ProtoKV: A Hybrid Semantic Prototype-based Framework for Efficient KV Cache Compression

Anonymous ACL submission

Abstract

Key-Value (KV) caching accelerates LLM inference but incurs high memory overhead. Existing methods focus on preserving only critical KV pairs for inference. While clustering-based strategies excel preserve critical KV pairs with semantic coherence, they suffer from computational inefficiency and limited parallelization. In this paper, we identify a dichotomy in token representations: while most tokens exhibit semantic similarity to their surrounding tokens, a distinct subset deviating from this pattern exhibits clustered semantic embeddings in the latent space. Leveraging this, we propose ProtoKV, a novel KV cache compression framework that combines chunk-aware local aggregation and LSH-driven global consolidation to construct hybrid semantic prototypes. These prototypes guide head-wise attention redistribution via cluster-aware pooling, efficiently retaining critical KV pairs. Experiments on LongBench show ProtoKV achieves 2.11% higher accuracy than state-of-the-art under identical memory constraints; in Needle-In-A-Haystack task, it achieves 96.8% retrieval accuracy at 1.6% cache retention. Furthermore ProtoKV reduces inference latency by up to 3.9× compared with clustering-based strategies.

1 Introduction

Large language models (LLMs), exemplified by *GPT* (Brown et al., 2020), *PaLM* (Chowdhery et al., 2023), and *LLaMA* (Touvron et al., 2023), have emerged as revolutionary forces in modern artificial intelligence. Their architectures are built upon the transformer framework (Vaswani et al., 2017), which superseded conventional sequential modeling paradigms through parallelized self-attention mechanisms and multi-scale feature representation. Built upon it, LLMs have demonstrated remarkable capabilities on tasks like dialogue generation (Li et al., 2024a), question answering (Ho et al., 2020) and logical reasoning (Wei et al., 2022).

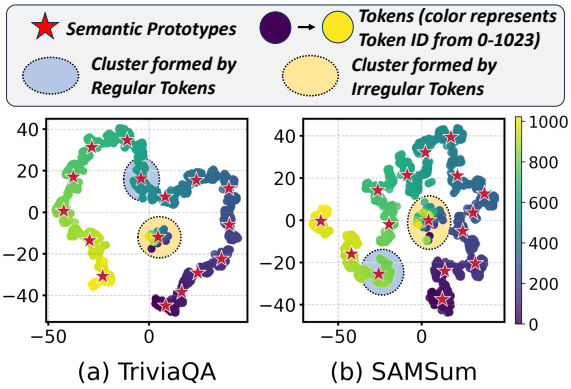


Figure 1: Clustering visualization of ProtoKV-processed key vectors from `Llama2-7B-chat`, where color represents token ID from 0-1023. As illustrated, regular tokens exhibit similar representations to their surrounding tokens, forming clusters with internally consistent colors. In contrast, irregular tokens typically violate such locality, resulting in clusters containing elements of diverse colors.

However, deploying large language models under fixed-memory hardware constraints entails substantial computational challenges.The primary bottleneck stems from the Key-Value (KV) cache mechanism, which maintains historical KV vectors to prevent recomputation but exhibits memory requirements scaling with respect to batch size b and sequence length n. Specifically, the cache memory footprint grows as $\mathcal{O}(b \cdot n)$ and often surpasses static model parameter memory by an order of magnitude. Empirical analysis reveals that a LLaMA-7B model with $b = 8$ and $n = 65{,}536$ generates KV cache exceeding 256 GB, which brings critical pressure on memory budgets during long-context processing.

To address this challenge, eviction strategies have been proposed to optimize the KV cache by prioritizing and retaining important tokens for generation. The importance of tokens is typically determined by various configurable schemes, including prior knowledge (e.g., "attention sink" in (Zhang et al., 2023)) and cumulative attention

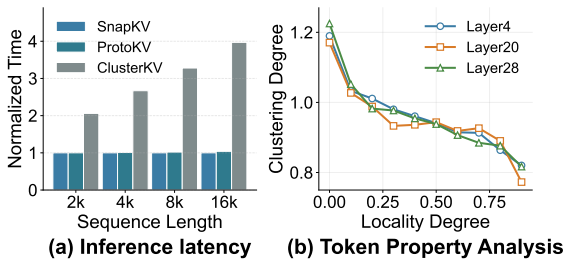


Figure 2: (a) Comparison of inference latency with varying sequence length. (b) Relationship between locality and clustering property on *SAMSum*, with detailed measurement and implementation in Appendix D.

scores (Zhang et al., 2023). Yet these token-wise retention approaches struggle to globally preserve semantic coherence. Recently, clustering-based KV cache methods (Wang et al., 2024; Tang et al., 2024; Wu et al., 2024; Fountas et al., 2024) group tokens into semantic clusters (as illustrated in Figure 1) according to their key vectors and selectively recalls them during inference, but always face computational inefficiency and parallelization bottlenecks. We take ClusterKV (Liu et al., 2024b) as the example and demonstrate its great inference latency in Figure 2(a). This motivates the need for an efficient KV-cache compression algorithm that preserves semantic integrity.

In this paper, we propose a novel KV cache compression method named ProtoKV for long-context inference via hybrid semantic prototype construction. Unlike extant clustering-based strategies that require iterative refinement, ProtoKV eschews complex clustering process by directly extracting semantic prototypes as clustering anchors. Figure 1 demonstrates that our ProtoKV exhibits excellent clustering performance, and Figure 2(a) shows our ProtoKV ensures semantic coherence with negligible additional inference latency compared with SnapKV (Li et al., 2024b), reducing by up to 3.9× compared to ClusterKV (Liu et al., 2024b).

In ProtoKV, we categorize tokens of input sequences into two distinct types: regular tokens, which exhibit strong semantic similarity (evidenced by high cosine similarity between key embeddings) with their contextual neighbors; and irregular tokens, which lack such local semantic coherence but demonstrate precise geometric clustering patterns of key embeddings in the semantic space. We visualize this in Figure 2(b). For regular tokens, we employ positional segmentation to partition them into distinct chunks and construct regular semantic prototypes (RSP) via intra-chunk aggregation. For irregular tokens, we employ Locality-Sensitive Hashing (LSH) to cluster them into buckets, and generate irregular semantic prototypes (ISP) through bucket-wise feature consolidation. After obtaining the hybrid prototypes, importance score of each token is then quantified via sliding window voting and redistributed through intra-cluster mean pooling. Tokens with top scores are retained for answer generation.

Extensive experiments demonstrate that ProtoKV achieves superior semantic preservation efficacy under identical memory budgets compared to existing KV cache baselines. Specifically, ProtoKV delivers SOTA accuracy (2.11% gain) on LongBench under fixed memory constraints, maintains 96.8% retrieval accuracy with 1.6% KV cache retention, and shows complementary benefits when combined with budget allocation methods.

2 Related Work

KV Cache Quantization involves converting high-precision numerical values of KV states into lower-precision formats, primarily through two paradigms: uniform compression for general tokens (Yao et al., 2022; Sheng et al., 2023; Zandieh et al., 2025) and dynamic bit allocation guided by token relevance (Hooper et al., 2024b; Liu et al., 2024e; Kang et al., 2024), with layer-adaptive precision schemes prioritizing semantically rich initial layers (Liu et al., 2024d; Tao et al., 2025). Despite its simplicity, current quantization methods always face accuracy degradation from irreversible information loss during low-bit quantization.

KV Cache Selection focuses on retaining critical key-value pairs while permanently discarding unimportant ones to optimize memory and inference. Two dominant strategies emerge: (1) static methods with prefill-phase token selection (Ge et al., 2024; Li et al., 2024b; Zeng et al., 2024), and (2) dynamic approaches updating cached entries via attention-based metrics or structural patterns during decoding (Xiao et al., 2024; Han et al., 2024; Zhang et al., 2023; Zhao et al., 2024). Recent advancements address persistent eviction challenges through multi-tier caching and asynchronous retrieval (Lee et al., 2024; Tang et al., 2024; Zhang et al., 2024a; Hooper et al., 2024a; Liu et al., 2024a). However, existing solutions fail to efficiently preserve semantic coherence, leading to suboptimal selection decisions.

KV Cache Budget Allocation LLMs' hierarchical layers exhibit distinct information extraction

patterns, motivating adaptive memory allocation across layers/heads. Layer-wise strategies (Cai. et al., 2024; Yang et al., 2024; Huang et al., 2024; Zhang et al., 2024b) prioritize resource distribution by analyzing attention concentration gradients, where lower layers retain uniform contextual signals while higher layers preserve semantic focal points. Head-wise approaches (Feng et al., 2024; Zhang et al., 2024c; Fu et al., 2024b) further enable finer-grained optimization through intra-layer importance differentiation.

3 Preliminary and Motivation

3.1 Problem Formulation

Consider an autoregressive transformer layer with L layers and H attention heads. Let $\mathbf{x}_t \in \mathbb{R}^d$ denote the input token embedding at decoding step t, and $\{\mathbf{K}_{1:t-1}^{(h)}, \mathbf{V}_{1:t-1}^{(h)}\}_{h=1}^{H}$ represent the cached key-value pairs from previous steps for each of the H heads. The attention output $\mathbf{o}_t^{(h)} \in \mathbb{R}^{d_h}$ at step t is computed by:

$$\mathbf{o}_t^{(h)} = \text{Softmax}\left(\frac{\mathbf{q}_t^{(h)}(\mathbf{K}_{1:t}^{(h)})^\top}{\sqrt{d_h}}\right)\mathbf{V}_{1:t}^{(h)} \quad (1)$$

where $\mathbf{q}_t^{(h)} = \mathbf{x}_t\mathbf{W}_q^{(h)}$, $\mathbf{k}_t^{(h)} = \mathbf{x}_t\mathbf{W}_k^{(h)}$, $\mathbf{v}_t^{(h)} = \mathbf{x}_t\mathbf{W}_v^{(h)}$ are query/key/value vectors, with projection matrices $\mathbf{W}_q^{(h)}, \mathbf{W}_k^{(h)}, \mathbf{W}_v^{(h)} \in \mathbb{R}^{d\times d_h}$.

Our objective is to find compressed representations $\{\tilde{\mathbf{K}}_{1:t}^{(h)}, \tilde{\mathbf{V}}_{1:t}^{(h)}\}_{h=1}^{H}$ satisfying:

$$\|\mathbf{o}_t^{(h)} - \tilde{\mathbf{o}}_t^{(h)}\|_2 \le \epsilon,\ \forall h \in [H] \quad (2)$$

where $\tilde{\mathbf{o}}_t^{(h)}$ denotes the approximate output using compressed KV pairs, and ϵ is a pre-defined error tolerance. The compression ratio ρ should satisfy:

$$\max_{h\in[H]}\left\{\frac{\|\tilde{\mathbf{K}}_{1:t}^{(h)}\|_0}{\|\mathbf{K}_{1:t}^{(h)}\|_0}, \frac{\|\tilde{\mathbf{V}}_{1:t}^{(h)}\|_0}{\|\mathbf{V}_{1:t}^{(h)}\|_0}\right\} \le \rho \ll 1 \quad (3)$$

with $\|\cdot\|_0$ counting non-zero elements, while maintaining $\|\tilde{\mathbf{K}}_{1:t}^{(h)}\|_0 = \|\tilde{\mathbf{V}}_{1:t}^{(h)}\|_0$ for computation alignment.

3.2 Selection Strategy

Early KV cache compression methods dynamically determine and retain KV pairs for critical tokens based on their cumulative attention score (Zhang et al., 2023). Recently, SnapKV (Li et al., 2024b) simplifies it through an observation window-based selection mechanism. Given an input sequence of length $L_{\text{prompt}} = L_{\text{prefix}} + L_{\text{obs}}$, where L_{obs} denotes the observation window at the sequence end, the method first computes attention patterns in the final L_{obs} tokens to identify critical positions in the prefix. For each attention head $h \in [H]$, we calculate the attention weight summation across the observation window queries:

$$C^{(h)} = \sum_{i=L_{\text{prefix}}+1}^{L_{\text{prompt}}} \mathbf{W}_{\text{obs}}^{(h)}[i,:] \quad (4)$$

where $\mathbf{W}_{\text{obs}}^{(h)} \in \mathbb{R}^{L_{\text{obs}}\times L_{\text{prefix}}}$ contains the softmax-normalized attention weights for head h in the observation window. The top-k indices $I^{(h)} = \text{Top}_k(C^{(h)}, k)$ are selected where $k = \lfloor \rho L_{\text{prefix}} \rfloor$ with ρ being the target compression ratio, as defined in Equation (3). To preserve contextual continuity, SnapKV applies 1D max-pooling with kernel size κ along the prefix dimension:

$$\tilde{C}^{(h)} = \text{MaxPool}_\kappa(C^{(h)}) \quad (5)$$

The compressed KV representations $\tilde{\mathbf{K}}_{1:t}^{(h)}$ and $\tilde{\mathbf{V}}_{1:t}^{(h)}$ are constructed by concatenating: 1) The clustered features around top-pooled positions $\{\mathbf{k}_i^{(h)}, \mathbf{v}_i^{(h)}\}_{i\in I^{(h)}}$. 2) The full KV pairs from the observation window $\{\mathbf{k}_j^{(h)}, \mathbf{v}_j^{(h)}\}_{j=L_{\text{prefix}}+1}^{L_{\text{prompt}}}$.

3.3 Clustering-based Strategy

The window-based selection strategy in SnapKV was proposed under the assumption that selecting only top-scoring tokens would lead to the loss of semantic coherence. The max-pooling operation (Equation 5) preserves tokens that are semantically similar to the top features, even if their own scores are relatively low. However, it operates on the premise that adjacent tokens typically share similar semantics (measured by key-vector similarity), which fails to hold in many practical scenarios.

To overcome this limitation, clustering-based strategy (Liu et al., 2024b; Hooper et al., 2024a) employs a more robust approach by globally capturing semantically similar tokens through clustering. During inference, it compares query vectors against clustering centroids to identify and load the most semantically relevant keys from the prefix.

However, the clustering-based strategy introduces several critical limitations. First, their iterative clustering process (mostly through K-means algorithm) requires excessive computation. As illustrated in Figure 3, Squeeze Attention (Hooper

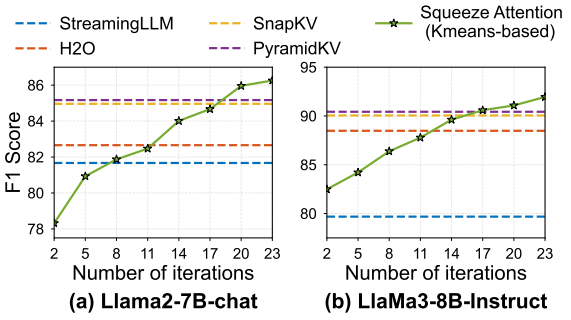


Figure 3: Clustering-based strategy requires 20 iterations before surpassing existing KV cache compression methods on *TriviaQA* dataset.

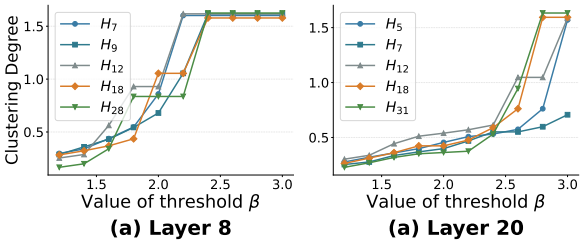


Figure 4: Irregular tokens form progressively compact clustering with the increasing the threshold β on *SAMSum*, with H_i denoting the ith attention head.

et al., 2024a) takes over *20* iterations to outperform baselines and increases compression time remarkably (2.8× on *Multi-Document QA* tasks and 3.2× on *Summarization* tasks). Moreover, such clustering algorithms pose significant challenges to parallelization efficiency. For example, ClusterKV (Liu et al., 2024b) only achieves thread-level parallelism across attention heads. These issues demand a redesigned clustering-based KV retention approach that balances algorithmic performance and computation overhead.

3.4 Observation

In LLMs, we revealed an interesting phenomenon (based on Fig 1 and Fig 2(b)): **tokens that satisfy locality prior are typically scattered, while those that violate locality prior are likely clustered**. First, we provide the formal definition of "locality". For token i with key $\mathbf{k}_i^{(h)}$, we its κ-neighborhood similarity in head h as follows:

$$\mathcal{S}_\kappa^{(h)}(i) = \frac{1}{\kappa} \sum_{j=i-\lfloor \kappa/2 \rfloor}^{i+\lceil \kappa/2 \rceil} \cos(\mathbf{k}_i^{(h)}, \mathbf{k}_j^{(h)}) \quad (6)$$

This metric quantifies the outlier degree of the ith token within the chunk spanning from $i-\lfloor \kappa/2 \rfloor$ to $i+\lceil \kappa/2 \rceil$. Our analysis reveals that despite a high expected value $\mathbb{E}[\mathcal{S}_\kappa^{(h)}(i)]$, a specific $\mathcal{S}_\kappa^{(h)}(i)$ frequently attains low values for a small subset of tokens across different attention heads. We define tokens with low $\mathcal{S}_\kappa^{(h)}(i)$ as violating the locality prior. Interestingly, instead of randomly distributed in the semantic space, these outliers seem to exhibit tight mutual proximity, and in most cases form a single cohesive group.

To quantitatively validate it, we first measure the outlier degree $\Theta(i)$ for ith token as:

$$\Theta(i) = \left(\mathcal{S}_\kappa^{(h)}(i) - \mathbb{E}[\mathcal{S}_\kappa^{(h)}(i)]\right) / \sqrt{\mathbb{V}[\mathcal{S}_\kappa^{(h)}(i)]} \quad (7)$$

By setting a threshold β, we group tokens with $\Theta(i) > \beta$ into cluster $\mathcal{C}$. We then compute its intra-cluster and inter-cluster similarity metrics with:

$$\mathcal{S}_{\text{intra}}(\mathcal{C}) = \frac{1}{|\mathcal{C}|^2} \sum_{i,j \in \mathcal{C}} \cos(\mathbf{k}_i^{(h)}, \mathbf{k}_j^{(h)}), \quad (8)$$

$$\mathcal{S}_{\text{inter}}(\mathcal{C}) = \frac{\sum_{i \in \mathcal{C}, j \notin \mathcal{C}} \cos(\mathbf{k}_i^{(h)}, \mathbf{k}_j^{(h)})}{|\mathcal{C}|(N - |\mathcal{C}|)}, \quad (9)$$

where N denotes the total number of tokens. The clustering degree that measures $\mathcal{C}$'s compactness is then defined as the ratio of $\mathcal{S}_{\text{intra}}/\mathcal{S}_{\text{inter}}$. As shown in Figure 4, increasing β leads to a more compact cluster of $\mathcal{C}$. Notably, when β exceeds a critical threshold, the clustering degree exhibits a sharp increase. We will provide complete experimental results in the Appendix G to validate the pervasiveness of this phenomenon.

Based on this observation, we categorize tokens into two distinct groups: 1) Regular tokens adhering to the spatial locality prior. Their semantics are determined by their positions, so we cluster them based on their positional attributes. 2) Irregular tokens violating locality principles. Despite positional dispersion, they exhibit semantic similarity, which can be clustered efficiently via a local-sensitive hashing approach. The detailed ProtoKV method is presented below.

4 Method

We introduce ProtoKV, which directly obtains semantic prototypes (i.e., cluster centroids) without iterative refinement. ProtoKV captures regular semantic prototypes (RSP) for regular tokens through position-aware chunk aggregation, while employing Locality-Sensitive Hashing (LSH) to derive irregular semantic prototypes (ISP) for irregular tokens. To accelerate compression, we determine token importance using observation window queries and preserve semantic integrity through intra-cluster mean pooling. Pseudo-code for our ProtoKV is presented in Appendix E.

4.1 Chunk-based RSP Consolidation

Given an input sequence of n tokens with corresponding key vectors $\{\mathbf{k}_t\}_{t=1}^{n} \subseteq \mathbb{R}^{d_k}$, we initially partition it into k consecutive chunks $\{\mathcal{C}_m\}_{m=1}^{k}$ of equal length $\lfloor n/k \rfloor$. For each chunk $\mathcal{C}_m$, we compute its mean μ_m and standard deviation σ_m to establish local reference patterns. The irregularity detection metric δ_t for each token within chunk $\mathcal{C}_m$ is computed through the standardized cosine dissimilarity with the chunk center:

$$\delta_t = \frac{1}{\|\sigma_m\|_2}\left(1 - \frac{\mathbf{k}_t^\top \mu_m}{\|\mathbf{k}_t\|_2 \|\mu_m\|_2}\right). \tag{10}$$

Tokens with top-p δ_t values across all chunks are identified as irregular ones $\mathcal{O} = \{\mathbf{k}_j\}_{j=1}^{p}$. After filtering these tokens, we distill each chunk into a compact semantic prototype. Specifically, the m-th regular semantic prototype derives with:

$$c_m^{(\text{regular})} = \frac{\sum_{\mathbf{k}_t \in \mathcal{C}_m \setminus \mathcal{O}} \mathbf{k}_t}{\| \sum_{\mathbf{k}_t \in \mathcal{C}_m \setminus \mathcal{O}} \mathbf{k}_t \|_2}. \tag{11}$$

4.2 LSH-guided ISP Consolidation

These identified irregular tokens are then allocated into u hash buckets. Specifically, the key vector for jth token is projected into low-dimensional space using Random Fourier Features (RFF) mapping $\phi : \mathbb{R}^{d_k} \to \mathbb{R}^r$ with Gaussian kernel approximation:

$$\phi(\mathbf{k}_j) = \sqrt{\frac{2}{r}} \cos\left(\mathbf{W}\mathbf{k}_j + \mathbf{b}\right), \tag{12}$$

where projection matrix $\mathbf{W} \sim \mathcal{N}(0, \gamma^2 I)$ and phase shift $\mathbf{b} \sim \text{Uniform}(0, 2\pi)$. The real-valued projections are then binarized to $\{0, 1\}$ codes:

$$\mathbf{h}_j = \mathbb{I}\left(\phi(\mathbf{k}_j) > 0\right) \in \{0, 1\}^r \tag{13}$$

with $\mathbb{I}(\cdot)$ denoting the element-wise indicator function. Each binary code $\mathbf{h}_j$ is subsequently interpreted as an r-bit integer for bucket allocation:

$$\mathcal{H}(\mathbf{k}_j) = \left(\sum_{i=1}^{r} 2^{r-i} h_j^{(i)}\right) \bmod u, \tag{14}$$

where $h_j^{(i)}$ denotes the i-th bit of $\mathbf{h}_j$, and u represents the total number of hash buckets. To ensure clustering effectiveness, we typically require that $u = 2^r$. For example, a 3-bit code $[1, 0, 0]$ converts to decimal 4 and would be assigned to bucket $\mathcal{B}_4$. This binary-to-decimal conversion preserves the Hamming distance between original codes while enabling efficient bucket indexing.

Following the hashing-based bucketing operation, we generate u irregular semantic prototypes. The s-th irregular semantic prototype aggregates bucket-specific directional patterns with:

$$c_s^{(\text{irregular})} = \frac{\sum_{\mathbf{k}_j \in \mathcal{B}_s} \mathbf{k}_j}{\| \sum_{\mathbf{k}_j \in \mathcal{B}_s} \mathbf{k}_j \|_2} \tag{15}$$

4.3 Attention Contribution Redistribution

The hybrid semantic prototypes $\mathcal{M}$ contains n elements, $\mathcal{M} = \{c_m\}_{m=1}^{n}$, which is obtained via:

$$\mathcal{M} = \{c_m^{(\text{regular})}\}_{m=1}^{k} \cup \{c_s^{(\text{irregular})}\}_{s=1}^{r} \tag{16}$$

with both components emerge from Equation (11) and Equation (15) respectively. Based on these semantic prototypes, we perform token clustering. Specifically, the cluster assignment for token $\mathbf{k}_t$ is determined by the semantic prototype with the highest cosine similarity to it:

$$c(\mathbf{k}_t) = \arg\max_{c \in \mathcal{M}} \frac{\mathbf{k}_t^\top c}{\|\mathbf{k}_t\|_2 \|c\|_2}. \tag{17}$$

After dividing all tokens into $\{\mathcal{C}_j\}_{j=1}^{n}$, we calculate the attention weight summation across the observation window queries with Equation (4) to get $C^{(h)}$, and redistribute it through intra-cluster mean pooling:

$$\hat{C}^{(h)}(i) = \frac{1}{|\mathcal{C}_j|} \sum_{i \in \mathcal{C}_j} C^{(h)}(i). \tag{18}$$

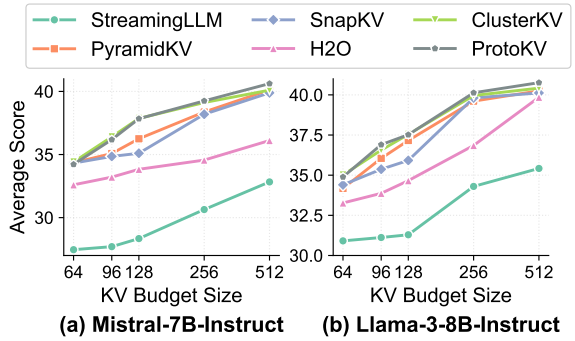
Given budget size $\mathcal{B}$, tokens with top $\mathcal{B} - L_{\text{obs}}$ importance scores $\hat{C}^{(h)}$ are retained for inference generation in head h, along with L_{obs} tokens in observation windows.

4.4 Efficiency Analysis

The efficiency of our algorithm is reflected in three aspects. Firstly, while other clustering-based strategies (Liu et al., 2024b; Hooper et al., 2024a) requires computational complexity for clustering with $O(n * k)$, our method achieves efficient prototype consolidation with complexity $O(n + p)$; Secondly, our approach employs the static token retention strategy, avoiding dynamic token retrieval that requires frequent interaction with CPU memory. Building on these two advantages, our ProtoKV also achieves efficient matrix parallelism, enabling our algorithm to process multiple attention heads concurrently.

Model	Size	Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
			NrtvQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PRe	Lcc	RB-P	
			18409	3619	4559	9151	4887	11214	8734	10614	2113	5177	8209	6258	11141	9289	1235	4206	–
LlaMA-3-8B -Instruct	–	FullKV	25.16	32.29	40.43	45.35	37.04	23.84	28.62	23.34	26.33	75.00	90.23	42.65	5.10	70.0	59.41	55.60	42.34
	128	SLM	17.47	8.55	21.31	32.86	26.28	15.54	17.91	20.42	20.16	45.00	73.36	30.78	5.75	68.50	48.38	49.31	31.29
		H2O	21.58	12.54	28.57	**39.86**	28.62	18.88	**20.23**	22.16	20.14	35.50	86.62	**39.19**	5.83	**69.50**	54.46	50.81	34.66
		SnapKV	**22.35**	16.00	31.52	36.82	28.39	**19.49**	19.06	21.36	20.07	50.00	87.74	38.94	5.75	68.00	57.42	51.84	35.92
		Pyramid	21.80	16.65	30.73	38.48	28.80	19.26	19.92	22.06	20.12	**66.50**	88.95	38.20	**5.92**	68.00	57.88	51.54	37.16
		ProtoKV	22.26	**17.05**	**31.84**	39.68	**29.28**	19.35	19.83	**22.31**	**20.82**	62.00	**89.35**	38.74	5.37	69.00	**58.84**	**54.61**	**37.52**
	256	SLM	17.98	11.09	23.85	37.83	29.97	16.02	20.30	20.94	24.56	52.00	79.68	34.82	**5.83**	**69.50**	54.84	50.46	34.30
		H2O	**23.67**	16.85	32.70	41.57	31.08	18.91	22.28	22.81	23.69	41.00	90.36	**40.19**	5.54	**69.50**	57.52	52.16	36.85
		SnapKV	23.32	**20.31**	**37.35**	42.70	31.08	20.47	22.63	23.04	23.93	71.00	90.39	39.78	5.50	**69.50**	60.27	**55.62**	39.81
		Pyramid	23.46	18.76	35.06	42.33	31.56	20.73	**23.37**	**23.11**	**24.37**	**72.00**	90.43	39.54	5.50	**69.50**	59.25	54.87	39.61
		ProtoKV	23.58	19.92	36.38	**43.72**	**32.29**	**20.89**	23.25	22.98	23.42	70.00	**90.81**	40.07	5.80	**69.50**	**61.22**	55.49	**39.95**
Mistral-7B -Instruct	–	FullKV	25.07	32.92	49.34	39.77	27.32	16.83	32.87	24.24	27.10	70.00	86.57	43.30	2.75	59.25	56.86	50.48	40.29
	128	SLM	17.76	13.46	35.11	27.25	22.29	9.80	18.26	19.02	19.16	43.50	74.12	36.50	2.67	27.17	43.65	43.79	28.34
		H2O	19.99	20.34	38.60	28.50	21.63	12.88	20.65	22.61	22.08	53.00	81.29	39.75	2.20	**75.38**	49.54	44.27	33.83
		SnapKV	22.14	21.14	42.98	32.96	22.12	14.12	19.19	21.89	21.01	64.00	83.77	39.92	2.51	66.50	51.81	46.51	35.84
		Pyramid	22.32	22.52	43.65	33.07	22.45	**15.72**	20.56	22.52	21.36	64.00	83.84	40.43	2.74	67.95	51.64	46.47	36.29
		ProtoKV	**23.11**	**23.70**	**44.89**	**36.12**	**22.88**	15.57	**21.63**	**23.75**	**22.49**	**67.50**	**84.89**	**41.96**	**3.10**	72.30	**53.54**	**47.95**	**37.84**
	256	SLM	19.26	17.78	36.82	27.74	22.78	10.53	24.47	19.84	**25.48**	51.00	76.39	40.24	2.50	31.92	46.15	45.56	31.14
		H2O	22.35	23.22	41.76	30.76	22.88	14.03	23.53	22.96	24.53	53.50	83.82	41.08	1.66	**78.49**	50.77	46.70	36.39
		SnapKV	23.08	25.95	48.04	34.79	24.75	14.41	24.14	**23.69**	24.47	67.50	85.64	41.51	1.95	68.11	53.74	49.31	38.19
		Pyramid	23.49	**26.39**	48.22	**35.23**	25.51	13.65	**24.79**	23.52	24.49	68.50	85.43	41.58	2.33	69.07	53.45	48.23	38.37
		ProtoKV	**23.76**	26.02	**48.82**	34.96	**26.32**	**14.66**	24.69	23.62	24.91	**70.50**	**86.02**	**42.76**	**2.90**	71.40	**55.62**	**51.14**	**39.25**
Llama2-7B -chat	–	Full	14.82	9.5	22.76	7.35	10.71	9.23	25.63	23.79	26.51	65.00	89.16	34.28	2.50	9.50	68.24	61.83	29.72
	128	SLM	10.12	4.94	15.8	5.93	9.07	3.05	18.07	19.30	18.30	42.50	76.97	24.18	**2.00**	3.11	61.47	44.26	22.93
		H2O	13.22	4.55	16.28	6.58	9.01	3.82	**20.92**	**21.86**	18.44	40.00	79.40	27.85	1.20	7.38	55.75	53.36	24.09
		SnapKV	13.13	5.84	21.62	7.12	9.19	3.90	18.91	21.41	18.21	45.00	84.12	27.85	1.60	7.02	61.48	54.87	25.62
		Pyramid	13.78	5.75	**22.37**	7.62	**9.68**	3.96	19.24	20.47	18.18	59.00	84.38	29.42	1.50	**8.22**	62.24	54.51	26.45
		ProtoKV	**13.97**	**5.94**	22.08	**7.76**	9.29	**4.09**	20.85	21.60	**19.02**	**59.50**	**84.69**	**29.99**	1.50	8.18	**63.22**	**56.97**	**26.94**
	256	SLM	12.74	4.94	15.8	5.93	9.12	3.48	**25.70**	19.31	**24.87**	54.00	81.67	31.47	2.00	4.38	61.87	52.20	25.60
		H2O	14.55	5.95	18.67	6.42	8.67	4.17	23.69	22.07	22.72	56.00	82.66	30.48	2.50	**8.89**	58.83	56.83	26.45
		SnapKV	17.12	6.75	**21.52**	7.38	10.03	4.12	24.56	22.39	23.07	63.00	84.96	31.54	1.52	7.25	64.94	56.88	28.01
		Pyramid	**17.84**	7.28	20.37	7.14	10.47	4.29	23.59	22.30	22.41	64.00	85.17	32.72	**2.67**	8.23	65.75	57.50	28.30
		ProtoKV	17.29	**7.34**	20.94	**7.58**	**11.43**	**4.80**	24.73	**22.57**	22.89	**68.00**	**86.78**	**34.51**	1.68	7.67	**66.88**	**60.34**	**29.11**

Table 1: Performance comparison on the **LongBench** dataset for full KV cache, extant KV baselines (including StreamingLLM, H2O, SnapKV, PyramidKV) and our ProtoKV. **Bold** indicates the best performance and underline the second performance.



(a) Mistral-7B-Instruct (b) Llama-3-8B-Instruct

Figure 5: Experimental results on LongBench dataset under different KV cache budget conditions. The final experimental results are the average score.

5 Experiment

5.1 Implementation

Dataset We use *LongBench* (Bai et al., 2024) dataset to assess the performance of ProtoKV on tasks involving long-context inputs. The dataset comprises 14 English tasks and 2 code-related tasks. The majority of these tasks have an average length ranging from 5k to 15k tokens, with a total of approximately 3,750 test samples. A detailed description of *LongBench* dataset is provided in the Appendix A.

Baseline We benchmark our method against StreamingLLM (Xiao et al., 2024), H2O (Zhang et al., 2024d), SnapKV (Li et al., 2024b), PyramidKV (Cai. et al., 2024). We use state-of-the-art open-sourced LLMs include the Llama family (Llama-2-7B-chat, LlaMA-3-8B-instruct) and Mistral-7B-Instruct-v0.2, which can handle up to 32k context length. Detailed description of these three LLMs is provided in the Appendix C.

Experiment Setup We maintain identical average KV cache sizes across baseline to ensure fair memory comparison. All experiments use two NVIDIA 3090 GPUs (48GB total) with consistent prompts across datasets. Our configuration balances experimental uniformity with task-specific optimizations. We use the same prompt for each dataset in all the experiments.

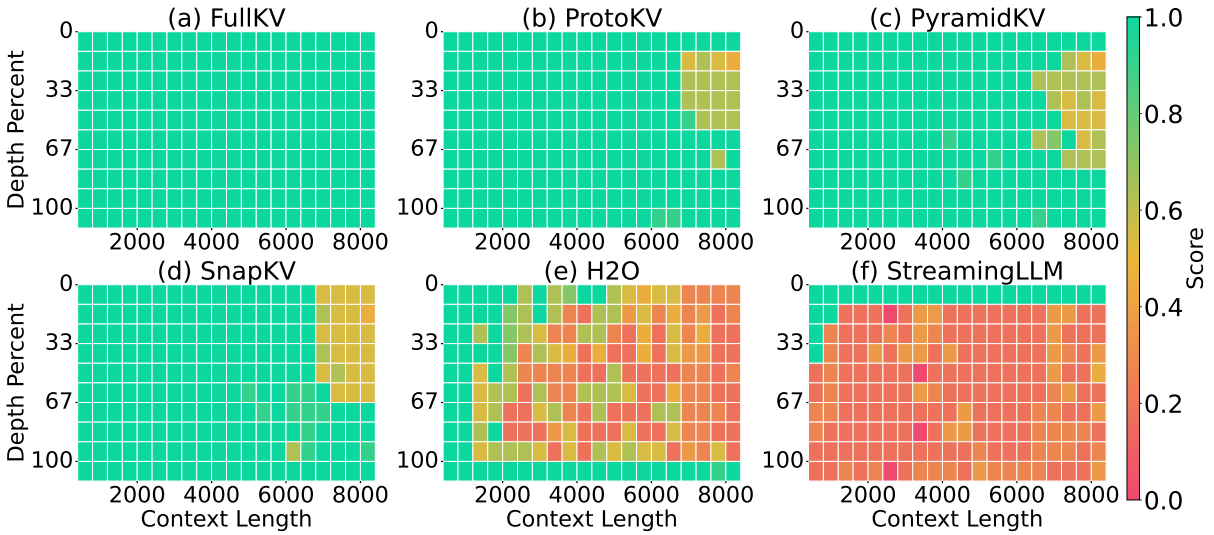


Figure 6: Results of the **Needle In A HayStack** experiment, where LLMs are required to retrieve a target sentence ("needle") **inserted** in long documents. The x-axis represents the context length while y-axis the depth where the needle is inserted. E.g., context length of 4000 and depth of 11.0 implies that the needle is inserted at location 4000 × 11% = 440 in the sequence. The color indicates retrieval accuracy, the greener, the better.

5.2 Result Analysis

Table 1 and Figure 5 demonstrate the experimental results on *LongBench* across diverse KV cache configurations. Generally, our method maintains the best performance between 64-512 budgets, with an average improvement of 2.11%. For LlaMA-3-8B-Insturct and Mistral-7B (Figure 5), ProtoKV outperforms Sota baselines by 0.35% to 4.27% across diverse budget sizes. We also compare ProtoKV with the clustering-based compression method ClusterKV (Liu et al., 2024b), and ProtoKV outperforms it under relaxed memory constraints (256 and 512 budget size).

Task-specific experimental results are reported in Table 1. As presented, our method demonstrates consistent performance advantages over diverse long-context tasks. Notably, with a cache size of 256, ProtoKV attains 39.95 (+0.35% over SnapKV) for LlaMA-3-8B-Instruct, 39.25 (+2.29% over PyramidKV) for Mistral-7B-Instruct, and 29.11 (+2.86% over PyramidKV) for Llama2-7B-chat, showing robust generalization capabilities. The improvements are particularly pronounced in challenging multi-document QA tasks (e.g., +2.38% over SnapKV on HotpotQA for LlaMA-3-8B-256) and code-related tasks (e.g., +3.73% over SnapKV on RepoBench-P for Mistral-7B-256). Moreover, ProtoKV maintains competitive performance compared to the FullKV baseline while using merely 2.6%-3.9% of the original KV cache size, demonstrating its effectiveness in resource-constrained scenarios. These results validate our method's ability to preserve critical attention patterns through context-aware retention while minimizing information loss.

Tasks	*ProtoKV*	+ *LA.*	+ *HA.*
SDQA	32.87	33.12$_{\uparrow(0.54\%)}$	**33.58**$_{\uparrow(1.02\%)}$
MDQA	25.31	25.72$_{\uparrow(1.46\%)}$	**26.14**$_{\uparrow(1.24\%)}$
SUM	24.41	24.89$_{\uparrow(1.47\%)}$	**25.29**$_{\uparrow(1.15\%)}$
Few shot	66.43	66.85$_{\uparrow(1.21\%)}$	**67.34**$_{\uparrow(1.65\%)}$
SYN	37.15	37.43$_{\uparrow(0.82\%)}$	**37.91**$_{\uparrow(1.73\%)}$
Code	59.38	59.77$_{\uparrow(4.08\%)}$	**60.16**$_{\uparrow(3.27\%)}$

Table 2: **Compatibility Analysis** on Mistral-7B-Instruct with KV budget size of 256, with ↑ (·) denoting the improvement compared with SnapKV+*LA./HA.*.

5.3 Needle In A Haystack

We conduct the "Fact Retrieval Across Context Lengths" (Needle In A Haystack) experiment (Liu et al., 2024c; Fu et al., 2024a) using LlaMA-3-8B-Instruct with 8K context length to evaluate in-context retrieval capabilities of LLMs. This task requires precise information retrieval from extensive contexts, simulating real-world scenarios where relevant data is buried among vast irrelevant information.We compare other KV Cache techniques at a consistent cache budget size of 128 (a retention ratio of 1.6%). Results in Fig 6 indicate that StreamingLLM and H2O almost collapses on retrieval task. In contrast, SnapKV achieves 94.2% accuracy, PyramidKV reaches 95.5%, and our ProtoKV attains 96.8% accuracy, which closely matches the fully cached case. These results show that our proposed method effectively maintains re-

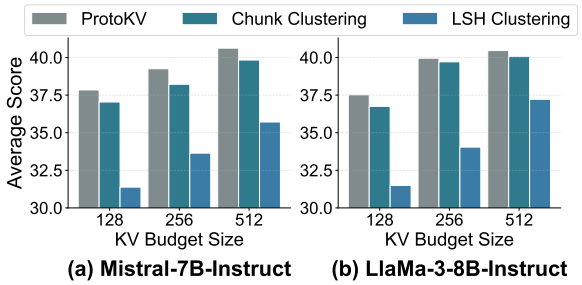


Figure 7: **Ablation Study** for different KV budget sizes.

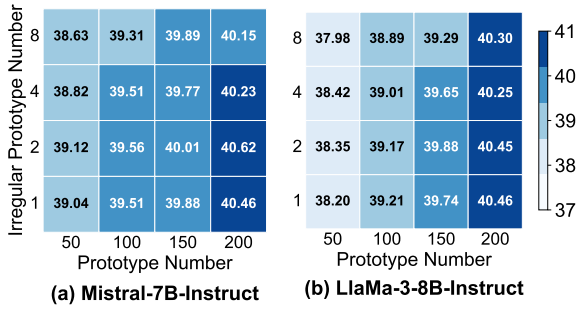


Figure 8: **Hyperparameter Analysis** for irregular clustering size (vertical axis) and total clustering size (horizontal axis). We report the averaged score on LongBench.

trieval performance across long contexts (up to 8K tokens) with minimal performance degradation.

5.4 Compatibility Analysis

KV cache *selection* and *budget allocation* represent orthogonal optimization directions, and we aim to investigate the compatibility between our method and the budget allocation methods. Specifically, we study two prominent allocation strategies: *Layer-wise Allocation (LA.)* (Nawrot et al., 2024) and *Head-wise Allocation (HA.)* (Feng et al., 2024), which dynamically distribute token budgets across attention different layers/attention heads. As demonstrated in Table 2, ProtoKV exhibits strong compatibility with these techniques by achieving additional compression effectiveness.

5.5 Ablation Study

We conduct ablation experiments to verify whether dividing tokens into regular/irregular ones and processing them separately are reasonable. Specifically, we compare ProtoKV with its two variants: *chunk-clustering* that only uses chunk-based aggregation to obtain semantic prototypes, and *LSH-clustering* that buckets and groups all tokens via LSH. Two variants adopt the same number of semantic prototypes (i.e., cluster count) as ProtoKV. Figure 7 shows both variants reduce KV cache compression performance, especially LSH-clustering, showcasing the validity of ProtoKV.

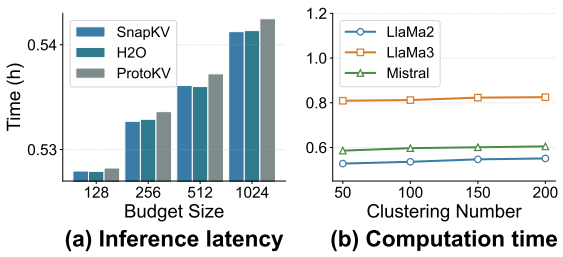


Figure 9: (LEFT) Comparison of inference latency with varying budget size. (RIGHT) ProtoKV's inference latency with different clustering number. Vertical axis denotes average time consumed for LongBench dataset.

Hyperparameter Analysis. We primarily investigate the impact of the irrgular semantic prototype number u and the regular semantic prototype number k on model performance. Experimental results in Figure 8 reveal that compression performance generally improves with increasing total prototype size, whereas the impact of u is less pronounced. In practice, we set detected irregular tokens number p to a small value as $3u$ to ensure that no position-determined tokens are included. Though this may exclude many tokens that violate locality assumptions, these selected tokens sufficiently represent viable irregular patterns.

5.6 Inference Latency Analysis

We investigate the inference latency of our proposed ProtoKV for KV cache compression. As illustrated in Figure 9, our method maintains comparable inference latency to most commonly used approaches like H2O and SnapKV. However, these approaches cannot globally retain semantic coherence, while ours can. Figure 2(a) shows our ProtoKV reduce inference latency by up to 3.9× compared to ClusterKV (Liu et al., 2024b). Furthermore, we find that different cluster numbers have minimal impact on ProtoKV's inference latency.

6 Conclusion

While existing clustering-based KV cache compression methods achieve impressive compression performance by globally retaining semantic coherence, they suffer from time-consuming clustering algorithms that are difficult to parallelize. To address this, we propose ProtoKV, a simplified algorithm that significantly improves time efficiency by identifying semantic prototypes before performing clustering. Our approach builds upon the phenomenon that tokens that satisfy locality prior are typically scattered, while those that violate locality prior are likely clustered.

Limitation

Our current understanding of the positional-semantic dichotomy phenomenon remains primarily at the statistical observation level. Several fundamental questions remain open, including why this phenomenon exhibits such patterns and whether it generalizes beyond the LongBench datasets or LLMs with more parameters. Furthermore, when the number of clusters is small, our clustering method still shows non-negligible performance gaps compared to existing clustering-based KV cache compression approaches. The underlying reasons for these gaps and potential improvement strategies warrant further investigation.

Ethical Considerations

The primary objective of this paper is to provide a KV cache compression framework that designed to accelerate inference. By eliminating the need for iteration refinements of cluster-based method, ProtoKV improves the memory usage and reduces latency. This work is based on the publicly available LongBench dataset, which predominantly contains English text, and the associated questions-answer pairs are also in English. We comply with all dataset licenses, and confirm the content contains neither private nor offensive information. We utilized Claude-3.7-Sonnet to assist us with code generation.

References

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3119–3137. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Zefan Cai., Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. 2024. PyramidKV: Dynamic KV Cache Compression based on Pyramidal Information Funneling. *arXiv preprint*. ArXiv:2406.02069 [cs].

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. 2024. Ada-KV: Optimizing KV Cache Eviction by Adaptive Budget Allocation for Efficient LLM Inference. *arXiv preprint*. ArXiv:2407.11550 [cs].

Zafeirios Fountas, Martin A Benfeghoul, Adnan Oomerjee, Fenia Christopoulou, Gerasimos Lampouras, Haitham Bou-Ammar, and Jun Wang. 2024. Human-like episodic memory for infinite context llms. *arXiv preprint arXiv:2407.09450*.

Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. 2024a. Data engineering for scaling language models to 128k context. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. 2024b. Not All Heads Matter: A Head-Level KV Cache Compression Method with Integrated Retrieval and Reasoning. *arXiv preprint*. ArXiv:2410.19258 [cs].

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. 2024. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *The Twelfth International Conference on Learning Representations*.

Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2024. LM-infinite: Zero-shot extreme length generalization for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3991–4008, Mexico City, Mexico. Association for Computational Linguistics.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Monishwaran Maheswaran, June Paik, Michael W Mahoney, Kurt Keutzer, and Amir Gholami. 2024a. Squeezed attention: Accelerating long context length llm inference. *arXiv preprint arXiv:2411.09688*.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. 2024b. Kvquant: Towards 10 million context length LLM inference with KV cache quantization. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024,*

NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024.

Jiawei Huang, Meiting Xue, Chenpu Li, Huan Zhang, and Bei Zhao. 2024. Dynamickv: Data storage strategy based on partition merging of log-structured merge tree. In *2nd International Conference on Computer, Vision and Intelligent Technology, ICCVIT 2024, Huaibei, China, November 24-27, 2024*, pages 1–6. IEEE.

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. 2024. GEAR: An Efficient KV Cache Compression Recipe for Near-Lossless Generative Inference of LLM. *arXiv preprint*. ArXiv:2403.05527 [cs].

Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. 2024. InfiniGen: Efficient generative inference of large language models with dynamic KV cache management. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 155–172, Santa Clara, CA. USENIX Association.

Jianan Li, Quan Tu, Cunli Mao, Zhengtao Yu, Ji-Rong Wen, and Rui Yan. 2024a. Streamingdialogue: Prolonged dialogue learning via long context compression with minimal losses. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024b. Snapkv: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, and 1 others. 2024a. Retrievalattention: Accelerating long-context llm inference via vector retrieval. *arXiv preprint arXiv:2409.10516*.

Guangda Liu, Chengwei Li, Jieru Zhao, Chenqi Zhang, and Minyi Guo. 2024b. ClusterKV: Manipulating LLM KV Cache in Semantic Space for Recallable Compression. *arXiv preprint*. ArXiv:2412.03213 [cs].

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024c. Lost in the middle: How language models use long contexts. *Trans. Assoc. Comput. Linguistics*, 12:157–173.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, Michael Maire, Henry Hoffmann, Ari Holtzman, and Junchen Jiang. 2024d. Cachegen: KV cache compression and streaming for fast large language model serving. In *Proceedings of the ACM SIGCOMM 2024 Conference, ACM SIGCOMM 2024, Sydney, NSW, Australia, August 4-8, 2024*, pages 38–56. ACM.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024e. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Piotr Nawrot, Adrian Lancucki, Marcin Chochowski, David Tarjan, and Edoardo M. Ponti. 2024. Dynamic memory compression: Retrofitting llms for accelerated inference. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. Flexgen: High-throughput generative inference of large language models with a single GPU. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 31094–31116. PMLR.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. 2024. QUEST: query-aware sparsity for efficient long-context LLM inference. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Qian Tao, Wenyuan Yu, and Jingren Zhou. 2025. Asymkv: Enabling 1-bit quantization of KV cache with layer-wise asymmetric quantization configurations. In *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pages 2316–2328. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Zheng Wang, Boxiao Jin, Zhongzhi Yu, and Minjia Zhang. 2024. Model tells you where to merge: Adaptive kv cache merging for llms on long-context tasks. *arXiv preprint arXiv:2407.08454*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. 2024. Retrieval Head Mechanistically Explains Long-Context Factuality. *arXiv preprint*. ArXiv:2404.15574 [cs].

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Dongjie Yang, Xiaodong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. 2024. Pyramidinfer: Pyramid KV cache compression for high-throughput LLM inference. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 3258–3270. Association for Computational Linguistics.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183.

Amir Zandieh, Majid Daliri, and Insu Han. 2025. Qjl: 1-bit quantized jl transform for kv cache quantization with zero overhead. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25805–25813.

Zihao Zeng, Bokai Lin, Tianqi Hou, Hao Zhang, and Zhijie Deng. 2024. In-context KV-Cache Eviction for LLMs via Attention-Gate. *arXiv preprint*. ArXiv:2410.12876 [cs].

Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. 2024a. Pqcache: Product quantization-based kvcache for long context llm inference. *CoRR*, abs/2407.12820.

Xuan Zhang, Cunxiao Du, Chao Du, Tianyu Pang, Wei Gao, and Min Lin. 2024b. Simlayerkv: A simple framework for layer-level KV cache reduction. *CoRR*, abs/2410.13846.

Yanqi Zhang, Yuwei Hu, Runyuan Zhao, John Lui, and Haibo Chen. 2024c. Unifying kv cache compression for large language models with leankv. *arXiv preprint arXiv:2412.03131*.

Zhenyu Zhang, Shiwei Liu, Runjin Chen, Bhavya Kailkhura, Beidi Chen, and Atlas Wang. 2024d. Q-hitter: A better token oracle for efficient LLM inference via sparse-quantized KV cache. In *Proceedings of the Seventh Annual Conference on Machine Learning and Systems, MLSys 2024, Santa Clara, CA, USA, May 13-16, 2024*. mlsys.org.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark W. Barrett, Zhangyang Wang, and Beidi Chen. 2023. H2O: heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Junqi Zhao, Zhijin Fang, Shu Li, Shaohui Yang, and Shichao He. 2024. BUZZ: Beehive-structured Sparse KV Cache with Segmented Heavy Hitters for Efficient LLM Inference. *arXiv preprint*. ArXiv:2410.23079 [cs].

A LongBench Dataset Details

Dataset LongBench is a large-scale benchmark dataset designed for evaluating language models' capabilities in understanding and generating long texts. It covers various types of tasks including, but not limited to, Single-Document Question Answering (QA), Multi-Document QA, Summarization, Few-shot Learning, and Synthetic tasks. The aim is to comprehensively assess models across different application scenarios.

Here are some specific tasks included in the LongBench dataset along with their characteristics:

NarrativeQA: Focuses on understanding narrative texts, requiring models to read and answer questions about stories or narratives.

Qasper: Involves asking and answering questions based on academic articles, testing the model's ability to understand scholarly literature.

MultiFieldQA-en: Covers QA tasks across multiple fields, enhancing the model's capability to understand texts from diverse domains.

HotpotQA, 2WikiMultihopQA, MuSiQue: These tasks emphasize reasoning and information integration across multiple documents, challenging the model's ability to find answers in a multi-document environment.

GovReport, QMSum, MultiNews: Concentrate on extracting key information and generating summaries from lengthy texts, assessing the model's summarization capability.

TREC, TriviaQA, SAMSum: Evaluate the model's learning ability and domain-specific knowledge acquisition through few-shot examples.

PassageCount, PassageRetrieval-en: Synthetic tasks designed to test the model's performance under specific conditions, such as document counting or retrieval accuracy.

LCC, RepoBench-P: Involve code understanding and evaluation of editing similarity, catering to the unique requirements of programming languages.

Each task comes with its own set of evaluation metrics (e.g., F1 Score, Rouge-L, Accuracy) to quantify model performance. Moreover, LongBench includes texts from different languages and domains, ensuring broad applicability and linguistic diversity of the models. This dataset plays a crucial role in advancing the field of natural language processing, especially in improving models' abilities to handle long texts. Detailed information is demonstrated in Table 3.

B Eval Mertic

F1 Score is the harmonic mean of Precision and Recall. It is particularly useful when dealing with imbalanced datasets. The formula is given by:

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{19}$$

Where P is the proportion of true positive predictions among all positive predictions and R is proportion of true positive predictions among all actual positive instances.

ROUGE-L measures the similarity between generated text and reference text based on the longest common subsequence (LCS). It takes into account both n-gram co-occurrence and word order. The score is calculated as:

$$ROUGE\text{-}L = \frac{LCS(X, Y)}{\max(|X|, |Y|)} \tag{20}$$

Where X is generated output, Y is reference text and $LCS(X, Y)$ is the length of the longest common subsequence between X and Y.

Edit Sim measures the similarity between two sequences based on the minimum number of edit operations (insertions, deletions, substitutions) required to transform one sequence into another. It is often normalized by the length of the longer string:

$$\text{Edit Sim} = 1 - \frac{LD(X, Y)}{\max(|X|, |Y|)} \tag{21}$$

X, Y are two input strings and LD is Minimum number of single-character edits needed to convert X to Y.

Accuracy is a basic evaluation metric that measures the proportion of correct predictions (both true positives and true negatives) among the total number of cases examined.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{22}$$

TP is when the model correctly predicts a positive class, TN is when it correctly predicts a negative class, FP is when it incorrectly predicts a positive class, and FN is when it incorrectly predicts a negative class.

C LLM Model Details

We introduce the three LLMs used in this paper. Detailed Statistics for them are shown in Table 4.

Meta-Llama-3-8B-Instruct is an 8B-parameter instruction-tuned variant of LLaMA-3, optimized

Task	Task Type	Source	Eval metric	Avg len	Language	License
NarrativeQA	Single-Doc. QA	Literature, Film	F1	18,409	EN	MIT License
Qasper	Single-Doc. QA	Science	F1	3,619	EN	MIT License
MultiFieldQA-en	Single-Doc. QA	Multi-field	F1	4,559	EN	MIT License
HotpotQA	Multi-Doc. QA	Wikipedia	F1	9,151	EN	MIT License
2WikiMultihopQA	Multi-Doc. QA	Wikipedia	F1	4,887	EN	MIT License
MuSiQue	Multi-Doc. QA	Wikipedia	F1	11,214	EN	MIT License
GovReport	Summarization	Government report	Rouge-L	8,734	EN	MIT License
QMSum	Summarization	Meeting	Rouge-L	10,614	EN	MIT License
MultiNews	Summarization	News	Rouge-L	2,113	EN	MIT License
TREC	Few shot	Web question	Accuracy	5,177	EN	MIT License
TriviaQA	Few shot	Wikipedia, Web	F1	8,209	EN	MIT License
SAMSum	Few shot	Dialogue	Rouge-L	6,258	EN	MIT License
PassageCount	Synthetic	Wikipedia	Accuracy	11,141	EN	MIT License
PassageRetrieval-en	Synthetic	Wikipedia	Accuracy	9,289	EN	MIT License
LCC	Code	Github	Edit Sim	1,235	Python/C#/Java	MIT License
RepoBench-P	Code	Github	Edit Sim	4,206	Python/Java	MIT License

Table 3: An overview of the dataset statistics in LongBench.

Configuration	**LlaMA-3-8B-Instruct**	**Mistral-7B-Instruct-v0.2**	**Llama2-7B-chat**
Hidden Size	4,096	4,096	4,096
Layers	32	32	32
Q Heads	32	32	32
KV Heads	8	8	32
Attention Heads	32	32	32
Max Position Embeddings	8,192	32,768	4,096
Intermediate Size	14,336	14,336	11,008
Vocabulary Size	128,256	32,000	32,000

Table 4: Configuration of Models.

for dialogue tasks. Using transformer architecture with SFT and RLHF, it features a 128K vocabulary and GQA for efficiency. The model supports 8K-context (extendable to 128K) and demonstrates strong performance in text generation and reasoning tasks.

Mistral-7B-Instruct-v0.2 is a 7.3B-parameter instruction-tuned model by Mistral AI, featuring 32K context length via optimized RoPE embeddings. With grouped-query attention for efficiency, it excels in conversational and coding tasks while supporting GGUF quantization. Benchmarks show it outperforms comparable 7B models, particularly in code generation.

Llama2-7B-chat is Meta's 7 billion parameter chat-optimized language model, fine-tuned for dialogue applications using RLHF. The model features a 4K token context window and demonstrates improved safety and helpfulness compared to its base version. It achieves strong performance in conversational tasks while maintaining efficient inference through optimized transformer architecture.

D Locality & Clustering Degree

In the main text, we introduced the relationship between locality degree and clustering degree, but did not explicitly specify their quantitative measures. We hereby provide the details:

D.1 Locality Degree

The locality of a token is quantified via κ-neighborhood similarity as introduced in Equation 6. In experiments, we set $\kappa = 5$ to capture local contextual interactions. We rank all tokens by their 5-neighborhood similarity $\mathcal{S}_5^{(h)}(i)$ in ascending order and partition them into 10 equal-frequency bins (deciles). Let $\mathcal{G}_k$ denote the k-th bin ($k = 1, \ldots, 10$), where $\mathcal{G}_1$ contains tokens with the lowest 10% similarity scores and $\mathcal{G}_{10}$ the highest 10%. The locality of a specific $\mathcal{G}_k$ is defined by the average 5-neighborhood similarity of the tokens belong to it.

D.2 Clustering Degree

For each bin $\mathcal{G}_k$, we compute its clustering degree with the ratio of intra-group cohesion to inter-

group separation according to Equation 8. The finally reported Locality Degree and Clustering Degree is averaged across different attention heads in a specific layer.

E Pseudocode

We present the pseudo-code for our ProtoKV as follows, it is worth noting that all loop statements in this code can be executed in parallel, significantly reducing inference latency.

Algorithm 1 Algorithm of ProtoKV

Require: Key vectors $\{k_t\}_{t=1}^{n}$, chunk size k, hash bits r, buckets u, irregular number p, observation window size L_{obs}

Ensure: Compressed KV cache $\{\tilde{K}^{(h)}, \tilde{V}^{(h)}\}_{h=1}^{H}$

for $m = 1$ **to** k **do**
  $\mathcal{C}_m \leftarrow \{k_{(m-1)\lfloor n/k \rfloor+1}, ..., k_{m\lfloor n/k \rfloor}\}$
  $\mu_m \leftarrow \frac{1}{|\mathcal{C}_m|}\sum_{k_t \in \mathcal{C}_m} k_t$
  $\sigma_m \leftarrow \sqrt{\frac{1}{|\mathcal{C}_m|}\sum(k_t - \mu_m)^2}$
  $\mathcal{O} \leftarrow$ Top-p of $\frac{1-\cos(k_t, \mu_m)}{\|\sigma_m\|_2}$
  $c_m^{(\text{regular})} \leftarrow \text{L2-Norm}\left(\sum_{k_t \in \mathcal{C}_m \setminus \mathcal{O}} k_t\right)$
end for
$\phi(k_j) \leftarrow \sqrt{2/r}\cos(Wk_j + b)$
$h_j \leftarrow \text{Binarize}(\phi(k_j))$
for $s = 1$ **to** u **do**
  $\mathcal{B}_s \leftarrow \{k_j | \mathcal{H}(k_j) = s\}$
  $c_s^{(\text{irregular})} \leftarrow \text{L2-Norm}\left(\sum_{k_j \in \mathcal{B}_s} k_j\right)$
end for
$\mathcal{M} \leftarrow \{c_m^{(\text{regular})}\}_{m=1}^{k} \cup \{c_s^{(\text{irregular})}\}_{s=1}^{u}$
for each token k_t **do**
  $\mathcal{C}(k_t) \leftarrow \arg\max_{c \in \mathcal{M}} \frac{k_t^\top c}{\|k_t\|_2 \|c\|_2}$
end for
for each head $h \in [H]$ **do**
  $C^{(h)} \leftarrow \sum_{i=L_{\text{prefix}}+1}^{L_{\text{prompt}}} W_{\text{obs}}^{(h)}[i, :]$
  $\hat{C}^{(h)}(k) \leftarrow \frac{1}{|\{t|\mathcal{C}(k_t)=k\}|}\sum_{\mathcal{C}(k_t)=k} C^{(h)}(t)$
  $I^{(h)} \leftarrow$ Top-$(\mathcal{B} - L_{\text{obs}})$ indices of $\hat{C}^{(h)}$
  $\tilde{K}^{(h)} \leftarrow \{k_i^{(h)}\}_{i \in I^{(h)}} \cup \{k_j^{(h)}\}_{j=L_{\text{prefix}}+1}^{L_{\text{prompt}}}$
  $\tilde{V}^{(h)} \leftarrow \{v_i^{(h)}\}_{i \in I^{(h)}} \cup \{v_j^{(h)}\}_{j=L_{\text{prefix}}+1}^{L_{\text{prompt}}}$
end for

F Clustering Visualization

We visualize some clustering examples, including the K-means algorithm, our ProtoKV clustering method, and the chunk-based method that relies solely on the locality assumption. We use circles to represent regular tokens whose key vectors are primarily determined by their positions, while irregular tokens are denoted by pentagrams. As shown in Figure 10, our method achieves nearly identical results to K-means, whereas the chunk-based method fails to effectively cluster tokens that violate the locality assumption.

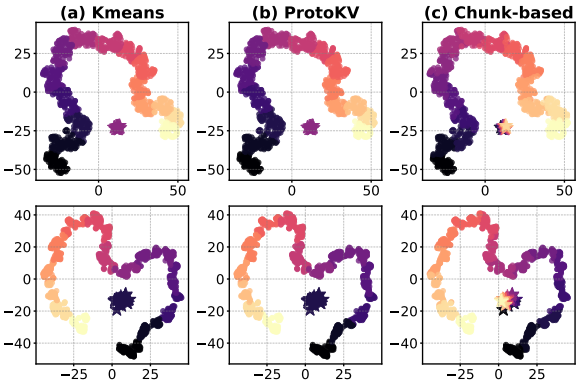


Figure 10: T-SNE visualization of key embeddings for clustering results of different methods.

G Additional Experimental Results

We present two additional experimental results: (1) a complete exposition of the results in Section 3.4, and (2) a visualization-based demonstration of the pervasiveness of the phenomenon identified in Section 3.4.

1. As mentioned in the Section 3.4, when we progressively tighten the selection criteria for irregular tokens, they form increasingly compact clusters. While the main text only reports results from two layers, here we present findings across all 32 layers (see Figure 11).
2. We visualize key embeddings with a color gradient from dark to light, representing the sequential order of tokens in a sentence (earlier tokens are darker). Since we define irregular tokens as those that violate the locality assumption yet form tight clusters, so if tokens with significantly different colors (i.e., distant in sequence) group into an isolated cluster, this confirms the existence of irregular tokens. As shown in the Figure 12 and Figure 13, we observe irregular tokens exist across different large language models, different datasets, and different attention heads, demonstrating that this is a widespread phenomenon.

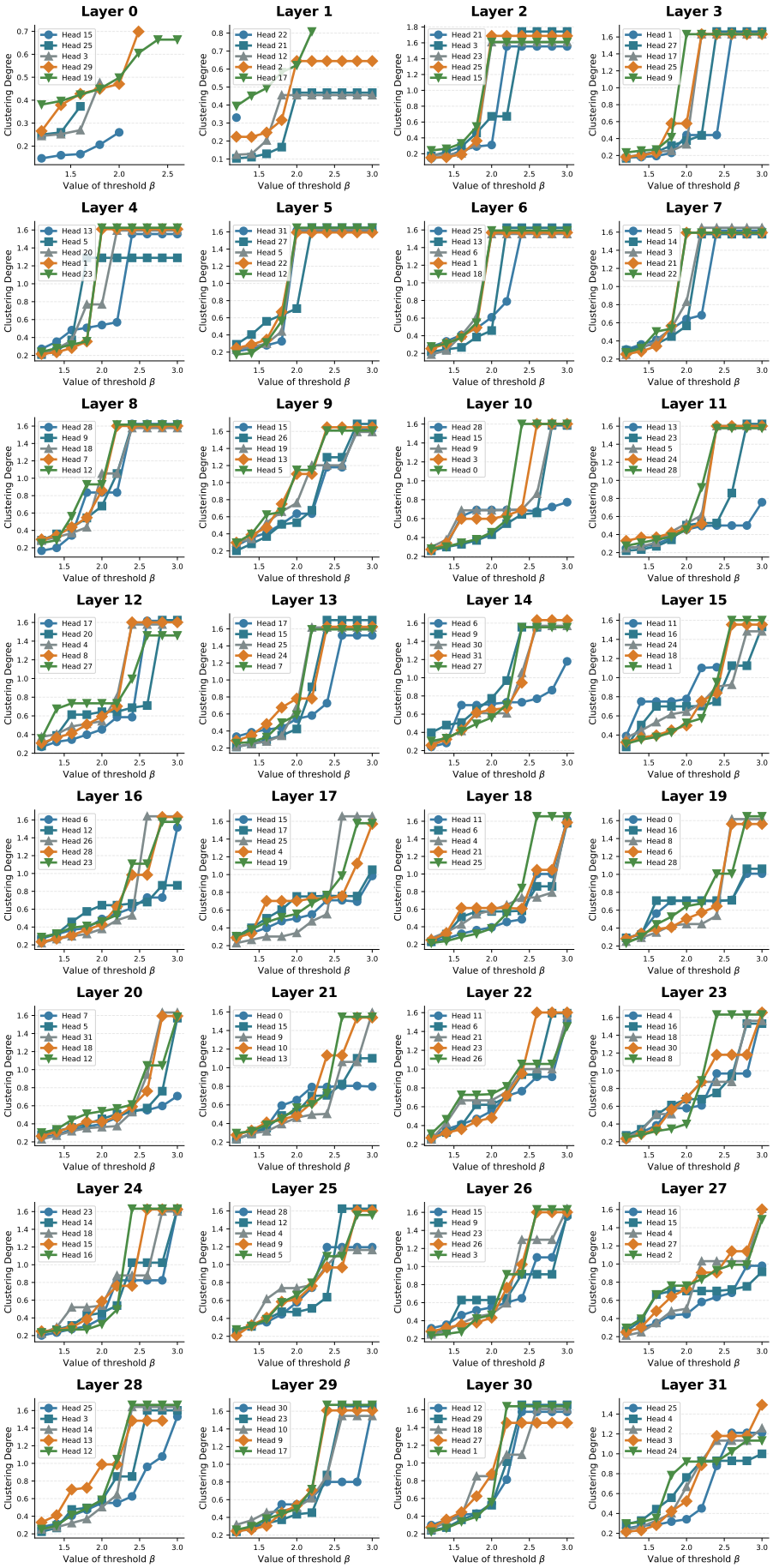


Figure 11: Non-regular tokens form progressively compact clustering with the increasing the threshold β, with H_i denoting the ith attention head.

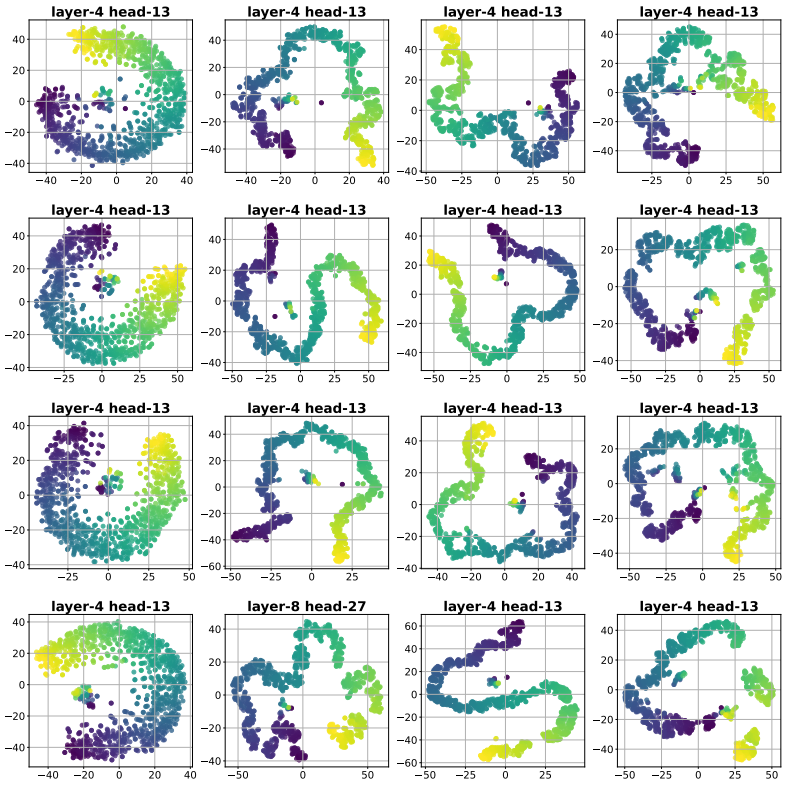


Figure 12: T-SNE visualization of key embeddings from Llama2-7B-chat on (qasper, hotpotqa, musique, 2wikimqa) dataset. Each subplot represents different combinations of layers and heads within the model architecture. The color gradient indicates the sequence of tokens in the input text.

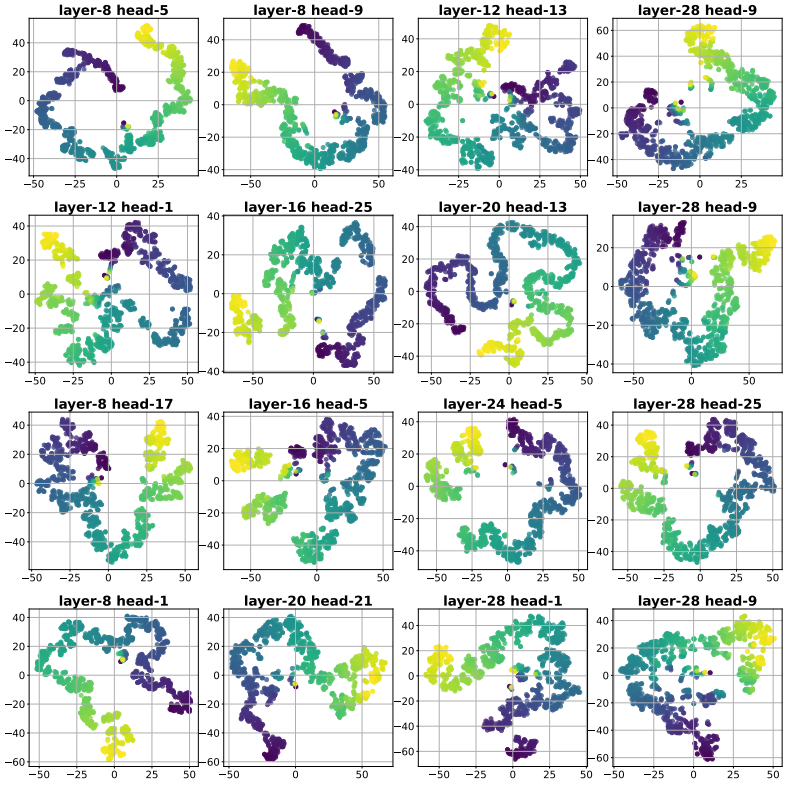


Figure 13: T-SNE visualization of key embeddings from Llama3-8B-Instruct on (qasper, hotpotqa, musique, 2wikimqa) dataset. Each subplot represents different combinations of layers and heads within the model architecture. The color gradient indicates the sequence of tokens in the input text.