Comparative Evaluation of Prompt Compression Techniques for Large Language Models: Extractive, Abstractive, and Token-Pruning

Carolina Suyungütmüs[1], **Claudio E. C. Campelo**[1]

[1]Systems and Computing Department
Federal University of Campina Grande (UFCG)
Campina Grande – PB – Brazil

carolina.freitas@ccc.ufcg.edu.br, campelo@dsc.ufcg.edu.br

Abstract. *Large Language Models (LLMs) face significant challenges when processing long prompts regarding computational cost and response quality. This paper presents a comparative evaluation of prompt compression methods, including an exploratory hybrid approach that combines reranking and semantic clustering. Experiments with Llama 3.1 8B Instruct and GPT-4o Mini on LongBench v2 show that accuracy differences among the evaluated methods were generally small relative to the uncertainty of the evaluation, with the standalone reranker achieving the highest observed accuracy at the 8k budget. Extractive methods showed the most favorable accuracy-latency trade-off among the evaluated approaches.*

1. Introduction

Large Language Models (LLMs) have been increasingly applied to complex tasks that require long and detailed prompts, such as multi-document question answering, code assistance, and summarization. These use cases drive the need for extended context handling, prompting the development of models with larger context windows. To meet this demand, modern LLMs have expanded their context windows.

However, processing long inputs increases memory usage, inference cost, and latency [Hooper et al. 2024, Kim et al. 2023]. Additionally, large prompts are susceptible to positional biases, such as the "lost in the middle" effect [Liu et al. 2024], which may degrade output quality. Prompt compression methods [Ali et al. 2024, Li et al. 2023, Xu et al. 2023, Jiang et al. 2023b, Jiang et al. 2023c] aim to reduce prompt length while preserving key information. Yet, few studies offer consistent comparisons across techniques, models, and tasks, limiting their practical applicability. To investigate the trade-offs among prompt length, accuracy, and latency, this work conducts a comparative evaluation of representative extractive, abstractive, and token-pruning techniques, including an exploratory hybrid pipeline that combines semantic clustering with a relevance-based reranker. All approaches are benchmarked on the multiple-choice tasks of the LongBench v2 dataset [Bai et al. 2024], using accuracy and end-to-end latency as evaluation metrics with Llama 3.1 8B Instruct [META AI 2025] and GPT-4o Mini [OPENAI 2025].

2. Related Work

Prompt compression has emerged as a strategy to mitigate the computational and performance limitations of LLMs when processing long contexts. Prompts can be broadly

categorized as hard prompts (discrete prompts) or soft prompts (continuous prompts) [Liu et al. 2023]. Hard prompts consist of human-readable natural language tokens, while soft prompts are continuous vector representations tied to a specific model architecture. This article focuses exclusively on hard prompt compression, since these techniques operate on human-readable text and are directly applicable to black-box LLM APIs, where users have no access to embedding-level manipulation.

Recent work proposes different taxonomies for hard prompt compression. In [Jha et al. 2024], the authors categorize methods as extractive, abstractive, and token-pruning, and compare these paradigms using the original LongBench dataset [Bai et al. 2023], finding that extractive methods, particularly reranker-based approaches, often yield the best results. However, their analysis omits latency and uses metrics such as ROUGE and F1, which have known reliability issues [Novikova et al. 2017]. In contrast, our work employs LongBench v2 [Bai et al. 2024], reformulated for multiple-choice evaluation, enabling clearer accuracy-based comparisons and a better assessment of performance versus cost. In [Li et al. 2025], the authors instead propose two broader sub-categories: filtering approaches, which select or remove content from the prompt, and paraphrasing approaches, which rewrite it into a more concise form, corresponding respectively to the extractive/token-pruning and abstractive categories above. However, this taxonomy is informative but lacks empirical benchmarks and does not address deployment issues such as latency or compatibility with closed-source models, which are examined in this study.

In a related empirical study, an evaluation of six prompt compression methods across 13 datasets is conducted in [Zhang et al. 2025], finding that compression has a greater impact on long-context tasks and that moderate compression can even enhance performance in such scenarios. Some prior works explore hybrid methods by combining rerankers with token pruning or using semantic graphs [Jha et al. 2024, Ali et al. 2024]. However, clustering has received limited attention in prompt compression pipelines. In this context, we evaluate an exploratory approach that integrates semantic clustering with a relevance-based reranker, aiming to reduce redundancy in compressed prompts. Our contributions are twofold: (i) a systematic comparative evaluation across distinct compression paradigms using multiple-choice accuracy alongside end-to-end latency; and (ii) the evaluation of an exploratory hybrid compression pipeline that integrates a relevance-based reranker with semantic clustering.

3. Theoretical Background

Prompt compression aims to reduce prompt length while preserving essential information, improving efficiency without compromising output quality. This study adopts the extractive, abstractive, and token-pruning categorization proposed by [Jha et al. 2024], rather than the broader filtering/paraphrasing distinction of [Li et al. 2025]. The latter groups extractive and token pruning approaches under the single filtering category, losing a distinction that is technically relevant: extractive methods, such as rerankers [Nogueira and Cho 2019, Devlin et al. 2019], operate on larger textual units, such as passages or chunks, whereas token pruning methods, such as LLMLingua [Jiang et al. 2023b], remove individual tokens. Since this study analyzes and compares individual methods, including their latency and behavior, the finer-grained taxonomy of [Jha et al. 2024] is more appropriate for this purpose.

Abstractive methods use summarization to condense input while preserving meaning. RECOMP [Xu et al. 2023] employs a fine-tuned T5-Large model for query-relevant summaries. PromptSAW [Ali et al. 2024] builds a knowledge graph with Vicuna-7B and retrieves relevant content via embedding-based similarity. Extractive methods select relevant passages, such as sentences or paragraphs, without modifying them. It often relies on document reranking techniques, as seen in RAG systems. Methods like BERT-based rerankers [Nogueira and Cho 2019, Devlin et al. 2019] and zero-shot approaches using large language models [Pradeep et al. 2023b, Pradeep et al. 2023a] assign relevance scores to passages based on the query. Token pruning removes low-value tokens based on relevance signals. Selective-Context [Li et al. 2023] uses LLaMA-7B to estimate self-information, while LLMLingua [Jiang et al. 2023b] ranks tokens by perplexity, applying coarse-to-fine pruning. LongLLMLingua [Jiang et al. 2023c] extends this approach to longer inputs by evaluating token relevance relative to supporting documents.

Prompt compression is also related to, but distinct from, Retrieval-Augmented Generation (RAG), which enhances LLM inference by retrieving external content at runtime [Lewis et al. 2020]. Their motivations differ: RAG broadens accessible knowledge by injecting external content, while prompt compression reduces and restructures existing input, without the added system complexity of indexing, retrieval, and synthesis that RAG requires. Interestingly, techniques common in RAG, such as clustering, which promotes diversity and reduces redundancy in retrieval pipelines, have received limited attention in prompt compression. Motivated by this, this work explores the use of clustering to guide content selection within compressed prompts, together with relevance-based reranking.

4. Methodology

This section outlines the methods used to compare prompt compression techniques for LLMs. The goal is to assess their impact on model performance in long-context, multiple-choice tasks, using accuracy and latency as metrics. The following subsections describe the compression methods, evaluation criteria, dataset, models, and computational setup.

4.1. Evaluated Methods

This section describes the six compression methods assessed in this study. To ensure consistency, all methods employ a standard zero-shot prompt during the inference stage, instructing the model to answer the multiple-choice question based on the provided context, except for the iterative approach, which adopts a distinct structure based on content segmentation and sequential processing. The complete prompt templates used in the experiments are available in the repository[1].

For all methods, a fixed token budget was adopted instead of a fixed compression rate, since document lengths in LongBench v2 vary widely, ranging from approximately 8k to 2M words. A fixed compression rate (for example, keeping 20% of the original content) would result in very different prompt lengths across samples, with many still exceeding the models' context windows. In contrast, a fixed budget provides a more consistent target across samples and methods. Target budgets of 8,000 and 2,000 tokens were selected as round figures approximating $8\times$ and $32\times$ compression relative to the

[1] https://github.com/carolmfreitas/prompt-compression-eval/blob/main/prompts/prompts.md

median document length in the evaluated dataset, representing mild and aggressive compression regimes, respectively. Since the actual compression ratio varies across samples and methods, especially for abstractive compression, whose output length cannot be precisely controlled, the achieved ratio is reported for each method in the Results section. A chunk size of 512 tokens was adopted for all chunk-based methods (Reranker, LLMLingua, and Hybrid), following the size used in LongBench v2's own retrieval-augmented evaluation setup and ensuring that these methods operate on the same unit of text. The evaluated prompt compression methods are described below:

Truncated Original: This method serves as a naïve baseline in which no compression is applied. The original document is truncated directly to the target token budget, using the first 8,000 or 2,000 tokens depending on the experimental setting. The truncated context is then provided to the language model using the same inference prompt adopted by the compression methods. The purpose of this baseline is to provide a reference for evaluating whether selecting or transforming the available context offers an advantage over simply retaining the beginning of the document under the same token budget.

Original with Iterations: This baseline is intentionally designed as a naïve sequential-processing strategy. The original document is divided into chunks according to the target token budget, either 8,000 or 2,000 tokens, and each chunk is processed independently using a prompt that instructs the model to answer only when the current chunk contains sufficient information, or to respond "I don't know" otherwise (see repository above). The model receives only the current chunk and has no access to previously processed content. Processing continues while the model abstains and stops as soon as a valid multiple-choice answer is produced. The remaining chunks are therefore not evaluated. If no valid answer is produced after all chunks are processed, the final output is treated as an abstention. This baseline is not intended to approximate a strong full-document aggregation strategy, but rather to provide a simple reference for sequentially exposing the model to the original content without any compression or content selection.

LLMLingua: LLMLingua [Jiang et al. 2023b] was configured with a target token count defined as the global budget minus the fixed prompt-template overhead, computed once by tokenizing the template, for the two budgets evaluated in this study (2,000 and 8,000 tokens). The original text was first segmented into 512-token chunks and passed to LLMLingua, which uses a small language model to estimate token-level perplexity and removes tokens considered less informative through a budget controller and iterative token compression. The question was also provided to the compressor, making the configuration query-aware, unlike the query-agnostic setup typically associated with LLMLingua in prior taxonomies [Jha et al. 2024]. The specified budget acts as a target rather than a strict final length constraint; therefore, LLMLingua outputs can occasionally exceed the nominal budget. Actual compressed context lengths are reported in Table 2.

Reranker: The `mxbai-rerank-large-v1` model [Shakir et al. 2024], a DeBERTa-based architecture [He et al. 2020] fine-tuned for relevance classification, was employed as a reranker model. Given a question–chunk pair, the model outputs a continuous relevance score in the range [0, 1], indicating the degree of alignment between the chunk and the query. The original text was first split into chunks of 512 tokens, which were then scored against the question using the cross-encoder. Chunks were selected in order of score, from highest to lowest, until the token budget was filled. After selection, the chosen chunks were reordered to match their original position in the document before being

passed to the LLM.

Abstractive Compression: The Mistral 7B Instruct model [Jiang et al. 2023a] was used for abstractive compression. The original text was first segmented into chunks of up to 3,000 tokens, keeping each chunk small relative to the full document and reducing the risk of degraded summarization quality from processing very long inputs in a single pass. Each chunk was summarized independently, guided by the question, using a prompt that instructs the model to compress the chunk into a summary of at most N tokens, relevant to the question (see repository above). To approximate the global budgets of 8,000 and 2,000 tokens, the target summary length was divided evenly across the number of chunks and bounded between 64 and 512 tokens per chunk, and the resulting summaries were concatenated in their original order to form the compressed context. Since LLM-generated summary length cannot be precisely controlled, a final truncation step was applied as a safety net whenever the concatenated summaries exceeded the target budget; the frequency with which this step was required is reported in the Results Section. The choice of a 7B-parameter model was motivated by computational efficiency during the compression stage, since larger models, while potentially more capable, would introduce additional cost and overhead.

Reranker with Clustering: We designed an exploratory compression pipeline that combines semantic clustering with cross-encoder reranking, aiming to reduce redundancy while preserving query-relevant content. The original text was first divided into 512-token chunks, following the same chunking scheme used by the other chunk-based methods. Chunk embeddings are generated using `BAAI/bge-base-en-v1.5` model [Xiao et al. 2023], and the chunks are grouped using Agglomerative Clustering. This method initially treats each chunk as an individual cluster and iteratively merges the closest clusters until k clusters remain. Closeness between chunks is measured using cosine distance between their embeddings, while the distance between two clusters is defined as the average pairwise distance between chunks belonging to each cluster (average linkage). The number of clusters k is selected independently for each document using the silhouette coefficient [Rousseeuw 1987], with the search restricted to $k \in [2, \lfloor B_{\text{tokens}}/512 \rfloor]$, where B_{tokens} denotes the token budget. This bounds the search to the approximate number of chunks that could fit within the compressed prompt.

Each cluster is then assigned a relevance score based on the similarity of its chunks to the query, computed as a weighted combination of the maximum and mean similarities. The corresponding weights, together with the cosine-similarity threshold used for deduplication, were selected independently via grid search for each token budget (8,000 and 2,000 tokens) on a disjoint development set of 30 Multi-Document QA examples from LongBench v2, which was not used in the final evaluation. The same configuration, $w_{\text{max}} = 1.0$, $w_{\text{mean}} = 0.0$, with a deduplication threshold of 0.9, was selected for both budgets based on development-set performance. Under this configuration, cluster relevance is determined solely by the most query-relevant chunk in each cluster.

The cluster scores were min-max normalized, and the available token budget was distributed across clusters in proportion to these normalized scores. Each cluster was assigned at least one chunk to maintain topical coverage and any unused budget was then allocated to the highest-scoring remaining chunks across clusters. Within each cluster, chunks were ranked using the `mxbai-rerank-large-v1` cross-encoder [Shakir et al. 2024], the same model used in the standalone Reranker method, and selected un-

til the cluster-specific budget was filled. Finally, the selected chunks were deduplicated, reordered according to their original position in the document, and concatenated to form the compressed prompt.

4.2. Metrics for Comparative Analysis

Prompt compression methods were evaluated using two metrics: accuracy and latency. Accuracy reflects model performance on the multiple-choice dataset, while latency captures the end-to-end processing time of each compression pipeline, including both the compression step and model inference. Recent work has shown that prompt compression should be evaluated using end-to-end latency rather than token reduction alone, since the compression process itself introduces an additional computational cost that may reduce the overall latency gains [Kummer et al. 2026].

4.3. Dataset

Prompt compression techniques were evaluated using the LongBench v2 dataset [Bai et al. 2024], which tests LLMs on long-context tasks requiring deep reasoning across real-world domains. Compared to its predecessor [Bai et al. 2023], LongBench v2 includes 503 multiple-choice questions with contexts ranging from 8,000 to 2 million words. This study focuses on the single-document question answering subset, consisting of 175 questions from domains such as academia, literature, law, finance, government, and narrative text. The multiple-choice format enables objective, accuracy-based evaluation. Other task categories in LongBench v2 were excluded primarily due to computational constraints, particularly for tasks involving longer average contexts, such as multi-document QA and long-dialogue history understanding, while tasks like code and structured data understanding were excluded because they fall outside the scope of natural language prose addressed in this study. Therefore, the single-document setting was adopted as an initial, more controlled scenario for comparing compression techniques.

4.4. Models and Computational Setup

The primary language models used in this study were Llama 3.1 8B Instruct [META AI 2025] and GPT-4o Mini [OPENAI 2025]. Llama 3.1 8B Instruct, developed by Meta, is an open-source model that supports a context window of up to 128,000 tokens and was accessed through a Hugging Face inference provider [Wolf and Debut 2020]. GPT-4o Mini is a proprietary model accessed via OpenAI's API, also supporting context lengths of up to 128,000 tokens. Both models were evaluated under the same fixed input budgets of 8,000 and 2,000 tokens rather than using their maximum context capacities, as described previously in Section 4. This means that, within each budget, the models receive comparable amounts of context across compression methods, reducing input length as a confounding factor in the accuracy comparison.

A preliminary experiment with the Truncated Original baseline on Llama 3.1 8B Instruct found similar accuracy at 8,000 and 100,000 tokens (32.0% and 31.4%, respectively), with the observed difference remaining small relative to the uncertainty of the evaluation. This observation is consistent with prior evidence that increasing context length does not necessarily improve performance on long-context reasoning tasks [Liu et al. 2024]. Based on this result, 8,000 tokens was retained as the less aggressive compression budget instead of adopting a substantially larger context size. All compression-related processing steps, including chunking, embedding generation, clustering, and

reranking, were implemented in Jupyter Notebooks and executed on an NVIDIA A100 GPU with 80 GB of memory. Language model inference for Llama 3.1 8B Instruct, GPT-4o Mini, and Mistral 7B Instruct, used for abstractive compression, was performed through external APIs.

5. Results

This section presents the results of applying different prompt compression methods to long-context questions from the single-document question answering subset of the LongBench v2 dataset, using Llama 3.1 8B Instruct and GPT-4o Mini for evaluation. The metric "Accuracy" refers to the proportion of correct answers over all 175 questions, with a single inference per question. Multiple runs were not performed due to computational constraints. Temperature was set to 0 for all experiments to minimize output variability and support reproducibility. While this does not fully eliminate stochasticity in API-served models, it substantially reduces it, and the large number of distinct questions helps limit the impact of randomness on the overall accuracy estimates.

Table 1. Accuracy results for Llama 3.1 8B Instruct and GPT-4o Mini

	Llama 3.1 8B		GPT-4o Mini	
Method	**8k**	**2k**	**8k**	**2k**
Truncated Original	0.320	0.314	0.280	0.286
Original with Iterations	0.263	0.314	0.291	0.263
LLMLingua	0.297	0.309	0.286	0.240
Reranker	0.337	0.314	0.331	0.286
Abstractive Compression	0.286	0.251	0.263	0.286
Reranker with Clustering	0.326	0.320	0.320	0.309

Table 1 reports the accuracy obtained by each method under the 8,000 and 2,000 token budgets. Reranker achieved the highest observed accuracy at 8,000 tokens for both models (0.337 for Llama, 0.331 for GPT-4o Mini), while Reranker with Clustering achieved the highest accuracy at 2,000 tokens for both (0.320 and 0.309, respectively). The two base models achieved similar performance under equivalent input budgets, with accuracy in most settings remaining relatively close to the 0.25 random-guess baseline for a four-choice task. Given the sample size (N=175), the 95% confidence interval for each accuracy estimate has a margin of approximately 6.5–7.0 percentage points, and these intervals overlap substantially across methods (Figure 1). Since many of the differences between the highest-performing methods are only around 1–3 percentage points, they should be interpreted as observed trends rather than statistically conclusive improvements. Formal pairwise tests would be required to establish statistical significance.

Figure 1 shows that no method clearly dominates across all settings, though Reranker and Reranker with Clustering tend to achieve the highest observed accuracies, while the remaining methods vary more depending on model and budget. Table 2 reports the effective compression ratio, measured as the median ratio between original and compressed context lengths across documents, along with the realized final compressed context length for each method. At 8k, LLMLingua, Reranker, and Reranker with Clustering achieved ratios between 9.8$\times$ and 10.4$\times$, while Abstractive Compression reached 19.4$\times$ because generated summaries often fell below their allocated limits; the truncation safeguard was triggered in only 8 of 175 questions. Reranker and Reranker with Clustering, evaluated under identical budgets, produced similarly sized contexts: 7,734 versus

7,803 tokens on average at 8k, and 1,694 versus 1,844 tokens at 2k, making context length a less likely explanation for the observed differences between them. Isolating the specific contribution of clustering would still require a dedicated ablation study (Section 6). Abstractive Compression showed the widest variation, with 8k outputs ranging from 575 to 7,926 tokens, consistent with the difficulty of controlling summary length.

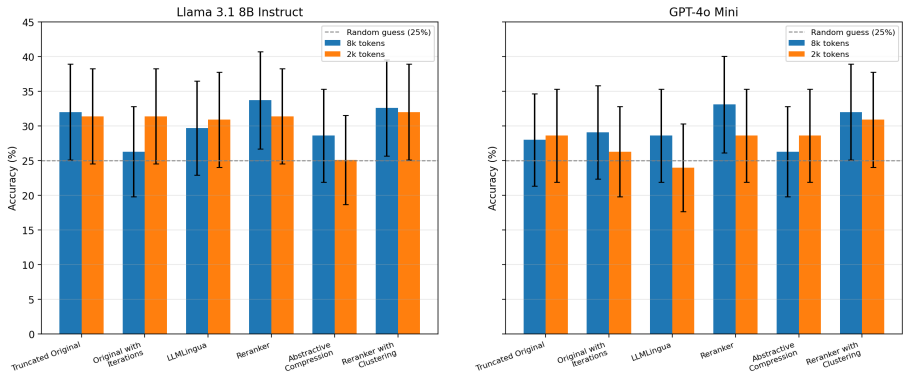


Figure 1. Accuracy by compression method and token budget, for Llama 3.1 8B Instruct and GPT-4o Mini. Error bars denote 95% confidence intervals ($N = 175$). The dashed line marks the random-guessing baseline (25%).

Table 2. Compression ratio and final compressed context length (in tokens) provided to the model, per method and token budget

Method	8k budget			2k budget		
	Ratio	Mean (Med.)	Min–Max	Ratio	Mean (Med.)	Min–Max
LLMLingua	10.4×	7361 (7337)	6413–8421	46.1×	1724 (1697)	1371–2244
Reranker	9.9×	7734 (7694)	7429–7938	47.0×	1694 (1679)	1420–1927
Abstractive Compression	19.4×	3973 (3893)	575–7926	42.3×	1580 (1715)	516–1926
Reranker with Clustering	9.8×	7803 (7829)	7499–7926	43.3×	1844 (1896)	1484–1930

5.1. Results by Compression Paradigm

This section examines the three compression paradigms evaluated in this study. Latency is discussed here in relation to accuracy for each paradigm. Section 5.2 examines its sources in more detail, including the distinction between locally executed and API-based components.

Extractive compression was represented by the standalone Reranker and Reranker with Clustering, the exploratory hybrid pipeline evaluated in this work. At the 8k budget, Reranker achieved the highest observed accuracy for both models (0.337 [59/175] for Llama, 0.331 [58/175] for GPT-4o Mini), slightly above Reranker with Clustering (0.326 [57/175] and 0.320 [56/175]). At the more restrictive 2k budget, this pattern reversed: the Hybrid method reached 0.320 (56/175) versus 0.314 (55/175) for Reranker on Llama, and 0.309 (54/175) versus 0.286 (50/175) on GPT-4o Mini. These differences, corresponding to only one to four questions, remain well within the overlapping confidence intervals and do not indicate that either method consistently outperforms the other. Reranker was also consistently faster, with total latency at 8k of 5.98s (Llama) and 4.43s (GPT-4o Mini), versus 10.37s and 7.44s for the Hybrid method.

Token pruning was represented by LLMLingua, used here in a query-aware configuration. It obtained 0.297 (52/175) at 8k and 0.309 (54/175) at 2k with Llama, and

0.286 (50/175) and 0.240 (42/175) with GPT-4o Mini, at or below the Reranker in the evaluated settings. This may be related to the token-level nature of perplexity-based pruning, which may discard information that appears locally unimportant but is contextually relevant, whereas the Reranker evaluates larger chunks directly against the query. LLMLingua also showed higher compression latency than the Reranker at the 8k budget (6.66s versus 2.39s), as shown in Table 3, without showing a clear advantage in accuracy or compression latency in this setting.

Abstractive Compression, which summarizes the document chunk by chunk using Mistral 7B Instruct, achieved generally lower accuracy than the extractive methods: 0.286 (50/175) and 0.251 (44/175) with Llama at 8k and 2k, and 0.263 (46/175) and 0.286 (50/175) with GPT-4o Mini, a pattern that reversed between models and showed no clear accuracy advantage from summarization. It also introduced by far the largest computational cost (98.70s at 8k, 75.30s at 2k) and the widest variation in final context length, ranging from 575 to 7,926 tokens at 8k, yielding a much higher effective compression ratio (19.4$\times$ versus approximately 10$\times$ for the other methods). These results indicate that, under the constraints considered in this study, abstractive summarization did not show a favorable trade-off relative to extractive selection.

5.2. Latency Analysis

Table 3 reports the average latency per sample for each method, separating compression latency from final generation latency, following the end-to-end latency definition introduced in Section 4. Because the compressed contexts were generated independently of the final inference language model and reused for both models, compression latency is identical for both models for LLMLingua, Reranker, Abstractive Compression, and Reranker with Clustering. Reported latencies combine components executed under different infrastructures: chunking, embedding, clustering, and reranking were run in Google Colab on an NVIDIA A100 GPU, while all language models were accessed through external APIs. Therefore, generation latency reflects not only model computation but also network conditions and provider-side load, which were not controlled in this study. Reported values should therefore be interpreted as indicative of relative latency under this infrastructure, rather than as precise, hardware-normalized measurements; comparisons among methods within the same model setting are more informative than absolute latency values taken in isolation.

Table 3. Latency (in seconds per sample), averaged across all 175 questions.

	8k tokens			2k tokens		
Method	**Comp.**	**Gen.**	**Total**	**Comp.**	**Gen.**	**Total**
Llama 3.1 8B Instruct						
Truncated Original	0.47	3.00	3.47	0.39	3.94	4.33
Original with Iterations	–	5.07	5.07	–	2.39	2.39
LLMLingua	6.66	2.93	9.59	3.63	3.74	7.37
Reranker	2.39	3.59	5.98	2.38	4.74	7.12
Abstractive Compression	98.70	2.80	101.50	75.30	2.35	77.65
Reranker with Clustering	5.21	5.16	10.37	4.94	6.30	11.24
GPT-4o Mini						
Truncated Original	0.04	2.11	2.15	0.04	0.77	0.81
Original with Iterations	–	13.34	13.34	–	13.68	13.68
LLMLingua	6.66	2.08	8.74	3.63	0.70	4.33
Reranker	2.39	2.04	4.43	2.38	0.73	3.11
Abstractive Compression	98.70	1.13	99.83	75.30	0.68	75.98
Reranker with Clustering	5.21	2.23	7.44	4.94	0.68	5.62

Abstractive Compression presented the highest latency, driven mainly by the compression stage (98.70s at 8k, 75.30s at 2k), reflecting the need for multiple generative summarization calls per document. Among the compression methods, the standalone Reranker had the lowest compression latency at both budgets (2.39s and 2.38s), followed by Reranker with Clustering (5.21s and 4.94s) and LLMLingua (6.66s and 3.63s). The Original with Iterations baseline involves no separate compression step, so its latency depends entirely on how many chunks are processed before a valid answer is produced, explaining the higher latency for GPT-4o Mini (13.34–13.68s) than for Llama (2.39–5.07s): Llama stopped after relatively few chunks in most cases, whereas GPT-4o Mini more often continued to later chunks, resulting in more API calls per sample.

6. Threats to Validity and Future Directions

There are some limitations in this study that should be considered. Due to time and computational constraints, each method was evaluated under only two token budgets (8,000 and 2,000), with each configuration run once; the small accuracy differences should therefore be interpreted together with the confidence intervals, rather than as definitive rankings. The hyperparameters of Reranker with Clustering were selected via grid search on only 30 held-out Multi-Document QA questions. Although this set was disjoint from the evaluation data, its small size and different task setting mean that the selected configuration may not represent a robust optimum, and a dedicated ablation study would still be needed to isolate clustering's specific contribution. Reported latency values combine locally executed and API-based components and may be affected by network and provider-side conditions (Section 5). Abstractive Compression was evaluated with only Mistral 7B Instruct, and the evaluation was limited to the Single-Document QA subset of LongBench v2 and two inference models. Future work should evaluate additional token budgets, models, and long-context tasks, along with stronger controlled baselines, including full-context inference when the document fits the model window, no-document and random-selection baselines, and a Reranker condition matched exactly to the Hybrid method's token count. The Original with Iterations baseline intentionally uses a simple early-stopping rule without aggregating predictions across chunks. Future work could evaluate voting or confidence-based aggregation instead. A controlled ablation of the clustering stage, together with evaluations of query expansion and positional reordering, is also left for future work.

7. Conclusion

This work presented a comparative study of prompt compression techniques, analyzing trade-offs between computational efficiency and predictive performance of LLMs in long-context question answering. The standalone Reranker achieved the highest observed accuracy at 8k for both models while maintaining relatively low latency. The Hybrid method achieved comparable accuracy, with differences within the evaluation uncertainty, providing no clear benefit from clustering. Abstractive Compression showed higher latency without a consistent accuracy advantage. Prompt compression remains relevant for reducing input length in long-context LLM applications, while stronger statistical evaluation is needed to establish clear differences between strategies.

Acknowldgements

This work was supported by CNPq's National Institute of Science and Technology in Responsible AI for Computational Linguistics, Information Treatment, and Dissemination (INCT-TILDIAR), grant no. 408490/2024-1.

References

Ali, M. A., Li, Z., Yang, S., Cheng, K., Cao, Y., Huang, T., Hu, L., Yu, L., and Wang, D. (2024). Prompt-saw: Leveraging relation-aware graphs for textual prompt compression. *arXiv preprint arXiv:2404.00489*.

Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., and Li, J. (2023). Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

Bai, Y., Tu, S., Zhang, J., Peng, H., Wang, X., Lv, X., and Li, J. (2024). Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

He, P., Liu, X., Gao, J., and Chen, W. (2020). Deberta: Decoding enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Hooper, C., Kim, S., Mohammadzadeh, H., Mahoney, M. W., Shao, Y. S., Keutzer, K., and Gholami, A. (2024). Kvquant: Towards 10 million context length llm inference with kv cache quantization. *arXiv preprint arXiv:2401.18079*.

Jha, S., Erdogan, L. E., Kim, S., Keutzer, K., and Gholami, A. (2024). Characterizing prompt compression methods for long context inference. *arXiv preprint arXiv:2407.08892*.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M. A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sablayrolles, W. E. (2023a). Mistral 7b.

Jiang, H., Wu, Q., Lin, C.-Y., Yang, Y., and Qiu, L. (2023b). LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376. Association for Computational Linguistics.

Jiang, H., Wu, Q., Luo, X., Li, D., Lin, C. Y., Yang, Y., and Qiu, L. (2023c). Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *arXiv preprint arXiv:2310.06839*.

Kim, S., Hooper, C., Wattanawong, T., Kang, M., Yan, R., Genc, H., Dinh, G., Huang, Q., Keutzer, K., Mahoney, M. W., Shao, Y. S., and Gholami, A. (2023). Full stack optimization of transformer inference: a survey.

Kummer, C., Jurkschat, L., Färber, M., and Vahdati, S. (2026). Prompt compression in the wild: Measuring latency, rate adherence, and quality for faster llm inference. In *European Conference on Information Retrieval*, pages 257–271. Springer.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kuttler, H., and Lewis, M. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *ArXiv*, abs/2005.11401.

Li, Y., Dong, B., Lin, C., and Guerin, F. (2023). Compressing context to enhance inference efficiency of large language models. *arXiv preprint arXiv:2310.06201*.

Li, Z., Liu, Y., Su, Y., and Collier, N. (2025). Prompt compression for large language models: A survey. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7182–7195.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9).

META AI (2025). Llama 3.1 8b instruct. https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct. Accessed: 2025-03-15.

Nogueira, R. and Cho, K. (2019). Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Novikova, J., Dušek, O., Curry, A. C., and Rieser, V. (2017). Why we need new evaluation metrics for nlg. *arXiv preprint arXiv:1707.06875*.

OPENAI (2025). Gpt-4o mini: Advancing cost-efficient intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/. Accessed: 2025-03-15.

Pradeep, R., Sharifymoghaddam, S., and Lin, J. (2023a). Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *arXiv preprint arXiv:2309.15088*.

Pradeep, R., Sharifymoghaddam, S., and Lin, J. (2023b). Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *arXiv preprint arXiv:2312.02724*.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

Shakir, A., Koenig, D., Lipp, J., and Lee, S. (2024). Boost your search with the crispy mixedbread rerank models.

Wolf, T. and Debut, L. (2020). Transformers: State-of-the-art natural language processing. https://huggingface.co.

Xiao, S., Liu, Z., Zhang, P., and Muennighoff, N. (2023). C-pack: Packaged resources to advance general chinese embedding.

Xu, F., Shi, W., and Choi, E. (2023). Recomp: Improving retrieval augmented lms with compression and selective augmentation. *arXiv preprint arXiv:2310.04408*.

Zhang, Z., Li, J., Lan, Y., Wang, X., and Wang, H. (2025). An empirical study on prompt compression for large language models. *arXiv preprint arXiv:2505.00019*.