

ASP: Adaptive Semantic Preservation for Context Compression

Shaoshen Chen^{1,2}, Peiyang Li^{1,2}, Jiaming Mu²,
Guixiong Chen^{2,3}, Fengting Li^{2*}, Chao Feng²

¹Tsinghua University, ²Ant Group,

³Southern University of Science and Technology

css24@mails.tsinghua.edu.cn,

lifengting.lft@antgroup.com

Abstract

Context compression has emerged as a vital approach to mitigate both performance degradation and resource consumption for large language models (LLMs) in long context scenarios. Current state-of-the-art methods allocate the compression budget at fixed-size block granularity, either uniformly or with adaptation across blocks. Both strategies fragment sentence-level semantic units and overlook the relevance variation within each block. In this paper, we propose the Adaptive Semantic Preservation (ASP) compressor, built on two innovations: (1) To preserve semantic units, our Semantic-Preserving Segmentation partitions context along natural sentence boundaries while maintaining the morphological atomicity of words. (2) To prioritize critical information, our task-aware Adaptive Budget Allocation dynamically assigns compression budget based on each sentence’s informational salience for the task instruction. Extensive experiments across diverse long-context scenarios show that ASP outperforms strong task-aware and task-agnostic compression baselines on both Llama-3-8B-Instruct and Qwen-2.5-7B-Instruct. In particular, ASP delivers stronger robustness compared with other baselines even at the extreme $16\times$ high compression ratio.

1 Introduction

Large language models (LLMs) (Achiam et al., 2023; Bai et al., 2023; Touvron et al., 2023) excel at processing extended contexts, resulting in remarkable performance on tasks such as Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Huang and Huang, 2024) and In-Context Learning (ICL) (Dong et al., 2024). However, scaling context length introduces two major challenges: (1) attention drift or collapsing, where the model’s focus becomes dispersed across distant tokens, leading

to performance degradation, and (2) quadratic computational complexity of self-attention with respect to context length, resulting in substantial resource consumption and latency during inference on long context. Consequently, **context compression** has become an important technique to reduce the input length while maintaining semantic fidelity.

Existing context compression methods typically fall into two categories. **Explicit methods** (Jiang et al., 2023; Pan et al., 2024; Jiang et al., 2024) directly shorten the context by truncating or summarizing, which preserves core meaning of context but often loses important fine-grained details. In contrast, **implicit methods** (Mu et al., 2023; Cheng et al., 2024; Deng et al., 2025b) encode context into dense latent representations (denoted as compressed tokens, $\langle CT \rangle$), offering better retention of overall semantics. Early implicit methods typically appended compressed tokens to the end of the entire text, which is poor at preserving local details. Current state-of-the-art (SOTA) implicit methods (Ge et al., 2024; Chevalier et al., 2023; Zhang et al., 2025) employ recursive compression strategies, which split context into **fixed-size** blocks and uniformly distribute compressed tokens across blocks to balance local and global information.

First, prior methods allocate the compression budget at block granularity, either uniformly (Zhang et al., 2025; Ge et al., 2024; Chevalier et al., 2023) or adaptively per block via attention/perplexity signals (Chen et al., 2025). Both regimes operate at the coarse block boundary rather than the natural semantic unit, so a high-relevance sentence is still diluted when it shares a block with low-relevance content. Second, fixed-size segmentation and uniform distribution of compressed tokens often fragment sentences or words, violating the principle of morphological atomicity, i.e., meaningful units should remain intact. For example, as shown in Figure 1, due to the uniform insertion of compressed tokens, the word “Canterbury” be-

*Corresponding author.

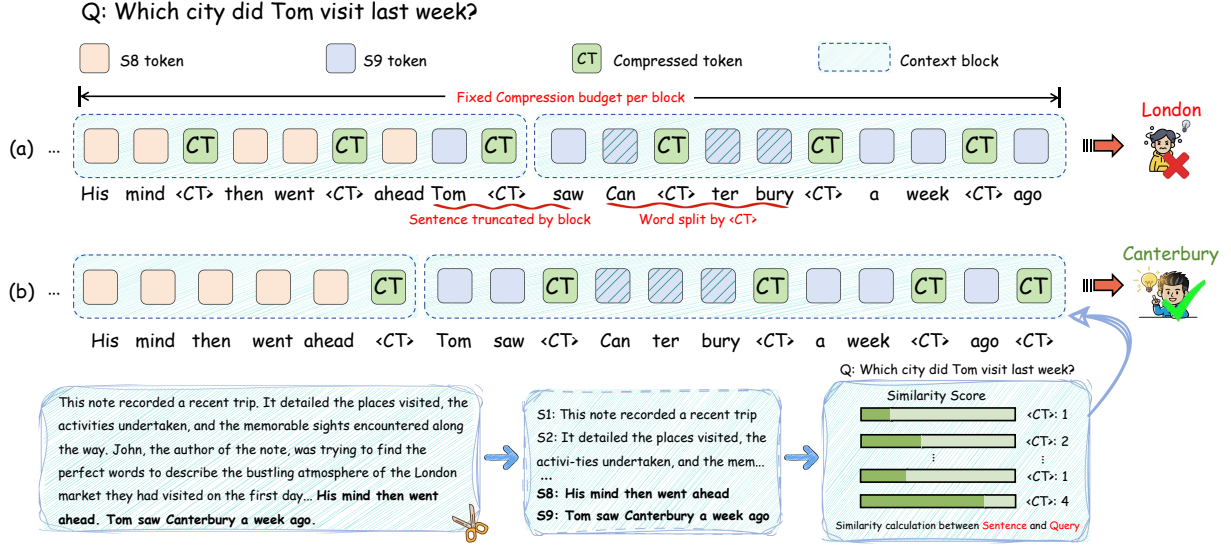


Figure 1: (a) Previous recursive compression methods with their problems and (b) our method, where “Canterbury” is a morphologically atomic word and <CT> represents a compressed token.

comes discontinuous.

To address these issues, we propose the Adaptive Semantic Preservation (ASP) compressor, designed to maximize semantic preservation under the same compression budget. ASP includes two novel modules: (i) the Semantic-Preserving Segmentation module partitions the original context along natural sentence boundaries and adaptively adjusts the insertion positions of compression tokens based on tokenization; (ii) the Adaptive Budget Allocation module dynamically assigns sentence-level compression budgets based on each sentence’s relevance to the task instruction, moving allocation granularity from the block level used by prior adaptive methods down to the sentence level and ensuring critical content is preserved with the highest fidelity.

Our main contributions can be summarized as follows:

- We propose the Adaptive Semantic Preservation compressor, a novel method that achieves superior semantic information preservation compared to existing compression methods under the same compression ratio.
- ASP’s effectiveness stems from two core innovations: (1) **Semantic-Preserving Segmentation**, which divides the original context along natural sentence boundaries while maintaining word integrity to prevent semantic distortion, and (2) **Adaptive Budget Allocation**, a task-aware mechanism that moves allocation granularity from the block level used by

prior adaptive methods (Chen et al., 2025) down to the sentence level, made feasible by the Semantic-Preserving Segmentation module and ensuring that the most critical content is preserved with the highest fidelity.

- Extensive experiments across diverse long-context scenarios show that ASP consistently outperforms multiple existing SOTA baselines. In particular, ASP delivers remarkable robustness compared with other baselines even at the extreme $16\times$ high compression ratio.

2 Related Work

2.1 Task-Agnostic Context Compression Methods

Task-agnostic methods compress input context without relying on specific queries, preserving global semantics to support diverse downstream tasks. This category can be further split into two technical routes: (1) Explicit compression, which retains explicit textual tokens via metric-based pruning, structured truncation or general summarization (Bolya et al., 2023; Xu et al., 2024; Pan et al., 2024; Li et al., 2023b; Zhou et al., 2026). (2) Implicit compression, which maps long context into dense compressed embeddings, including one-shot encoding frameworks (Mu et al., 2023; Kim et al., 2024; Qin et al., 2024; Zhao et al., 2025a; Tan et al., 2024; Liao et al., 2025; Petrov et al., 2025; Wang et al., 2024), recursive autoregressive paradigms and a series of efficiency-optimized designs (Li et al., 2025; Ge et al., 2024; Chevalier

et al., 2023; Zhang et al., 2025; Dai et al., 2025; Liu et al., 2026; Deng et al., 2025a; Fei et al., 2024; Berton et al., 2025). *Despite good cross-scenario generalization, these methods inevitably dilute task-critical information and degrade instruction-driven downstream task performance.*

2.2 Task-Aware Context Compression Methods

Task-aware methods incorporate task queries during compression to prioritize task-relevant content, the mainstream direction for high-performance long-context compression. Representative works include query-guided explicit context pruning and summarization (Yoon et al., 2024; Hwang et al., 2025; Liskavets et al., 2025; Fang et al., 2025; Zhao et al., 2025b; Jiang et al., 2023, 2024), query-aware implicit one-shot and recursive compression (Chen et al., 2025; Rau et al., 2024; Choi et al., 2025; Li et al., 2024a; Cheng et al., 2024), and query-guided KV cache compression that optimizes inference efficiency via relevance-based KV pruning (Xiao et al., 2023; Li et al., 2024b; Feng et al., 2025). *While these methods effectively preserve task-specific content, they either irreversibly discard low-relevance content (at the token level for LongLLMLingua (Jiang et al., 2024), or at the sentence level for CPC (Liskavets et al., 2025)), or apply adaptive allocation only at the block level, which leaves the sentence-level relevance variation within each block unexploited and still inherits the morphological fragmentation caused by fixed-size segmentation (e.g., DAST (Chen et al., 2025)).*

3 Method

3.1 Notation

Let $X = \{x_i\}_{i=1}^L$ denote an input context of length L tokens. Context compression aims to produce condensed $\tilde{X} = \{\tilde{x}_i\}_{i=1}^{\tilde{L}}$, where $\tilde{L} \ll L$. The compression rate is defined as $\tau = \frac{\tilde{L}}{L}$.

3.2 Overview

As illustrated in Figure 1, existing implicit compression methods split the context into fixed-size blocks, leading to two main issues: (1) Even methods that adapt the budget across blocks (Chen et al., 2025) cannot exploit the finer relevance variation *within* each block, so a high-relevance sentence still receives a sub-block uniform share of compressed tokens. (2) More importantly, they frequently cause

fragmentation at boundaries and overlook the principle of morphological atomicity that refers to treating each complete word as the smallest meaningful unit. In contrast, our proposed ASP method addresses these limitations by maintaining context integrity and prioritizing important content while reducing redundancy. ASP operates through four key stages: semantic-preserving segmentation (Section 3.3), dynamic compression budget allocation (Section 3.4), end-to-end compression framework (Section 3.5) and compression learning (Section 3.6).

3.3 Semantic-Preserving Segmentation

We define a morphologically atomic word as the smallest sequence of subword tokens whose surface form constitutes a complete word in the original text. The morphological atomicity of a segmentation refers to the constraint that no compressed token $\langle CT \rangle$ may split a morphologically atomic word. Our objective in this section is to convert the input context X into a sequence composed of adaptive context blocks interleaved with compressed tokens. This approach aims to preserve both the meaning of individual blocks and the overall semantic coherence of the context.

We begin by segmenting X into sentences using a standard sentence splitter¹:

$$X = \{s_1, s_2, \dots, s_i, \dots, s_n\}, \quad (1)$$

where s_i denotes the i -th sentence in the context and n is the number of sentences split from X .

Unlike previous methods that insert compressed tokens $\langle CT \rangle$ at uniform intervals within fixed-size blocks, our method employs a rule-based algorithm to determine the morphological atomicity of each word. These atomic units serve as the smallest units for segmentation. The details of this algorithm are shown in Appendix A. We organize context and compressed tokens as follows within s_i :

$$\tilde{s}_i = \{\dots, \langle CT \rangle, \underbrace{x_{i,j-1}, x_{i,j}, x_{i,j+1}}_{\text{morphologically atomic word}}, \dots, \langle CT \rangle, \dots\}. \quad (2)$$

Here, $x_{i,j-1}, x_{i,j}, x_{i,j+1}$ together make up a complete word, ensuring the morphological atomicity. By grouping tokens this way, our method

¹We use a basic sentence splitter for its simplicity and reliability in practice.

maintains the semantic integrity of each word during compression, preventing meaningful units from being split. It is worth mentioning that the length of each word is not fixed.

3.4 Adaptive Budget Allocation

We assign each sentence a compression budget proportional to its similarity to the task instruction. The total budget $M = \lfloor \tau \cdot L \rfloor$ is fixed; sentences that score higher receive a larger budget at the expense of less relevant ones.

Given a task instruction I , we use cosine similarity to measure the importance of s_i as follows:

$$\begin{aligned}\bar{\mathbf{h}}_{s_i} &= \frac{1}{|s_i|} \sum_{j \in s_i} \mathbf{H}(j), \\ \bar{\mathbf{h}}_I &= \frac{1}{|I|} \sum_{j \in I} \mathbf{H}(j),\end{aligned}\quad (3)$$

$$\text{sim}_i = \frac{\bar{\mathbf{h}}_{s_i} \cdot \bar{\mathbf{h}}_I}{\|\bar{\mathbf{h}}_{s_i}\| \cdot \|\bar{\mathbf{h}}_I\|}. \quad (4)$$

Here $\mathbf{H}(\cdot)$ denotes the last-layer hidden states of the frozen backbone LLM, and $\bar{\mathbf{h}}_{s_i}, \bar{\mathbf{h}}_I$ are mean-pooled embeddings of sentence s_i and the task instruction I . Because scores depend only on each sentence individually, all sim_i can be computed in parallel within a single LLM forward pass. The compression budget assigned to each sentence is:

$$\begin{aligned}\hat{\text{sim}}_i &= \max(0, \text{sim}_i), \\ m_i &= \max\left(1, \left\lfloor M \times \frac{\hat{\text{sim}}_i}{\sum_{j=1}^n \hat{\text{sim}}_j} \right\rfloor\right),\end{aligned}\quad (5)$$

where the shift to non-negative similarities ensures valid normalization, the $m_i \geq 1$ constraint preserves every sentence with at least one $\langle CT \rangle$ token, and any total exceeding M is reconciled by proportionally reducing the highest-budget sentences. When $M < n$, we merge adjacent sentences before allocation.

3.5 End-to-End Compression Framework

Similar to previous methods (Ge et al., 2024; Zhang et al., 2025; Chen et al., 2025), our method employs a recursive compression mechanism, ensuring that both local details and global coherence are preserved. Rather than dividing context into fixed-size blocks, we segment the context into blocks containing complete sentences, constrained by a maximum block length. For the i -th block, let $B_i = \{s_1, \dots, s_n\}$ denote the set of sentences in

the block. We then insert compressed tokens to obtain the enhanced block representation:

$$\tilde{B}_i = \{\tilde{s}_1, \dots, \tilde{s}_n\}. \quad (6)$$

Compression of the i -th block conditions on the cached $\langle CT \rangle$ from all previous blocks together with the current block’s tokens and its own newly inserted $\langle CT \rangle$:

$$\text{LLM}\left(\langle CT \rangle_1^i, \dots, \langle CT \rangle_k^i, \tilde{B}_i\right), \quad (7)$$

where k represents the budget before $\tilde{B}_i$. We freeze all backbone parameters and add a separate set of trainable projections $\mathbf{W}_q^{ct}, \mathbf{W}_k^{ct}, \mathbf{W}_v^{ct}$ that operate only on $\langle CT \rangle$ tokens (Figure 2). Decoupling the two paths lets us train only $\sim 6\%$ parameters (Appendix J) without touching the original LLM weights, following Petrov et al. (2025).

$$\begin{aligned}\mathbf{Q}^{l+1} &= \mathbf{W}_q^* \mathbf{H}^l, \\ \mathbf{K}^{l+1} &= \mathbf{W}_k^* \mathbf{H}^l, \\ \mathbf{V}^{l+1} &= \mathbf{W}_v^* \mathbf{H}^l,\end{aligned}\quad (8)$$

where $*$ $\in \{ot, ct\}$ selects between the original projections (applied to context tokens) and the newly introduced projections (applied to $\langle CT \rangle$ tokens). Then we can separate the original text and the compressed tokens:

$$\mathbb{I}_{ot} = \{j \mid x_j \neq \langle CT \rangle\}, \mathbb{I}_{ct} = \{j \mid x_j = \langle CT \rangle\}. \quad (9)$$

$$\begin{aligned}\mathbf{Q}_{ot}^l &= \mathbf{Q}^l[\mathbb{I}_{ot}], \mathbf{Q}_{ct}^l = \mathbf{Q}^l[\mathbb{I}_{ct}], \\ \mathbf{K}_{ot}^l &= \mathbf{K}^l[\mathbb{I}_{ot}], \mathbf{K}_{ct}^l = \mathbf{K}^l[\mathbb{I}_{ct}], \\ \mathbf{V}_{ot}^l &= \mathbf{V}^l[\mathbb{I}_{ot}], \mathbf{V}_{ct}^l = \mathbf{V}^l[\mathbb{I}_{ct}].\end{aligned}\quad (10)$$

where $\mathbf{Q}^l, \mathbf{K}^l, \mathbf{V}^l \in \mathbb{R}^{w \times D}$ represent the queries, keys, and values for both token types, consolidated into the full matrices at layer l , D is the model’s hidden size and w is the sum of context tokens and compressed tokens of $\tilde{B}_i$. Let $\mathbf{K}_p^l, \mathbf{V}_p^l$ indicate the hidden states of previous compressed tokens at layer l , the combined matrices are:

$$\mathbf{K}^l = [\mathbf{K}_{ot}^l; \mathbf{K}_{ct}^l], \mathbf{V}^l = [\mathbf{V}_{ot}^l; \mathbf{V}_{ct}^l]. \quad (11)$$

Finally, standard causal self-attention is applied:

$$\mathbf{H}^{l+1} = \text{Softmax}\left(\text{Mask}\left(\frac{\mathbf{Q}^l(\mathbf{K}^l)^\top}{\sqrt{d_k}}\right)\right) \mathbf{V}^l, \quad (12)$$

where d_k is the key dimension. Through this recursive autoregressive approach, each compressed

token is trained to encode task-relevant information from the preceding context. Consequently, only the hidden states of these compressed tokens need to be saved for subsequent blocks. Thus, we update the past key-value states as follows:

$$\mathbf{K}_p^l = [\mathbf{K}_p^l; \mathbf{K}_{ct}^l], \mathbf{V}_p^l = [\mathbf{V}_p^l; \mathbf{V}_{ct}^l] \quad (13)$$

3.6 Compression Learning

ASP is trained with next-token prediction on the concatenation of the compressed prefix and the current block. Let $\mathcal{T}_{<i}$ denote the set of $\langle CT \rangle$ tokens accumulated before position x_j . Only the compressed-token projections θ_{ct} are updated, and the backbone θ stays frozen:

$$\min_{\theta_{ct}} \mathbb{E}_{\mathbf{x}} \left[- \sum_{\tilde{\mathbf{B}}_i \in \mathbf{X}} \sum_{x_j \in \tilde{\mathbf{B}}_i} \log P(x_j | \mathcal{T}_{<i}, x_{<j}; \theta_{ct}, \theta) \right], \quad (14)$$

where θ denotes the frozen LLM parameters and θ_{ct} the trainable compressed token projections.

4 Experiment

4.1 Setting

Backbone LLMs Llama-3-8B-Instruct and Qwen-2.5-7B-Instruct are employed as the backbone LLMs.

Implementation To ensure fair comparability across all methods, we apply the same overall compression ratio to all baselines. Following established practices in recursive compression (Zhang et al., 2025; Chen et al., 2025), the maximum block size is set to 1,024 for Llama-3-8B-Instruct and 2,048 for Qwen-2.5-7B-Instruct (In Appendix D, we compared the performance of different window sizes.). For a rigorous comparison, all baseline results are re-implemented using their official open-source code with the same hyperparameter settings. Unless otherwise specified, we apply $8 \times$ compression to Llama-3-8B-Instruct and Qwen-2.5-7B-Instruct. Detailed training configurations of ASP are provided in Appendix B.

Baselines We conduct comprehensive comparisons against both task-aware and task-agnostic context compression methods. Specifically, we include task-aware approaches: LLMLingua (Jiang et al., 2023), LongLLMLingua (Jiang et al., 2024), DAST (Chen et al., 2025) and AdaKV (Feng et al., 2025), as well as task-agnostic approaches: LLMLingua-2 (Pan et al., 2024) and Activation

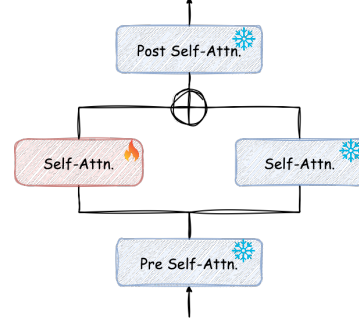


Figure 2: The model training architecture.

Beacon (Zhang et al., 2025). Additionally, we fine-tune the original LLMs on the same data (denoted as **LLM-FT**) and evaluate their performance under naive compression (denoted as **LLM-Compressed**) to establish the upper and lower performance bounds for our experiments.

Evaluation Benchmark We conduct the primary evaluation of ASP on LongBench, the mainstream and widely adopted benchmark for long-context LLM evaluation. This benchmark covers a diverse set of tasks that rigorously test models' long-context understanding abilities, spanning question answering, text summarization, and code understanding, with input context lengths reaching up to 32K tokens. For scenario-specific validation, we further test ASP on four dedicated datasets: MSC (Packer et al., 2023), ShareGPT (Chiang et al., 2023), NaturalQuestions (Kwiatkowski et al., 2019), and Topic Retrieval (Li et al., 2023a). Supplementary experiments on ultra-long context understanding, which are performed on LongBench v2 (Bai et al., 2025) and InfiniteBench (Zhang et al., 2024), are fully documented in the appendix B.

4.2 Main Experiments

Table 1 presents the performance of ASP and representative baselines on LongBench (Bai et al., 2024), a multi-task benchmark for long-context understanding. We summarize three key findings below:

1) ASP achieves the best results among the compared compression baselines on both backbones. Specifically, ASP achieves an average score of 43.38 on Llama-3-8B-Instruct and 44.25 on Qwen-2.5-7B-Instruct, surpassing the previous best task-aware baseline AdaKV by 4.46 and 2.87 absolute points, respectively. Notably, ASP shows its largest improvements in challenging Single-Document and

Methods	LongBench						
	SingleDoc	MultiDoc	Summ.	FewShot	Code	AVG	Latency
Llama-3-8B							
Llama-3-8B-FT	35.42	36.00	26.32	68.91	62.98	44.71	26.5
Llama-3-8B-Compressed	22.66	25.66	21.63	61.74	45.31	34.69	4.8
LLMLingua	23.26	21.35	20.12	40.36	31.52	27.02	9.6
LLMLingua-2	24.12	23.13	21.01	40.51	30.19	27.62	11.3
LongLLMLingua	25.63	26.21	<u>24.32</u>	43.51	35.13	30.66	10.0
Activation Beacon	30.92	29.94	23.78	49.93	40.56	34.63	7.3
AdaKV	28.53	<u>33.99</u>	21.04	<u>66.20</u>	<u>47.78</u>	<u>38.92</u>	10.2
DAST	<u>33.56</u>	32.61	20.16	60.21	36.42	36.60	12.3
ASP (Ours)	37.60	36.92	26.26	67.22	51.65	43.38	8.1
Qwen-2.5-7B							
Qwen-2.5-7B-FT	42.75	43.78	27.27	67.50	56.12	46.87	25.2
Qwen-2.5-7B-Compressed	29.51	31.55	23.90	64.07	39.67	37.60	4.3
LLMLingua	21.03	31.13	19.92	49.87	44.81	32.53	9.3
LLMLingua-2	25.12	32.21	18.10	50.12	43.16	33.13	12.1
LongLLMLingua	25.92	33.25	20.97	56.51	34.22	34.17	9.8
Activation Beacon	35.63	36.70	<u>24.46</u>	60.63	45.13	40.18	6.9
AdaKV	<u>37.21</u>	35.21	21.23	<u>65.31</u>	<u>51.20</u>	<u>41.38</u>	9.5
DAST	36.23	<u>36.91</u>	20.32	63.12	40.21	39.30	11.3
ASP (Ours)	39.53	38.49	24.94	66.94	54.91	44.25	7.8

Table 1: Performance on the LongBench benchmark (Bai et al., 2024). This benchmark includes at least two datasets for each task type to ensure robust evaluation. **Bold** indicates the best performance, and underline indicates the second-best performance among compression methods for each task.

Multi-Document QA tasks, outperforming all task-aware and task-agnostic implicit/explicit compression baselines. By better preserving task-relevant semantic information and prioritizing task-critical content, ASP ensures that the model retains high-quality knowledge under strict context limitations.

2) ASP delivers strong results in tasks requiring global context retention. On Summarization, ASP matches the full-context fine-tuned models on Llama-3-8B-FT (0.06 points) and falls within 2.33 points on Qwen-2.5-7B-FT. This finding highlights ASP’s ability to maintain comprehensive task-relevant content and logical integrity without explicit query signals during compression. Similarly, ASP overtakes all compression baselines in Code tasks, indicating that its adaptive selection enables the model to keep both granular task-critical details and high-level structural information necessary for accurate code generation.

3) ASP achieves a remarkable balance between task performance and inference efficiency. For

Llama-3-8B-Instruct, ASP closes the performance gap with the full-context fine-tuned model to just 1.33 absolute points, while delivering a $3.3\times$ end-to-end speedup over the uncompressed prompt. Compared with naive compression, ASP yields an 8.69-point improvement in average task performance for Llama-3-8B-Instruct and a 6.65-point gain for Qwen-2.5-7B-Instruct, all while maintaining latency competitive with or lower than most SOTA compression baselines. These results affirm that ASP efficiently identifies and preserves task-critical information, achieving both high task quality and high inference efficiency.

4.3 Ablation Studies

To quantify the contribution of each component within ASP, we conduct comprehensive ablation studies using the Qwen-2.5-7B-Instruct backbone on the SingleDoc and Summarization tasks from LongBench. The results are summarized in Table 3.

First, we validate the effectiveness of our two-stage training strategy. Omitting the pre-training

Method	1 Turn (765 tokens)		2 Turn (3006 tokens)		3 Turn (6491 tokens)	
	PPL↓	ROUGE-L	PPL↓	ROUGE-L	PPL↓	ROUGE-L
LLMLingua	4.563	42.12	3.892	31.01	3.523	30.21
LLMLingua-2	4.312	43.26	3.731	33.52	3.313	32.16
LongLLMLingua	4.231	43.52	3.612	32.96	3.213	36.13
Activation Beacon	3.072	48.99	2.572	37.82	2.557	42.80
AdaKV	3.261	46.20	2.413	35.21	2.389	41.20
DAST	3.125	50.21	2.398	39.32	2.401	40.63
ASP (Ours)	2.875	51.26	2.331	41.61	2.346	44.50

Table 2: Experiments in dialogue scenarios on ShareGPT dataset (Chiang et al., 2023).

phase reduces performance by 4.98 points on SingleDoc and 3.83 points on Summarization. Removing the fine-tuning stage leads to even more pronounced drops of 6.13 and 4.69 points on SingleDoc and Summarization, respectively. These consistent declines across both tasks confirm that the two stages are highly complementary: pre-training aligns the semantic space of the compressed tokens with the frozen LLM backbone, while fine-tuning adapts the method to instruction-driven task requirements.

Ablation Setting	SingleDoc	Summ.
Full ASP (Ours)	39.53	24.94
w/o Pre-training Stage	34.55	21.11
w/o Fine-tuning Stage	33.40	20.25
w/o Semantic-Preserving Segmentation	36.62	22.31
w/o Adaptive Budget Allocation	37.57	23.31
w/ PPL-based Unsupervised Scoring	38.82	23.59

Table 3: Ablation study results on Qwen-2.5-7B-Instruct backbone, evaluated on LongBench.

Next, we evaluate our two core modules. Replacing our Semantic-Preserving Segmentation with fixed-length chunking (a common practice in prior work such as LongLLMLingua (Jiang et al., 2024) and DAST (Chen et al., 2025)) degrades performance by 2.91 points on SingleDoc and 2.63 points on Summarization. This decline occurs because fixed-length chunking risks ignoring and fracturing the underlying semantic structure. Furthermore, ablating the Adaptive Budget Allocation (i.e., reverting to a fixed, uniform compression rate) impacts SingleDoc slightly more than Summarization (a 1.96-point versus 1.63-point drop). This disparity suggests that dynamic budget prioritization is particularly beneficial for sparse, long documents where task-relevant information is unevenly dis-

Method	Compression Constraint			
	~2 x	~4 x	~8 x	~16 x
Llama-3-8B				
LLMLingua	45.23	39.26	33.26	31.51
LLMLingua-2	46.16	43.31	39.15	36.28
LongLLM.	63.68	57.21	41.61	37.58
Activation.	74.31	62.69	52.04	30.08
AdaKV	76.23	66.14	59.03	42.53
DAST	76.56	69.39	60.21	50.01
ASP (Ours)	79.73	79.27	76.05	64.64
Qwen-2.5-7B				
LLMLingua	41.70	32.08	31.14	29.79
LLMLingua-2	45.18	35.21	33.20	30.09
LongLLM.	55.20	42.66	38.01	38.93
Activation.	63.89	56.55	44.72	24.35
AdaKV	71.37	61.67	53.65	39.21
DAST	65.21	58.35	49.63	46.12
ASP (Ours)	76.09	74.21	67.94	60.38

Table 4: Evaluation of different compression on MSC.

tributed.

Finally, we replace our instruction-aware scoring with an unsupervised PPL-based metric to assess robustness when no task query is available at compression time. ASP maintains 38.82 on SingleDoc and 23.59 on Summarization, only 0.71 and 1.35 points below the full instruction-aware version, confirming that the method extends seamlessly to instruction-free scenarios.

4.4 Evaluation with Different Compression

As shown in Table 4, we further analyze performance under varying compression constraint on the Multi-Session Conversation (MSC) dataset (Packer et al., 2023), using ROUGE-L to evaluate response quality. Across both Llama and Qwen models, ASP consistently outperforms all other baselines. No-

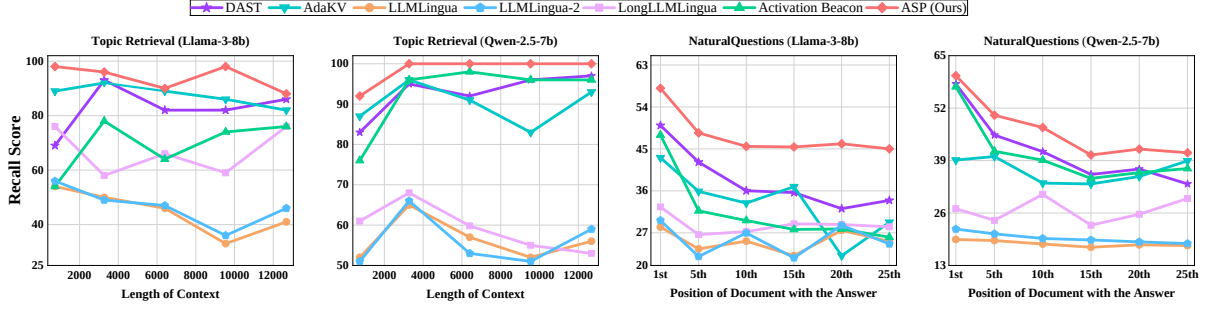


Figure 3: Evaluation of Long-Term and Fine-grained Information Compression on Topic Retrieval and NaturalQuestions.

tably, while a competitive method like Activation Beacon performs well at $2\times$ compression, its effectiveness sharply declines as compression becomes more aggressive. In contrast, ASP demonstrates robust performance, maintaining high-quality outputs even under stringent compression constraints, which highlights its superior efficiency.

4.5 Evaluation in Multi-turn Dialogue

To evaluate the performance of context compression in multi-turn dialogue scenarios, we conducted experiments on the ShareGPT dataset (Chiang et al., 2023). The evaluation spanned three consecutive turns, with the average context length scaling progressively from 765 to 6,491 tokens. We utilized PPL to assess the linguistic integrity of the model’s responses and ROUGE-L to quantify information retention following compression.

As shown in Table 2, ASP consistently outperforms all established baselines across all conversational turns. Under the same compression constraint, ASP maintains the lowest PPL and the highest ROUGE-L scores, demonstrating superior robustness as the context grows. Notably, in the 3-turn setting, ASP achieves a ROUGE-L of 44.50, significantly surpassing AdaKV and LongLLMLingua. This performance gap suggests that ASP can preserve critical information more accurately than other compression approaches as the dialogue complexity increases.

4.6 Long-Term and Fine-grained Information Compression

To evaluate ASP’s ability to retain and retrieve specific information regardless of its position in a long context, we conducted two targeted experiments.

Long-Term Memory We first use the Topic Retrieval benchmark (Li et al., 2023a) to assess the model’s memory. In this task, the model is asked to

recall the first topic as the length of context is progressively increased. As shown in Figure 3, ASP demonstrates clear superiority over all baselines at $4\times$ compression. Crucially, despite the growth of context, our method still achieves solid performance, a result that highlights the effectiveness of ASP in long-term information memory.

Fine-grained Information Next, we conduct a more fine-grained analysis to assess performance sensitivity to the position of the relevant information. For this, we use the NaturalQuestions (NQ) dataset (Kwiatkowski et al., 2019) to synthetically construct test cases by placing the answer-bearing context at different locations (e.g., beginning, middle, end) within a long document. Using recall as the evaluation metric at $4\times$ compression, the results again confirm ASP’s advantage. It consistently outperforms all baselines and, more importantly, maintains high recall across all tested positions. This demonstrates ASP’s robustness to positional bias, showcasing the precision of our compression approach in preserving key information wherever it may appear.

5 Conclusion

In this paper, we propose the ASP compressor, a novel method designed to significantly preserve semantic integrity while selectively focusing on the most critical information during context compression. ASP’s effectiveness stems from two core innovations. First, Semantic-Preserving Segmentation partitions context along natural sentence boundaries while maintaining the morphological atomicity of words, thereby preserving the semantic integrity. Second, our task-aware Adaptive Budget Allocation dynamically assigns compression budget to each sentence based on its relevance to the task instruction, moving allocation granularity from the block level used by prior adaptive methods

down to the sentence level. Our extensive experiments demonstrate that ASP consistently outperforms state-of-the-art context compression methods across various benchmarks and compression ratios, establishing a superior balance between task performance and efficiency.

Limitations

First, ASP’s Semantic-Preserving Segmentation and Adaptive Budget Allocation operate at sentence granularity. Exceptionally long sentences and sentences with internal noise therefore receive a single budget assignment, with no mechanism for intra-sentence redistribution. Extending this to a hierarchical allocator that operates jointly at clause and sentence levels (preserving the soft-allocation paradigm while gaining finer control) is a natural next step.

Second, the Adaptive Budget Allocation is task-aware by construction, following the standard design adopted by recent task-aware compression methods (Jiang et al., 2024; Chen et al., 2025; Liskavets et al., 2025): a task instruction must be available at compression time. For instruction-free deployment scenarios (e.g., document indexing), our PPL-based unsupervised variant (Table 3) recovers within 0.71 / 1.35 points of the full version and serves as a drop-in replacement; explicit instruction is therefore only required for the most performance-critical settings.

Finally, we focus our multilingual evaluation on English and Chinese (Appendix H, Table 11), where ASP already exceeds the fine-tuned backbone on average. Extending the evaluation to morphologically rich languages, low-resource scripts, and writing systems without explicit sentence boundaries is a direction we are pursuing.

Acknowledgements

This work was supported by Ant Group Research Intern Program.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei

Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

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, multi-task benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand. Association for Computational Linguistics.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2025. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3639–3664, Vienna, Austria. Association for Computational Linguistics.

Gabriele Berton, Jayakrishnan Unnikrishnan, Son Tran, and Mubarak Shah. 2025. *Compllm: Compression for long context q&a*. *Preprint*, arXiv:2509.19228.

Sumithra Bhakthavatsalam, Daniel Khashabi, Tushar Khot, Bhavana Dalvi Mishra, Kyle Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, and Peter Clark. 2021. *Think you have solved direct-answer question answering? try arc-da, the direct-answer AI2 reasoning challenge*. *CoRR*, abs/2102.03315.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2023. *Token merging: Your vit but faster*. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Shaoshen Chen, Yangning Li, Zishan Xu, Yongqin Zeng, Shunlong Wu, Xinshuo Hu, Zifei Shan, Xin Su, Jiwei Tang, Yinghui Li, and Hai-Tao Zheng. 2025. *DAST: Context-aware compression in LLMs via dynamic allocation of soft tokens*. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20544–20552, Vienna, Austria. Association for Computational Linguistics.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024. LongLoRA: Efficient fine-tuning of long-context large language models. In *The Twelfth International Conference on Learning Representations*.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. 2024. *xrag: Extreme context compression for retrieval-augmented generation with one token*. 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*.

- Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. [Adapting language models to compress contexts](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3829–3846. Association for Computational Linguistics.
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. See <https://vicuna.lmsys.org> (accessed 14 April 2023), 2(3):6.
- Eunseong Choi, June Park, Hyeri Lee, and Jongwuk Lee. 2025. [Conflict-aware soft prompting for retrieval-augmented generation](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 26981–26995, Suzhou, China. Association for Computational Linguistics.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *CoRR*, abs/2110.14168.
- Yuhong Dai, Jianxun Lian, Yitian Huang, Wei Zhang, Mingyang Zhou, Mingqi Wu, Xing Xie, and Hao Liao. 2025. [Pretraining context compressor for large language models with embedding-based memory](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28715–28732, Vienna, Austria. Association for Computational Linguistics.
- Chenlong Deng, Zhisong Zhang, Kelong Mao, Shuaiyi Li, Tianqing Fang, Hongming Zhang, Haitao Mi, Dong Yu, and Zhicheng Dou. 2025a. [Unigist: Towards general and hardware-aligned sequence-level long context compression](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Chenlong Deng, Zhisong Zhang, Kelong Mao, Shuaiyi Li, Xinting Huang, Dong Yu, and Zhicheng Dou. 2025b. [A silver bullet or a compromise for full attention? A comprehensive study of gist token-based context compression](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2025, Vienna, Austria, July 27 - August 1, 2025, pages 4861–4879. Association for Computational Linguistics.
- Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. [A survey on in-context learning](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 1107–1128. Association for Computational Linguistics.
- Yixiong Fang, Tianran Sun, Yuling Shi, and Xiaodong Gu. 2025. [Attentionrag: Attention-guided context pruning in retrieval-augmented generation](#). *Preprint*, arXiv:2503.10720.
- Weizhi Fei, Xueyan Niu, Pingyi Zhou, Lu Hou, Bo Bai, Lei Deng, and Wei Han. 2024. [Extending context window of large language models via semantic compression](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5169–5181, Bangkok, Thailand. Association for Computational Linguistics.
- Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. 2025. [Ada-KV: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Tao Ge, Hu Jing, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2024. [In-context autoencoder for context compression in a large language model](#). In *The Twelfth International Conference on Learning Representations*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. [Measuring massive multitask language understanding](#). In *International Conference on Learning Representations*.
- Yizheng Huang and Jimmy Huang. 2024. [A survey on retrieval-augmented text generation for large language models](#). *CoRR*, abs/2404.10981.
- Taeho Hwang, Sukmin Cho, Soyeon Jeong, Hoyun Song, SeungYoon Han, and Jong C. Park. 2025. [EXIT: Context-aware extractive compression for enhancing retrieval-augmented generation](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 4895–4924, Vienna, Austria. Association for Computational Linguistics.
- Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. [Llmlingua: Compressing prompts for accelerated inference of large language models](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 13358–13376. Association for Computational Linguistics.
- Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. [LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1658–1677, Bangkok, Thailand. Association for Computational Linguistics.
- Jang-Hyun Kim, Junyoung Yeom, Sangdoo Yun, and Hyun Oh Song. 2024. [Compressed context memory for online language model interaction](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

- Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. 2022. Booksum: A collection of datasets for long-form narrative summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6536–6558.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. [Natural questions: a benchmark for question answering research](#). *Trans. Assoc. Comput. Linguistics*, 7:452–466.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. [Retrieval-augmented generation for knowledge-intensive NLP tasks](#). In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.
- Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. 2023a. [How long can context length of open-source LLMs truly promise?](#) In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.
- Xinze Li, Zhenghao Liu, Chenyan Xiong, Shi Yu, Yukun Yan, Shuo Wang, and Ge Yu. 2024a. Say more with less: Understanding prompt learning behaviors through gist compression. *arXiv preprint arXiv:2402.16058*.
- Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. 2023b. [Compressing context to enhance inference efficiency of large language models](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6342–6353, Singapore. Association for Computational Linguistics.
- 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. *Advances in Neural Information Processing Systems*, 37:22947–22970.
- Zongqian Li, Yixuan Su, and Nigel Collier. 2025. [500xCompressor: Generalized prompt compression for large language models](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25081–25091, Vienna, Austria. Association for Computational Linguistics.
- Huanxuan Liao, Wen Hu, Yao Xu, Shizhu He, Jun Zhao, and Kang Liu. 2025. [Beyond hard and soft: Hybrid context compression for balancing local and global information retention](#). *Preprint*, arXiv:2505.15774.
- Barys Liskavets, Maxim Ushakov, Shuvendu Roy, Mark Klibanov, Ali Etemad, and Shane K Luke. 2025. Prompt compression with context-aware sentence encoding for fast and improved llm inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24595–24604.
- Xin Liu, Runsong Zhao, Pengcheng Huang, Xinyu Liu, Junyi Xiao, Chunyang Xiao, Tong Xiao, Shengxiang Gao, Zhengtao Yu, and JingBo Zhu. 2026. [Autoencoding-free context compression for LLMs via contextual semantic anchors](#). In *The Fourteenth International Conference on Learning Representations*.
- Jesse Mu, Xiang Li, and Noah D. Goodman. 2023. [Learning to compress prompts with gist tokens](#). 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*.
- Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. 2023. [Memgpt: Towards llms as operating systems](#). *CoRR*, abs/2310.08560.
- Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. [Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression](#). In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 963–981. Association for Computational Linguistics.
- Aleksandar Petrov, Mark Sandler, Andrey Zhmoginov, Nolan Miller, and Max Vladymyrov. 2025. [Long context in-context compression by getting to the gist of gisting](#). *CoRR*, abs/2504.08934.
- Guanghui Qin, Corby Rosset, Ethan Chau, Nikhil Rao, and Benjamin Van Durme. 2024. [Dodo: Dynamic contextual compression for decoder-only LLMs](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9961–9975, Bangkok, Thailand. Association for Computational Linguistics.
- David Rau, Shuai Wang, Hervé Déjean, and Stéphane Clinchant. 2024. [Context embeddings for efficient answer generation in rag](#). *Preprint*, arXiv:2407.09252.
- Sijun Tan, Xiuyu Li, Shishir G Patil, Ziyang Wu, Tianjun Zhang, Kurt Keutzer, Joseph E. Gonzalez, and Raluca Ada Popa. 2024. [LLoCO: Learning long contexts offline](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17605–17621, Miami, Florida, USA. 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, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Xiangfeng Wang, Zaiyi Chen, Tong Xu, Zheyong Xie, Yongyi He, and Enhong Chen. 2024. [In-context former: Lightning-fast compressing context for large language model](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 2445–2460, Miami, Florida, USA. Association for Computational Linguistics.

Maurice Weber, Daniel Y. Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, Ben Athiwaratkun, Rahul Chalamala, Kezhen Chen, Max Ryabinin, Tri Dao, Percy Liang, Christopher Ré, Irina Rish, and Ce Zhang. 2024. Redpajama: an open dataset for training large language models. *NeurIPS Datasets and Benchmarks Track*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024. [RECOMP: Improving retrieval-augmented LMs with context compression and selective augmentation](#). In *The Twelfth International Conference on Learning Representations*.

Chanwoong Yoon, Taewhoo Lee, Hyeon Hwang, Minbyul Jeong, and Jaewoo Kang. 2024. [CompAct: Compressing retrieved documents actively for question answering](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21424–21439, Miami, Florida, USA. Association for Computational Linguistics.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. 2025. [Long context compression with activation beacon](#). In *The Thirteenth International Conference on Learning Representations*.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. 2024. [\$\infty\$ Bench: Extending long context evaluation beyond 100K tokens](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277, Bangkok, Thailand. Association for Computational Linguistics.

Runsong Zhao, Xin Liu, Xinyu Liu, Pengcheng Huang, Chunyang Xiao, Tong Xiao, and JingBo Zhu. 2025a. [Position IDs matter: An enhanced position layout for efficient context compression in large language](#)

[models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 17715–17734, Suzhou, China. Association for Computational Linguistics.

Yunlong Zhao, Haoran Wu, and Bo Xu. 2025b. Leveraging attention to effectively compress prompts for long-context llms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 26048–26056.

Yiqing Zhou, Yu Lei, Shuzheng Si, Qingyan Sun, Wei Wang, Yifei Wu, Hao Wen, Gang Chen, Fanchao Qi, and Maosong Sun. 2026. [From context to edus: Faithful and structured context compression via elementary discourse unit decomposition](#). *Preprint*, arXiv:2512.14244.

A Building Morphological Atomicity of Word Indices

Algorithm 1 details how we compute the word-to-token index mapping used by the Semantic-Preserving Segmentation module (Section 3.3). For each sentence, it incrementally grows a window of subword tokens until the decoded surface form matches the next word, yielding an index list I that constrains $\langle CT \rangle$ insertion to word boundaries.

Algorithm 1 Building Morphological Atomicity of Word Indices

Require: Context C , Tokenizer T
Ensure: WordTokenIndices I

```

1: function BUILDINDICES( $C, T$ )
2:    $S \leftarrow \text{Split}(C)$   $\triangleright$  Split context into sentences
3:    $I \leftarrow []$ 
4:   for each sentence  $s_i$  in  $S$  do
5:      $start \leftarrow 0$ 
6:      $end \leftarrow 1$ 
7:     Split  $s_i$  into words  $W_i$ 
8:     Tokenize  $s_i$  into tokens  $t_i$ 
9:     for each word  $w$  in  $W_i$  do
10:      while  $T.\text{decode}(t_i[start : end]).\text{strip}() \neq w$  and  $end \leq |t_i|$  do
11:         $end \leftarrow end + 1$ 
12:      end while
13:       $I.\text{append}([start, end])$ 
14:       $start \leftarrow end$ 
15:       $end \leftarrow end + 1$ 
16:    end for
17:  end for
18:  return  $I$ 
19: end function
```

B Experiment Setting

Backbone LLMs Llama-3-8B-Instruct and Qwen-2.5-7B-Instruct are employed as the backbone LLMs. In Appendix K, we also utilized the Llama-2-7B-Chat and Qwen-2.5-14B-Instruct models to compare the performance of our method across different model architectures and sizes.

Implementation ASP adopts the two-stage training protocol from Activation Beacon and DAST (Zhang et al., 2025; Chen et al., 2025) to ensure a fair comparison. The first stage involves pre-training on 1B tokens from the RedPajama dataset (Weber et al., 2024) with a learning rate of $5e-5$. This is followed by a fine-tuning stage on a mixed dataset comprised of LongAlpaca (Chen et al., 2024), BookSum (Kryściński et al., 2022), and 16,000 synthetic samples generated via GPT-3.5 (Zhang et al., 2025), using a learning rate of $1e-5$ and a batch size of 8. We run each experiment 10 times with different random seeds and report the average results. Unless otherwise noted, all performance gaps between ASP and the next-best baseline reported in our result tables are statistically significant under a paired t -test at $p < 0.05$ across 10 seeds. All experiments are conducted using the Hugging Face framework on eight NVIDIA A100 (80GB) GPUs. During pretraining, the maximum sequence length is set to 20,000 tokens and the minimum to 2,400 tokens; for fine-tuning, the maximum is also 20,000 tokens, with a minimum of 7,200 tokens. The AdamW optimizer is used, with a single training epoch. To ensure comparability across methods, we apply the same overall compression ratio to all baselines and unless otherwise specified, we apply $8\times$ compression to Llama-3-8B and Qwen-2.5-7B. Following established practices (Zhang et al., 2025; Chen et al., 2025), the maximum block size is set to 1,024 for Llama and 2,048 for Qwen (In Appendix D, we compared the performance of different window sizes). **For a rigorous comparison, all baseline results are re-implemented using their official open-source code with the same hyperparameter settings.**

Baselines We conduct comprehensive comparisons against both task-aware and task-agnostic context compression methods. Specifically, we include task-aware approaches: LLMingua (Jiang et al., 2023), LongLLMingua (Jiang et al., 2024), DAST (Chen et al., 2025) and AdaKV (Feng et al., 2025), as well as task-agnostic approaches: LLMingua-2 (Pan et al., 2024) and Activation Beacon (Zhang et al., 2025). Additionally, to establish upper-bound references, we fine-tune the original LLMs using the same training data; these are referred to as **LLM-FT** (Llama-3-8B-FT and Qwen-2.5-7B-FT). For a fair comparison under compression constraints, we also evaluate these fine-tuned models with applied compression, denoted

as **LLM-Compressed** (Llama-3-8B-Compressed and Qwen-2.5-7B-Compressed).

Evaluation Benchmark

Primary Long-Context Benchmarks To rigorously evaluate the performance and efficiency of ASP, we leverage two widely recognized long-context evaluation suites: **LongBench** (Bai et al., 2024) and **InfiniteBench** (Zhang et al., 2024). LongBench presents a diverse collection of tasks challenging models’ understanding abilities over extended input sequences, spanning single and multiple document question answering as well as summarization and code tasks, with contexts reaching up to 32K tokens. In parallel, infiniteBench covers a variety of multi-task evaluations where input lengths routinely surpass 200K tokens, placing substantial demands on both context processing and model scalability.

Supplementary Scenario-specific Benchmarks

Beyond core long-context comprehension, we further analyze ASP across several specialized datasets targeting dialog compression, retrieval, and task diversity:

LongBench v2 As a recent addition to the suite of long-context evaluations (Bai et al., 2025), LongBench v2 encompasses six task types spanning input sizes from 8,000 tokens to as high as 2 million. Evaluation adopts a four-way multiple-choice format, using classification accuracy to quantify model performance.

MSC This dataset, adapted from the MemGPT framework (Packer et al., 2023), consists of extended, multi-session human dialogue exchanges, with typical conversation sequences averaging 2,000 tokens.

ShareGPT ShareGPT assembles authentic user-driven ChatGPT interactions from a broad spectrum of conversational and linguistic backgrounds (Chiang et al., 2023).

NaturalQuestions The NaturalQuestions benchmark (Kwiatkowski et al., 2019) replicates a real-world retrieval-augmented QA pipeline, challenging models to locate the correct answer within a cluster of relevant documents. Prompts mirror actual user queries from Google search logs, and answers are human-annotated Wikipedia extracts.

Methods	LongBench v2						
	SingleDoc	MultiDoc	ICL	History	Code	Data	AVG
Qwen-2.5-7B-FT	34.3	25.6	31.0	33.0	36.2	24.0	30.7
Qwen-2.5-7B-Compressed	18.2	19.9	20.8	23.0	32.2	18.3	22.1
LLMLingua (Jiang et al., 2023)	15.3	16.1	10.2	3.1	12.3	12.5	11.6
LLMLingua-2 (Pan et al., 2024)	16.1	15.3	12.1	2.1	13.5	15.6	12.5
LongLLMLingua (Jiang et al., 2024)	16.2	18.3	12.6	6.2	15.6	13.9	13.8
Activation Beacon (Zhang et al., 2025)	20.6	21.6	24.5	5.1	16.0	18.2	17.7
AdaKV (Feng et al., 2025)	<u>31.3</u>	24.5	25.6	<u>18.2</u>	<u>28.0</u>	<u>23.2</u>	<u>25.1</u>
DAST (Chen et al., 2025)	29.1	<u>25.6</u>	21.2	10.9	12.0	15.3	19.0
ASP(Ours)	35.4	27.2	<u>24.7</u>	30.8	30.0	24.3	28.7

Table 5: Experiment results (%) on LongBench v2 (Bai et al., 2025).

Topic Retrieval This framework (Li et al., 2023a) systematically probes models’ ability to manage variable-length multi-turn dialogue and adapt to fluctuating compression requirements, using structured exchanges between human participants and conversational AI agents.

C More Analysis Experiments

C.1 Evaluation on LongBench v2

Table 5 presents the performance on $8\times$ compression constraint of all methods across six sub-categories and the overall average. Our proposed ASP method achieves the highest average score, outperforming previous SOTA approaches under the same compression constraint. In particular, ASP demonstrates strong generalization, achieving the best performance in SingleDoc, MultiDoc, History, Code, and Data tracks. Compared to Qwen-2.5-7B-Compressed, yields substantial improvements in all categories, especially in complex tasks such as MultiDoc (+7.3%) and History (+7.8%). ASP also surpasses explicit compression methods such as LLMLingua and LongLLMLingua by a large margin (average improvement over 10%), highlighting its effectiveness in preserving essential information after aggressive compression. Among specialized long-context optimizations, AdaKV and Activation Beacon achieve competitive results, with AdaKV particularly excelling in ICL tasks. However, ASP maintains stronger and balanced performance across diverse scenario categories, indicating robust adaptability.

C.2 Evaluation on InfiniteBench

Table 7 presents a comprehensive evaluation on InfiniteBench, a challenging benchmark designed

Chunk Size	ROUGE-L			
	2x	4x	8x	16x
128	80.32	76.23	75.01	62.35
256	79.31	78.36	75.31	65.02
512	78.51	80.12	77.12	63.21
1024	79.73	79.27	76.05	64.64
std.	0.66	1.45	0.81	1.08

Table 6: The performance of different chunk size on MSC.

to assess long-context performance with inputs of up to 200K tokens. Our proposed method, ASP, achieves the highest average score of 28.95%. This result not only surpasses all other compression baselines but, remarkably, also exceeds the average performance of the uncompressed, fine-tuned baseline, Qwen-2.5-7B-FT (28.15%). Compared to the strongest compression baseline, AdaKV (26.31%), ASP shows a clear advantage. Specifically, ASP’s superiority is evident as it achieves SOTA performance among all evaluated methods on several demanding tasks, including En.MC (65.20%), Code.D (26.40%), and R.Num (37.46%).

Furthermore, when compared to explicit compression methods like LLMLingua and LongLLMLingua, ASP demonstrates a significant performance leap (+14.3% and +11.97% in average score, respectively), particularly in retrieval-intensive tasks like En.QA and R.Num. These results collectively demonstrate that by leveraging a dynamic budget allocation and a semantic-preserving method, ASP effectively maintains information integrity.

Methods	InfiniteBench							
	En.Sum	En.QA	En.MC	En.Dia	Code.D	Math.F	R.Num	AVG
Qwen-2.5-7B								
Qwen-2.5-7B-FT	14.71	27.75	61.14	6.50	23.86	36.00	27.12	28.15
Qwen-2.5-7B-Compressed	10.53	20.41	56.33	6.00	18.43	16.86	6.78	19.33
LLMLingua	5.31	10.30	48.31	2.03	12.16	5.83	18.61	14.65
LLMLingua-2	6.50	12.12	45.21	1.60	15.32	3.29	19.21	14.75
LongLLMLingua	6.26	12.53	51.20	3.12	18.92	7.21	19.60	16.98
Activation Beacon	10.22	20.75	<u>64.10</u>	<u>5.50</u>	24.62	10.86	15.42	21.64
AdaKV	<u>13.39</u>	27.65	61.57	6.50	<u>25.13</u>	32.65	17.28	<u>26.31</u>
DAST	8.16	23.21	63.52	4.13	20.36	8.71	<u>20.63</u>	21.25
ASP (Ours)	14.19	<u>25.25</u>	65.20	5.00	26.40	<u>29.14</u>	37.46	28.95

Table 7: Evaluation on InfiniteBench (Zhang et al., 2024). **Bold** indicates the best performance, and underline indicates the second-best performance among compression methods for each task.

D Impact of Chunk Size

As shown in Table 6, we conduct experiments to investigate the impact of varying chunk sizes on Llama3-8B’s summarization performance. Under $2\times$ to $16\times$ context scaling constraints with Llama3-8B as the backbone, we observe that the model maintains competitive ROUGE-L scores across all tested chunk size configurations on the MSC benchmark. The low standard deviations demonstrate minimal performance variance across different chunk granularities, highlighting the robustness and reliability of chunked processing with Llama3-8B in practical long-form conversation summarization deployment scenarios.

E Analysis of Dynamic Budget Allocation

To characterize the compression budget allocation mechanism of ASP and uncover its underlying distribution patterns, we construct controlled synthetic test cases based on the NaturalQuestions dataset. Each test case consists of a set of input documents, among which exactly one ground-truth document contains the complete answer corresponding to the given instruction. The core design of ASP is to dynamically allocate compression budgets based on the semantic relevance between each sentence and the input instruction, meaning budget assignment is entirely determined by the semantic content of sentences relative to the task objective, rather than a pre-defined fixed scheme.

In our experimental setup, we fix the ground-truth answer document at the 5th position in the input sequence to eliminate positional bias as a confounding variable in budget allocation. We

set a fixed per-document compression budget of 128 tokens for the uniform allocation baseline, and align the total global compression budget across all compared methods to ensure fair comparison. For answer sentence localization, we adopt an exact ground-truth span matching strategy to identify core answer-bearing sentences. Considering that accurate answer comprehension often relies on adjacent contextual sentences that provide complementary semantic information, we also include these relevant contextual sentences into the target answer span for evaluation. We then conduct a head-to-head comparison between our ASP and the uniform budget allocation strategy (Activation Beacon), focusing on two core dimensions: the amount of compression budget allocated to the ground-truth answer document and answer-bearing sentences, and the corresponding end-to-end retrieval performance measured by Recall score. Quantitative results are summarized in Table 8.

Methods	Doc.	Sen.	Recall Score
Activation Beacon	128	13	39.10
ASP (Ours)	258	35	48.19

Table 8: Average compression budget allocated to the ground-truth answer document (Doc.) and core answer-bearing sentences (Sen.), along with the final Recall score, with the answer document fixed at the 5th position in the input sequence.

Experimental results demonstrate that our method improves performance by allocating more compression budget to answer-bearing documents and sentences (and less to irrelevant ones).

Method	Memory Usage	Compression Latency	Inference Latency
Original Prompt	23,212.5	-	8.7
LLMLingua	30,415.0	5.1	2.3
LLMLingua-2	36,231.2	6.1	2.3
LongLLMLingua	31,521.1	5.2	2.4
Activation Beacon	22,422.9	2.6	1.8
AdaKV	26,315.1	5.6	1.8
DAST	23,215.2	5.8	1.8
ASP	22,633.0	3.8	1.8

Table 9: The comparison of Latency (s) and Memory Usage (MB) of $4\times$ constraint on NQ.

F Efficiency and Resource Analysis

Beyond model accuracy, the practical utility of a long context compression method hinges on its computational efficiency. We therefore analyzed ASP’s performance in terms of GPU memory usage and inference latency. The results, benchmarked against other methods, are presented in Table 9. Compared with the implicit compression method, ASP avoids significant memory overhead while maintaining consistent efficiency. It is worth noting that since explicit compression methods rely on external models for calculation and compression, they result in greater memory usage and higher latency.

G Performance on Short-Context

Table 10 presents results on three widely adopted short-context benchmarks: MMLU (Hendrycks et al., 2021), ARC-Challenge (ARC-C) (Bhaktavatsalam et al., 2021), and GSM8K (Cobbe et al., 2021), using Qwen-2.5-7B as the backbone. ASP consistently outperforms DAST across all three tasks, achieving an average score of 66.0 compared to 64.7 for DAST. ASP recovers 97.3% of the fine-tuned backbone’s average performance, demonstrating that ASP preserves general reasoning ability even when the input length is limited and the compression ratio is inherently higher.

H Multilingual Generalization Ability

Table 11 reports results on three Chinese NLP tasks—single-document QA (SingleDoc), multi-document QA (MultiDoc), and summarization (Summ.)—using Llama-3-8B as the backbone. ASP outperforms DAST on all three tasks, and

Method	MMLU	ARC-C	GSM8K
Qwen-2.5-7B-FT	67.1	60.5	75.8
DAST	65.1	58.2	70.8
ASP (Ours)	66.3	59.1	72.5

Table 10: Performance on short-context benchmarks with Qwen-2.5-7B backbone. The best compression result in each column is **bolded**.

notably surpasses the fine-tuned backbone on both SingleDoc QA (46.9 vs. 45.8) and MultiDoc QA (23.3 vs. 22.3). On summarization, ASP achieves 15.0, slightly below the backbone’s 16.1 but substantially ahead of DAST’s 12.9. Overall, ASP attains the highest average score (28.4), exceeding both the fine-tuned backbone (28.1) and DAST (26.2).

Method	CN SingleDoc	CN MultiDoc	CN Summ.
Llama-3-8B-FT	45.8	22.3	16.1
DAST	45.0	20.6	12.9
ASP (Ours)	46.9	23.3	15.0

Table 11: Performance on Chinese benchmarks with Llama-3-8B backbone. The best compression result in each column is **bolded**.

I FLOPs Calculation

Let s denote the number of tokens in the current block, and let s_{cache} be the cached token count from previous blocks. The quantity h_q indicates the number of query heads, whereas h_k is the number of key/value heads. We denote the hidden representation size by D , the dimensionality of each head by d , the intermediate (feed-forward) expansion size by I , and V for the vocabulary cardinality. We can compute the FLOPs calculation as follows:

$$\begin{aligned}
F_{\text{Att}} &= F_{\text{qkv}} + F_{\text{qk}} + F_{\text{softmax}} + F_{\text{av}} + F_{\text{out}}, \\
F_{\text{qkv}} &= 2sDdh_q + 4sDdh_k, \\
F_{\text{qk}} &= 2h_qs(s + s_{\text{cache}})d, \\
F_{\text{softmax}} &= h_q(s + s_{\text{cache}})^2, \\
F_{\text{av}} &= 2h_qs(s + s_{\text{cache}})d, \\
F_{\text{out}} &= 2sdh_qD
\end{aligned} \tag{15}$$

$$\begin{aligned}
F_{\text{Oth}} &= F_{\text{up}} + F_{\text{gate}} + F_{\text{down}} + F_{\text{lm}}, \\
F_{\text{up}} &= 4 s D I, \\
F_{\text{gate}} &= s I, \\
F_{\text{down}} &= 2 s D I, \\
F_{\text{lm}} &= 2 s D V
\end{aligned} \tag{16}$$

J Additional Parameters

To rigorously quantify the parameter overhead introduced by our ASP method, we present a detailed calculation of additional trainable parameters, taking the Qwen-2.5-7B architecture as the testbed. This architecture adopts Grouped-Query Attention (GQA); to ensure compressed tokens and raw text tokens are processed independently without mutual interference, ASP introduces a set of dedicated projection matrices (W'_q, W'_k, W'_v) exclusively for compressed tokens. The core architectural specifications of Qwen-2.5-7B are summarized in Table 13 for reference.

Based on these specifications, we first derive the per-head dimension of the attention module, the core basis for subsequent parameter calculation. For the GQA mechanism, the per-head dimension is calculated as:

$$d_{\text{head}} = \frac{d_{\text{model}}}{n_{\text{head}}} = \frac{3584}{28} = 128 \tag{17}$$

Projection Matrices (Per Transformer Layer)

For each transformer layer, the dedicated query projection matrix for compressed tokens retains the full hidden dimension consistent with the original model design, while the key and value projection matrices follow the reduced KV head count defined by the GQA architecture. The parameter volume of each projection matrix is calculated as follows:

$$\begin{aligned}
W'_q &= d_{\text{model}} \times d_{\text{model}} \\
&= 3584 \times 3584 \\
&= 12,845,056 \\
W'_k &= W'_v \\
&= d_{\text{model}} \times n_{\text{kv_head}} \times d_{\text{head}} \\
&= 3584 \times 4 \times 128 \\
&= 1,835,008
\end{aligned} \tag{18}$$

Bias Terms (Per Transformer Layer) The bias terms match the output dimensions of the corresponding projection matrices. The total number of

additional bias parameters per layer is calculated as:

$$\begin{aligned}
\text{Bias}' &= d_{\text{model}} + (n_{\text{kv_head}} \times d_{\text{head}} \times 2) \\
&= 3584 + 512 \times 2 \\
&= 4,608
\end{aligned} \tag{19}$$

Total Additional Parameters We first aggregate the additional parameters of a single transformer layer, including the projection matrices and corresponding bias terms:

$$\begin{aligned}
\text{Params}_{\text{layer}} &= W'_q + W'_k + W'_v + \text{Bias}' \\
&= 12,845,056 + 1,835,008 \\
&\quad + 1,835,008 + 4,608 \\
&= 16,519,680 \\
&\approx 16.52 \text{ M}
\end{aligned} \tag{20}$$

Given that Qwen-2.5-7B consists of 28 transformer layers, we further calculate the total additional parameters introduced by ASP across the full model:

$$\begin{aligned}
\text{Total Additional Parameters} &= n_{\text{layer}} \times \text{Params}_{\text{layer}} \\
&= 28 \times 16,519,680 \\
&= 462,551,040 \\
&\approx 462.55 \text{ M}
\end{aligned} \tag{21}$$

Impact on Model Inference Notably, the additional parameters introduced by ASP do not impair the model’s inference performance on raw text inputs, including both generation quality and inference efficiency. The core reason is that the original projection matrices (W_q, W_k, W_v) and the newly introduced dedicated matrices (W'_q, W'_k, W'_v) are completely decoupled in the inference pipeline: the original weight matrices are exclusively used for processing raw text tokens, while the additional matrices are only activated when processing compressed tokens. For standard text-only inference tasks, the additional parameters are not involved in forward computation at all, thus introducing no extra computational overhead.

K Diverse model architectures and scales

The main experiments evaluate ASP on two instruction-tuned models of comparable scale and release date (Llama-3-8B-Instruct, Qwen-2.5-7B-Instruct). To test whether the gains transfer beyond this setting, we re-run the LongBench protocol with two additional backbones: **Llama-2-7B-Chat**, an

Methods	LongBench											
	Single.	Multi.	Summ.	Few.	Code	AVG	Single.	Multi.	Summ.	Few.	Code	AVG
Llama-2-7B						Qwen-2.5-14B						
LLMLingua	20.01	17.39	18.88	43.81	29.82	25.71	26.32	36.35	21.69	58.71	43.63	36.89
LLMLingua-2	19.12	18.42	19.32	45.21	25.31	25.49	23.19	37.20	22.31	59.02	45.32	36.84
LongLLMLingua	23.52	21.42	20.56	46.91	26.24	27.84	31.53	38.61	23.32	60.21	45.31	39.40
Activation Beacon	25.32	26.50	20.50	44.32	29.26	29.17	41.35	40.93	25.14	61.78	39.98	41.97
AdaKV	27.02	25.12	19.13	33.21	28.13	26.41	42.15	38.21	20.21	63.21	42.23	41.13
DAST	23.21	25.22	18.51	46.21	27.21	28.13	40.23	38.71	22.01	65.21	46.32	42.22
ASP(Ours)	29.30	29.91	21.27	60.24	31.52	34.66	43.28	41.76	25.30	71.30	51.23	46.23

Table 12: Evaluation on LongBench in different models and different scale.

Layer (n_{layer})	Hidden Size (d_{model})	Attention Heads (n_{head})	KV Heads ($n_{\text{kv_head}}$)	Attention Style
28	3,584	28	4	GQA

Table 13: Specifications of the Qwen2.5-7B Architecture

earlier-generation model with a different tokenizer and chat template, and **Qwen-2.5-14B-Instruct**, which doubles the parameter count of the main Qwen backbone. Each of the six compression baselines is re-trained on the same data and schedule as ASP. Results are reported in Table 12.

ASP is the strongest compression method at both scales. On Llama-2-7B-Chat, ASP exceeds the next-best baseline (Activation Beacon, 29.17) by 5.49 points on average, with per-task margins ranging from 0.77 on Summarization to 13.33 on FewShot. On Qwen-2.5-14B-Instruct, ASP exceeds the next-best baseline (DAST, 42.22) by 4.01 points on average, with per-task margins ranging from 0.16 on Summarization to 6.09 on FewShot. No baseline wins any of the 10 task–backbone combinations evaluated here.

The advantage does not shrink with model size. On Qwen-2.5-14B-Instruct, the margin over the strongest competing method (+4.01) is in fact larger than that on the 7B backbone (+2.87, Table 1), showing that adaptive sentence-level budget allocation continues to contribute as the backbone scales up.