEL UNDER DIFFERENT EVICTION STRATEGIES.

length	method	Document/Choice QA			Understanding		Synthetic Retrieval		Code		Math	Avg	Δ
		QA.Zh	QA.En	MC	Dia.En	En.Sum	Number	PassKey	Debug	Run	Find		
32K	Full Cache	6.88	6.42	54.59	10.00	27.44	27.12	27.13	3.30	–	38.29	20.12	–
	H2O	4.14	4.21	51.67	3.10	16.28	20.48	24.36	3.24	–	38.27	16.58	-3.54
	NACL	6.53	5.09	**52.83**	5.50	20.00	23.89	**27.11**	3.55	–	38.29	18.28	-1.84
	StreamingLLM	5.14	5.01	48.23	1.24	17.61	16.75	18.57	1.94	–	36.47	15.10	-5.02
	Scissorhands	3.45	3.11	45.84	4.75	16.20	23.61	20.72	2.04	–	34.78	15.45	-4.67
	SEMANTICKV	**6.77**	**5.63**	**52.83**	**11.00**	**25.06**	**24.22**	26.95	**3.62**	**0.25**	**38.57**	**19.49**	**-0.63**
64K	Full Cache	7.24	6.26	63.20	11.70	28.61	50.98	54.27	1.38	0.74	34.11	25.85	–
	H2O	5.39	4.27	60.57	9.57	24.86	30.19	52.61	1.30	–	31.96	22.07	-3.78
	NACL	6.93	5.38	60.70	**11.50**	26.37	33.48	54.14	**1.78**	**1.25**	34.10	23.56	-2.29
	StreamingLLM	6.21	4.93	54.27	8.61	23.44	22.61	36.17	1.01	–	32.61	18.99	-6.86
	Scissorhands	5.10	3.24	55.18	8.37	22.18	21.97	49.51	1.46	0.69	**34.28**	20.20	-5.65
	SEMANTICKV	**7.21**	**6.10**	**60.96**	**11.50**	**28.59**	**34.69**	**54.23**	1.52	**1.25**	33.87	**23.99**	**-1.86**
72K	Full Cache	8.19	6.73	64.19	9.00	25.34	57.64	57.63	1.27	1.25	33.71	26.50	–
	H2O	7.18	2.37	63.02	7.74	19.61	37.94	53.32	1.03	0.93	30.74	22.39	-4.11
	NACL	7.22	4.03	65.06	9.00	24.42	39.15	**57.51**	1.02	**1.75**	34.00	24.32	-2.18
	StreamingLLM	6.25	5.09	61.84	5.70	20.75	32.26	51.07	0.74	0.61	31.87	21.62	-4.88
	Scissorhands	3.71	2.64	58.47	5.83	19.26	34.61	50.93	0.46	0.80	27.94	20.47	-6.03
	SEMANTICKV	**7.94**	**5.12**	**65.07**	**11.14**	**25.26**	**41.96**	56.42	**2.00**	1.74	**35.44**	**25.21**	**-1.29**



(a) Tested on En.MC with LLaMA3.2-3B (b) Tested on En.MC with Mistral-7B (c) Tested on Passkey with LLaMA3.2-3B (d) Tested on Passkey with Mistral-7B

Fig. 3. Comparison of benchmark results under a 64K input sequence across two large language models from distinct families.

56K. In contrast, SEMANTICKV maintains a stable throughput of around 13.7 tokens/s across all input lengths, delivering a approximately 2.4× improvement at 56K. Overall, SEMANTICKV significantly improves decoding efficiency, reducing latency and maintaining stable throughput for long sequences.

GPU memory usage. The GPU memory usage of SEMANTICKV with 20% KV cache retention is consistently lower than that of the full KV cache baseline across all tested input lengths. SEMANTICKV consistently reduces peak memory across all input lengths while preserving performance. Fig. 4(c) shows that at 64K tokens, peak memory decreases from 43.19 GB to 33.35 GB at 64K tokens and it drops from 58.24 GB to 40.98 GB at 112K tokens.

D. Ablation Study and Analysis

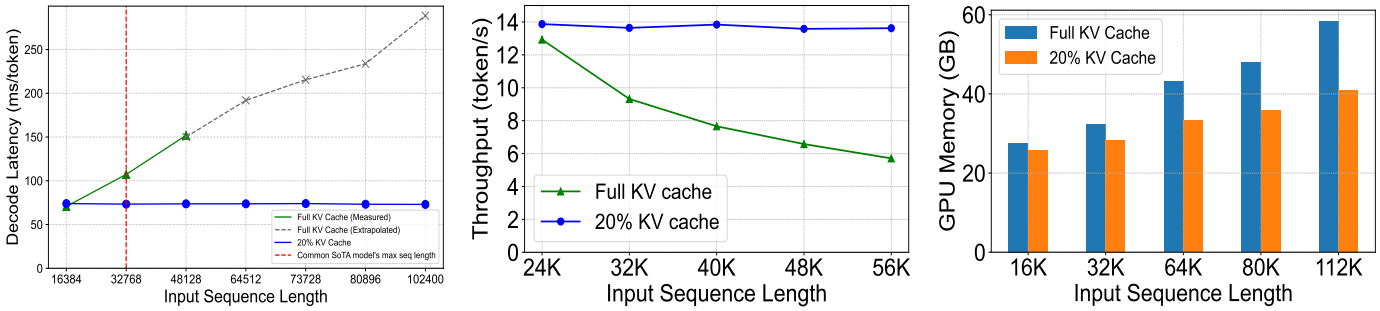
To validate the effectiveness of SEMANTICKV core design, we conduct ablation experiments including semantic query selection via L2-norm and head-wise structured retention on En.MC and Math.Find. Table II shows the accuracy.

TABLE II.

THE ABLATION STUDY AT 20% KV RETENTION. WE REPORT THE ACCURACY ON TWO TASKS.

	En.MC	Math.Find
SEMANTICKV	65.07	35.44
- sink token Reservation	63.84 (-1.23)	34.29 (-1.15)
- recent token Reservation	64.16 (-0.91)	34.01 (-1.43)
- L2 norm Selection	50.4 (-14.67)	28.82 (-6.62)
- Head-wise Eviction	62.27 (-2.8)	32.33 (-3.11)

Effectiveness of semantic query selection via L2-norm. Disabling semantic query selection via L2-norm decreases the accuracy from 65.07% to 50.4% on En.MC and from 35.44% to 28.82% on Math.Find. This shows that selecting queries with larger L2-norms effectively identifies high-information



(a) Comparison of decoding latency (b) Comparison of throughput (c) Comparison of peak memory usage

Fig. 4. Performance of the SEMANTICKV algorithm across various reasoning efficiency metrics.

context queries, guiding attention and avoiding errors from random selection.

Effectiveness of head-wise structured retention. Disabling head-wise structured retention drops the accuracy by 2.8% on En.MC and 3.11% on Math.Find, showing that multi-head attention encodes distinct semantic features. Removing the sink or recent segmentation further reduces performance. This result indicates the importance of early and recent tokens for long-range and short-range dependencies.

VI. CONCLUSION

We propose SEMANTICKV, a semantics-driven KV cache eviction framework. Inspired by a key observation on a small subset of semantic tokens, it leverages two key techniques, i.e., semantic query selection via L2-norm and head-wise structured retention strategy. The method reduces the KV cache to as low as 20% of its original size while maintaining output quality. The evaluation shows that SEMANTICKV improves the efficiency of LLM inference by reducing the memory usage while preserving the accuracy of LLMs, enabling more practical deployment of long-prompt LLMs.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (62502391).

REFERENCES

[1] M. Adnan, A. Arunkumar, G. Jain, P. J. Nair, I. Soloveychik, and P. Kamath. Keyformer: Kv cache reduction through key tokens selection for efficient generative inference. In *Proc. of the Seventh Annual Conference on Machine Learning and Systems*, page 114–127, 2024.

[2] J. Ainslie, J. Lee, M. De Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proc. of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, 2023.

[3] G. Chen, H. Shi, J. Li, Y. Gao, X. Ren, Y. Chen, X. Jiang, Z. Li, W. Liu, and C. Huang. Sepllm: Accelerate large language models by compressing one segment into one separator. In *Proc. of the 42nd International Conference on Machine Learning*, 2024.

[4] Y. Chen, G. Wang, J. Shang, and Cui. NACL: A general and effective KV cache eviction framework for LLMs at inference time. In *Proc. of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 7913–7926, 2024.

[5] ClaudeTeam and Anthropic. Introducing claude sonnet 4.5. https://www.anthropic.com/, Accessed: 2025-09-30.

[6] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré. Flashattention: fast and memory-efficient exact attention with io-awareness. In *Proc. of the 36th International Conference on Neural Information Processing Systems*, pages 16344–16359, 2022.

[7] GeminiTeam and Google. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint*. arXiv:2507.06261, 2025.

[8] L. Gong, K. Liu, X. Li, S. Han, P. P. Lee, Y. Hu, and D. Feng. Hypergen: Optimizing generative inference with long prompts for resource-constrained systems. In *Proc. of the 16th ACM SIGOPS Asia-Pacific Workshop on Systems*, pages 54–60, 2025.

[9] Y. Li, Y. Huang, B. Yang, B. Venkitesh, A. Locatelli, H. Ye, T. Cai, P. Lewis, and D. Chen. Snapkv: Llm knows what you are looking for before generation. In *Proc. of the 38th Annual Conference on Neural Information Processing Systems*, pages 22946–22970, 2024.

[10] Z. Liu, A. Desai, F. Liao, W. Wang, V. Xie, Z. Xu, A. Kyrillidis, and A. Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. In *Proc. of the 37th Annual Conference on Neural Information Processing Systems*, pages 52345–52364, 2023.

[11] Z. Liu, J. Yuan, H. Jin, S. Zhong, Z. Xu, V. Braverman, B. Chen, and X. Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. In *Proc. of the 41st International Conference on Machine Learning*, pages 32332–32344, 2024.

[12] OpenAI. Gpt-5 system card. https://cdn.openai.com/, Accessed: 2025-08-13.

[13] PaddlePaddleTeam. PaddlePaddle: An open-source deep learning platform originated from industrial practice. https://www.paddlepaddle.org.cn/, Accessed: 2024-07-22.

[14] O. Press, N. A. Smith, and M. Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *Proc. of the tenth International Conference on Learning Representations*, pages 3012–3036, 2022.

[15] Y. Sheng, L. Zheng, B. Yuan, Z. Li, M. Ryabinin, and Chen. Flexgen: High-throughput generative inference of large language models with a single gpu. In *Proc. of the 40th International Conference on Machine Learning*, pages 31094–31116, 2023.

[16] H. Tang, Y. Lin, J. Lin, Q. Han, S. Hong, Y. Yao, and G. Wang. Razorattention: Efficient kv cache compression through retrieval heads. In *Proc. of the 13th International Conference on Learning Representations*, pages 82100–82114, 2024.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Proc. of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010, 2017.

[18] G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis. Efficient streaming language models with attention sinks. In *Proc. of the twelfth International Conference on Learning Representations*, pages 16082–16102, 2023.

[19] X. Zhang, Y. Chen, S. Hu, Z. Xu, J. Chen, M. Hao, X. Han, Z. Thai, S. Wang, Z. Liu, and M. Sun. ∞Bench: Extending long context evaluation beyond 100K tokens. In *Proc. of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 15262–15277, 2024.

[20] Z. Zhang, Y. Sheng, T. Zhou, and T. Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *Proc. of the 37th Annual Conference on Neural Information Processing Systems*, pages 34661–34710, 2023.