Efficient Lossless Text Compression with Large Language Models: Enhancing Cross- Lingual and Cross-Domain Applications

Anonymous ACL submission

Abstract

In the era of information explosion, the rapid growth of multilingual and multi-domain textual data poses unprecedented challenges for efficient storage and transmission. Traditional lossless compression methods such as Huffman coding, LZ77, and zlib perform well in certain scenarios but often rely on fixed statistical rules. This limits their ability to capture deeper linguistic structures, especially in complex or domain-specific texts. To address these limitations, we propose two large language model-based lossless text compression methods: **DeepSeekZip** and **LlamaZip**, which respectively integrate DeepSeek-8B and Llama3-8B as predictive models with conventional zlib compression. By leveraging the models' capabilities in modeling complex language patterns, our approach significantly enhances compression performance. Extensive experiments across various languages and text domains demonstrate that DeepSeekZip and LlamaZip consistently achieve over 10% higher compression rates than zlib alone. Notably, DeepSeekZip performs better in Chinese text compression, while both models show comparable results in English. Furthermore, compression effectiveness varies across domains: news and medical texts are compressed more efficiently than legal and technical ones. This highlights the impact of structure, terminology, and contextual dependencies on compression outcomes.

1 Introduction

With the rapid advancement of the information age, the exponential growth of textual data has posed unprecedented challenges to storage and transmission efficiency. Developing effective text compression methods has thus become a pressing need, as it not only reduces storage costs but also significantly improves data transfer performance. Traditional compression algorithms, such as Huffman coding (Huffman, 1952) and LZ77 (Ziv and Lempel, 1977), have performed well in some settings, but their reliance on fixed statistical patterns and rules restricts their ability to capture complex semantic structures. This is especially evident in domain-specific or multilingual texts, where conventional techniques often yield lower compression rates.

Classical lossless text compression methods are generally classified into three categories. (1) Dictionary-based approaches, such as LZ77 and LZW, replace repetitive substrings to improve compression ratio but fail to capture long-range dependencies. (2) Statistical coding methods, such as Huffman and arithmetic coding, model character-level frequency but lack the ability to adapt to complex contextual patterns. (3) Grammar-based methods attempt to infer context-free grammar rules for structural compression. However, the grammar-based-methods suffer from high computational complexity and the absence of efficient random access (Shannon, 1948), making them less practical in real-world applications with dynamic or multilingual data.

To overcome the representational limitations of traditional methods, recent work has explored the use of deep neural networks to capture complex contextual structures. Goyal et al. (2018) proposed DeepZip, which leverages RNN-based conditional probability modeling combined with arithmetic coding. Transformer-based architectures have further improved the modeling of long-range dependencies. RWKV (Peng et al., 2023) introduces linear attention mechanisms to retain contextual information while reducing inference overhead. Similarly Perceiver (Jaegle et al., 2021) extends input scalability via iterative attention.

Building on this trend, researchers have integrated pre-trained language models into compression pipelines. Li et al. (2021) proposed two Transformer-based strategies for enhanced text compression: Explicit Text Compression (ETC) and Implicit Text Compression (ITC). ETC employs a separate sequence-to-sequence compression model with attention mechanisms to extract key semantic components, which are concatenated with the original input to enrich the encoding process. In contrast, ITC integrates the compression module directly into a non-autoregressive decoder, enabling end-to-end training and supporting seamless integration with downstream tasks such as machine translation. Both approaches effectively reduce input redundancy while maintaining task performance and demonstrate strong transferability across various NLP tasks. However, challenges persist. ETC necessitates a considerable amount of supervised data to define "key semantics," resulting in significant annotation costs. Although ITC offers greater flexibility, it may neglect low-frequency yet critical information, which could lead to semantic loss. Furthermore, ETC-generated compressed sequences often demonstrate low interpretability, and ITC's architecture demands extensive customization to integrate with existing models. While these methods show promise for semantic compression, they still fall short of achieving robust lossless compression performance and broad applicability.

Recent efforts have also explored leveraging BERT for lossless compression. Öztürk and Mesut (2024) proposed MLMCompress, which utilizes BERT's bidirectional contextual prediction to estimate token distributions and integrates it with arithmetic coding. The model achieves up to 38% higher compression on English datasets compared to NNCP and a 42% improvement in multilingual tasks. Furthermore, MLMCompress operates up to 35 times faster than GPTZip (Nishi et al., 2023) in certain settings, with 20% faster compression and up to 180% faster decompression. This demonstrates the efficiency and practicality of contextual modeling for compression. However, the use of masked language modeling inherently imposes limitations. BERT demonstrates continuity in prediction and effectively models long-range semantics. Furthermore, MLMCompress lacks optimization for Chinese and other non-Latin languages.

As large language models (LLMs) continue to evolve, their exceptional language modeling capabilities have made them increasingly attractive for compression tasks. Pre-trained models such as LLaMA and GPT learn high-quality token distributions over massive corpora, enabling accurate next-token prediction for entropy coding. Valmeekam et al. (2023) introduced LLMZip, which combines LLaMA with arithmetic coding and achieves state-of-the-art performance on English text. However, most LLM-based methods to date primarily focus on English, and their performance on Chinese or domain-specific content remains underexplored.

To address these limitations, we propose a novel lossless text compression framework based on two competitive open-source LLMs: **LlamaZip** (based on LLaMA3-8B (AI, 2024b)) and **DeepSeekZip** (based on DeepSeek-8B (AI, 2024a)). Our method utilizes the LLM to predict token-level probabilities and subsequently applies standard `zlib` compression to the residual errors. This hybrid approach balances expressiveness and efficiency. **Figure 1** illustrates the overall architecture of our compression framework, highlighting the two-stage process from semantic modeling to entropy coding. We evaluate the proposed methods on both Chinese and English corpora, encompassing diverse domains such as news, law, medicine, and technology. Experiments results demonstrate that our approach outperforms traditional zlib-based compression by over 10% on average, with DeepSeekZip achieving superior performance on Chinese text, and both models perform comparably on English datasets.

Our contributions are threefold:

(1) We utilize a lossless text compression framework that integrates large language models with traditional entropy coding, thereby enhancing compression rate and generalizability.

(2) We conduct a comprehensive analysis of model performance across various languages and domains, emphasizing differences in adaptability and robustness.

(3) We demonstrate the practical benefits of LLM-based compression through extensive experiments, providing new insights into language-aware compression in multilingual contexts.

2 Related Work

In recent years, text compression has made significant strides, with researchers exploring diverse strategies to enhance compression efficiency and modeling capability. This section reviews five major methodological directions in the field.

2.1 Dictionary-based Compression.

Classic dictionary-based methods, such as LZ77 and LZ78 (Ziv and Lempel, 1977), replace repeated

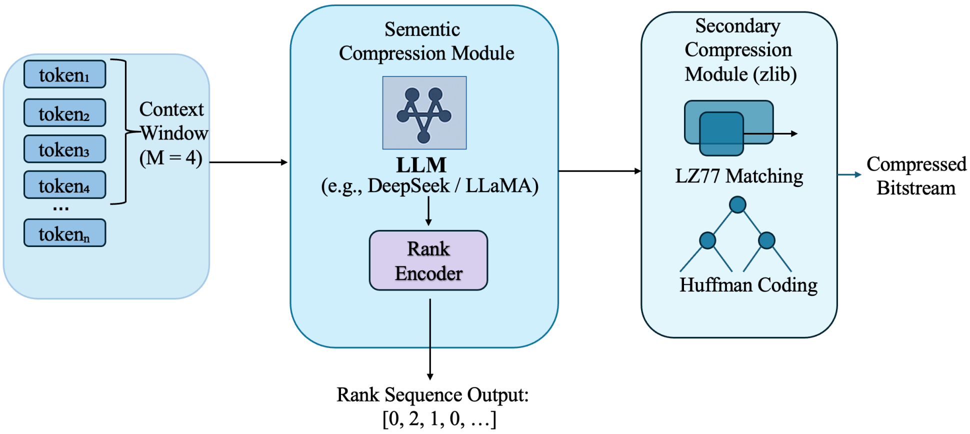

Figure 1: Semantic-to-Entropy Compression Pipeline. The LLM module generates token probability rankings, which are subsequently compressed using entropy-based methods such as zlib.

substrings with shorter references. More recently, L3TC (Zhang et al., 2025) leverages a lightweight Transformer variant (RWKV) combined with arithmetic coding to achieve lower latency and complexity. Chen et al. (Chen et al., 2003) further proposed parallel dictionary lookup mechanisms to enhance matching speed. However, dictionary-based approaches face challenges in dynamic dictionary construction, particularly in large-scale or online compression settings where memory and lookup overhead can become prohibitive. Multidictionary systems can boost compression ratios but significantly increase decoding complexity, which limits their use in resource-constrained or real-time devices.

2.2 Statistical Compression.

Statistical methods reduce redundancy by encoding symbol frequencies. Huffman coding (Huffman, 1952) and arithmetic coding (Witten et al., 1987) are foundational examples. Recently, k-th order context models augmented with neural networks, such as RWKV (Peng et al., 2023), have improved probabilistic estimation. Additionally, hybrid pipelines such as BWT+MTF+RLE (Adiego et al., 2007) perform well in compressing structured data. However, high-order models require large context buffers, resulting in increased memory usage and reduced adaptability to data streams with changing distributions.

2.3 Grammar-based Compression.

Grammar-based methods aim to infer context-free grammar rules that capture the underlying structure of text. Techniques based on smallest grammar approximation and grammar induction under higher-order entropy constraints have shown promise (Gańczorz, 2018). However, these approaches suffer from computational intractability (NP-hardness) and often relying on heuristics that struggle with long sequences. Moreover, the resulting compressed data structures lack efficient random access, and their decoding speed is suboptimal, which limits their deployment in time-sensitive scenarios.

2.4 Neural Arithmetic Coding.

Combining neural models with arithmetic coders has recently emerged as a promising direction in the field. L3TC (Zhang et al., 2025) employs adaptive context modeling to dynamically adjust token probabilities, approaching entropy bounds while alleviating the bottlenecks of arithmetic coding through parallel block encoding. Nonetheless, arithmetic coding requires high-precision floating-point operations, rendering it unsuitable for low-end devices or real-time compression tasks due to its computational demands and sensitivity to numerical instability.

2.5 LLM-based Compression.

Large language models (LLMs) such as LLaMA and GPT have garnered significant attention in the field of text compression due to their robust contextual modeling capabilities. LLMZip (Valmeekam et al., 2023) demonstrates the feasibility of combining LLaMA3-8B with arithmetic coding to achieve superior compression ratios on English corpora. However, LLMZip suffers from extreme latency—compressing just 10MB of text can take up to 9.5 days—raising serious concerns about its practicality.To address this challenge, Mittu et al. (2024) introduced FineZip, a novel LLM-based compression framework incorporating online memory and dynamic context windows. By leveraging parameter-efficient fine-tuning (PEFT), FineZip reduces compression time from 9.5 days to 4 hours—a 54-fold improvement—while maintaining comparable compression ratios. It also surpasses traditional algorithms by achieving approximately 50% better compression rates on benchmark datasets. This work demonstrates that while LLM-based compression remains computationally intensive, its performance bottlenecks can be mitigated through architectural and optimization-level innovations.

2.6 Motivation and Gap.

Building upon these insights, we propose two LLM-based compression systems: **LlamaZip** and **DeepSeekZip**, which are based on LLaMA3-8B and DeepSeek-8B, respectively. Unlike prior works that primarily focus on English, we conduct a comprehensive evaluation across both English and Chinese corpora, covering multiple domains including news, medicine, law, and technology.

3 Method

We propose a two-stage compression framework that integrates the semantic modeling capabilities of large language models (LLMs) with traditional lossless compression techniques. Our approach comprises of a **Semantic Compression Module** and a **Secondary Compression Module**. This section provides a detailed overview of both modules and their mathematical formulations.

3.1 Semantic Compression Module

This module leverages LLMs to estimate the conditional probability distribution of each token based on the context, subsequently encoding the rank of the true token. The objective is to utilize the LLM's language understanding to convert raw tokens into a more compressible rank sequence.

3.1.1 Context Modeling and Temperature Scaling

Given a token context $X = [x_{t-M}, \ldots, x_{t-1}]$, the LLM predicts the next token probability distribution:

$$P(x_t \mid X) = \mathrm{LLM}(X) \tag{1}$$

To control sampling diversity, a temperature parameter T is introduced to rescale the output distribution:

$$Q_i = \frac{P_i^{1/T}}{\sum_j P_j^{1/T}} \tag{2}$$

When $T \to 0$, the model becomes deterministic:

$$Q_i = \begin{cases} 1 & \text{if } P_i = \max(P) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

We set $T = 0$ in all experiments to ensure deterministic outputs and enable reproducible, lossless decompression.

3.1.2 Rank-Based Encoding

Instead of storing tokens directly, we record the rank of each ground-truth token under the LLM's predicted distribution. The rank is computed as:

$$R_t = \mathrm{rank}(P(w \mid X)) - 1 \tag{4}$$

where w is the true token at position t, and **the ranking is zero-based**—the most probable token has rank 0.

The final encoded sequence becomes:

$$\{R_1, R_2, \ldots, R_n\} \tag{5}$$

Example: Consider the sentence "Artificial intelligence is rapidly advancing," tokenized as "Artificial," "intelligence," "is," "rapidly," "advancing." Using a context window of size $M = 4$, the LLM receives the first four tokens and predicts the fifth. Suppose the predicted token probabilities are:

Candidate Token	Probability
"advancing"	0.50
"evolving"	0.30
"expanding"	0.15
"adapting"	0.05

Table 1: Predicted token distribution.

Here, "advancing" is the correct token, ranked first in terms of probability. **Using zero-based indexing, its final rank is 0.** Thus, this token is encoded as 0. Repeating this procedure over the entire sequence results in a compact, rank-encoded representation.

This design utilizes the LLM's comprehension of token dependencies to semantically compress text while maintaining reversibility.

3.2 Secondary Compression Module

Although the rank sequence is more compact than the original text, further compression is achievable through entropy coding. We utilize the Zlib library, which implements the DEFLATE algorithm by combining LZ77 and Huffman coding.

3.2.1 LZ77 Encoding

LZ77 minimizes redundancy by identifying repeated substrings within a sliding window. It encodes each match as a tuple $\langle \text{length}, \text{distance} \rangle$ or outputs a literal character when no match is found.

Let W represent the sliding window size, S denote the search buffer, and p the current position. The match length L and distance D satisfy:

$$L = \max \{ l \mid S[p-l : p] = S[p-D-l : p-D] \}, \quad D \leq W \tag{6}$$

3.2.2 Huffman Coding

Huffman coding assigns shorter bit sequences to more frequent symbols. Let $p(s)$ be the probability of symbol s, and $l(s)$ its code length. Then:

$$H(S) \leq \mathbb{E}[l(s)] \leq H(S) + 1 \tag{7}$$

where the entropy $H(S)$ is defined as:

$$H(S) = -\sum_s p(s) \log_2 p(s) \tag{8}$$

3.2.3 Bit Cost Estimation

To estimate the final compressed size, we compute the total number of bits in the output bitstream. After the rank sequences are passed through Huffman encoding, the bit length of each symbol s is denoted by $l(s)$, as introduced in Equation 7. The overall bit cost of the Zlib-compressed stream can be expressed as:

$$B_{\text{zlib}} = B_{\text{meta}} + \sum_{s \in \text{stream}} l(s) \tag{9}$$

Here, B_{meta} accounts for the overhead introduced by Zlib, including format headers, Huffman tree descriptors, and other control structures. The summation term represents the total number of bits consumed by symbol-level encoding in the stream.

Together, the semantic module reduces the entropy of the input by utilizing high-level language structures, while the Zlib module further minimizes bit-length through statistical coding. This hybrid approach achieves strong compression ratios while preserving exact reconstructibility.

4 Experiments

4.1 Datasets

We use the `text8` corpus from http://mattmahoney.net/dc/text8.zip as our base dataset. The `text8` corpus is a cleaned, compressed excerpt of Wikipedia articles, which is widely employed in benchmarking text compression. To evaluate multilingual and domain-specific performance, we categorize `text8` into four segments—**Medical**, **Legal**, **News**, and **Technical**—and use GPT-4o to generate corresponding Chinese translations. Note that we focus on small-scale data, with each text sample limited to around 10 KB, to examine the effectiveness of lightweight compression strategies.

The four domain subsets are described as follows:

(1) **Medical Text**: Includes medical literature, case reports, and clinical descriptions, rich in domain-specific terminology and syntactic complexity.

(2) **Legal Text**: Contains excerpts from legal codes and court decisions, featuring highly formal and nested grammatical structures.

(3) **News Text**: Covers journalistic commentary and investigative reports, characterized by journalistic vocabulary and structured phrasing.

(4) **Technical Text**: Consists of manuals and technical documentation, containing numerous domain-specific and instruction-oriented expressions.

All datasets undergo identical preprocessing steps, including text cleaning, tokenization, and normalization, to ensure fairness and consistency across all compression evaluations.

4.2 Experimental Setup

The experiments proceed in the following stages (see Figure 2):

(1) **Segmentation and Tokenization:** Each document is split into chunks suitable for LLM input, followed by tokenization using each model's native tokenizer.

(2) **Prediction and Compression:** DeepSeek or LLaMA models are used to compute

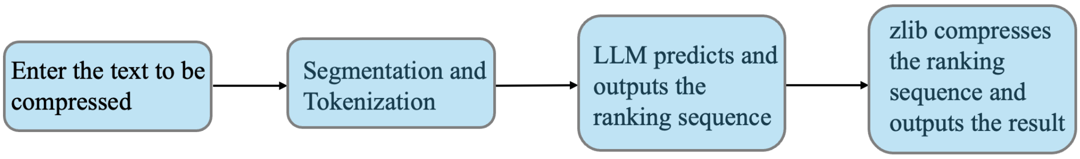

Figure 2: The experimental workflow of our compression framework.

next-token probability distributions and ranking sequences. These sequences are then compressed using `zlib` to produce final bitstreams.

(3) **Evaluation:** Compression efficiency is measured using the *space-saving rate* η, computed as:

$$\eta = \frac{\text{Original Size} - \text{Compressed Size}}{\text{Original Size}} \times 100\%. \quad (10)$$

We conduct ablation experiments by varying the memory length parameter $M \in \{128, 256, 512, 1024\}$ for each LLM to study the effect of context window size on compression rates. For each domain and language (Chinese and English), we select 5 representative 10 KB samples from `text8`, compress them using both DeepSeek and LLaMA models, and report the averaged results.

Hardware and Software Environment. All experiments are conducted on identical GPU servers. The compression pipeline is implemented in Python using PyTorch for model loading and inference.

4.3 Experimental Results

We conducted comparative experiments using both LLaMA and DeepSeek models. For each, we selected 5 Chinese and 5 English samples per domain (medical, legal, news, and technical) from the `text8` dataset. Each sample was approximately 10 KB in size. We then compressed these samples using each language model and computed the average space-saving rate. Additionally, we performed ablation studies by varying the memory length parameter $M \in \{128, 256, 512, 1024, 2048\}$, to evaluate the impact of context length on compression performance across different domains.

4.3.1 Experimental Variables

(1) **Text Domains and Language Types**

Model	M = 128	M = 256	M = 512	M = 1024
DeepSeekZip	42.09%	49.19%	56.19%	56.19%
LlamaZip	40.91%	48.72%	55.26%	55.26%
`zlib`	46.78%	46.78%	46.78%	46.78%

Table 2: Average space-saving rate (%) for different memory lengths M across all test samples.

We selected samples from four domains in the `text8` dataset. For each domain, we extracted **both Chinese and English** text samples of equal size to enable cross-linguistic comparison. Within each domain-language pair, we selected 5 thematically consistent samples and reported the average space-saving rate as the representative result.

(2) **Comparison Models and Baseline**

We applied two large language models: **DeepSeekZip** and **LlamaZip**. Additionally, we used the traditional **zlib** compression algorithm as a baseline to compare against the LLM-based approaches.

(3) **Main Variable: Memory Length M**

We conducted experiments with memory lengths $\mathbf{M} = \{128, 256, 512, 1024, 2048\}$. The objective was to observe how compression performance changes with increasing memory length, under a fixed input size constraint.

4.3.2 Effect of Memory Length on space-saving rate

We compare the average space-saving rate η achieved by DeepSeekZip, LlamaZip, and the traditional `zlib` algorithm across all test samples. The results are summarized for memory lengths $\mathbf{M} = \{128, 256, 512, 1024, 2048\}$. Since `zlib` is independent of model memory length, its space-saving rate remains constant across different settings.

As shown in Figure 3, the average space-saving rate of **DeepSeekZip** and **LlamaZip** increase significantly with the growth of memory length M,

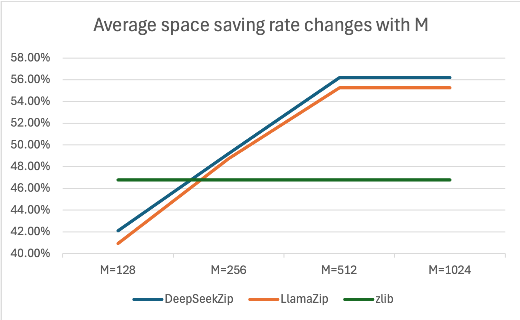

Figure 3: Average space-saving rate (%) as memory length M increases for DeepSeekZip, LlamaZip, and `zlib`.

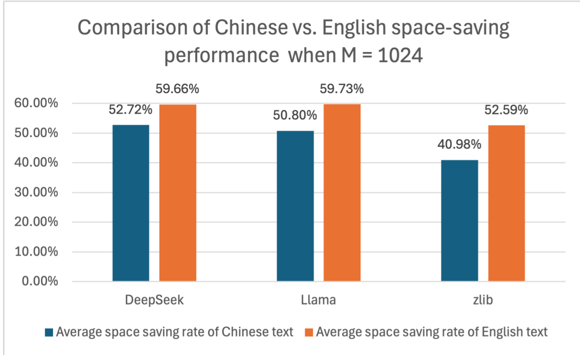

Figure 4: Comparison of Chinese vs. English space-saving performance when memory length $M = 1024$.

while the compression ratio of `zlib` remains constant. Detailed observations are as follows:

(1) **From 128 to 256:** DeepSeekZip improves from 42.09% to 49.19%, and LlamaZip from 40.91% to 48.72%, while zlib remains at 46.78%. This demonstrates that longer memory significantly enhances LLMs' contextual modeling, improving compression.

(2) **From 256 to 512:** DeepSeekZip and LlamaZip continue to improve to 56.19% and 55.26%, respectively, while zlib stays flat. This highlights the value of long-range context in LLM-based compression.

(3) **From 512 to 1024:** Space-saving rates plateau for both LLMs, with no further gains. This indicates diminishing returns beyond 512 tokens for fixed-length inputs (e.g., 10 KB).

Overall Observations: DeepSeekZip and LlamaZip consistently enhance space-saving rates as memory length grows, especially within $M = 128 \sim 512$, while zlib maintains constant due to its lack of contextual modeling. DeepSeekZip generally outperforms LlamaZip, though the gap narrows with larger M. Beyond $M = 512$, both models exhibit saturation, reflecting marginal gains.

Marginal Effect and Potential Causes:

(1) Emergence of Marginal Returns: A significant improvement is observed when M increases from 128 or 256 to 512. However, the space-saving rate gains diminish beyond $M = 512$ and become negligible up to $M = 2048$. This suggests that in moderate-length documents, once the model captures most of the redundant or structured information, further extending the memory yields limited additional benefits.

Model	Medical (zh)	Legal (zh)	Technical (zh)	News (zh)	Avg.
DeepSeekZip	56.03%	48.64%	48.36%	57.83%	52.72%
LlamaZip	51.83%	48.07%	46.88%	56.41%	50.80%
zlib	40.31%	38.76%	38.28%	46.55%	40.98%

Table 3: Space-saving rates of different models on **Chinese** texts in four domains ($M = 1024$).

(2) Possible Causes: There are two likely explanations. First, if the input document is relatively short or lacks recurring patterns, then increasing memory size may yield diminishing returns once redundancy is fully addressed. Second, Transformer-based models may experience de- creasing efficiency in utilizing extended con- text, resulting in weaker marginal benefits.

4.3.3 Comparison of Chinese vs. English Space-Saving Performance (M = 1024)

To evaluate the effectiveness of different models on Chinese and English texts, we compared the space-saving rate of DeepSeekZip, LlamaZip, and zlib under a fixed memory length $M = 1024$. The following tables present the average space-saving results across four domains (see Tables 3 and 4).

(1) Chinese Text Analysis DeepSeekZip achieves the highest average space-saving rate for Chinese texts at 52.72%, outperforming LlamaZip (50.80%) and zlib (40.98%). Across all four Chinese domains, DeepSeekZip consistently outperforms the other methods, demonstrating superior effectiveness in addressing the unique characteristics of the Chinese language. In contrast, zlib yields the lowest space-saving rate and remains unaffected by memory length, highlighting its lack of semantic awareness and context modeling.

Model	Medical (en)	Legal (en)	Technical (en)	News (en)	Avg.
DeepSeekZip	59.67%	58.08%	60.63%	60.27%	59.66%
LlamaZip	59.85%	58.72%	60.45%	59.90%	59.73%
zlib	52.51%	51.64%	52.80%	53.41%	52.59%

Table 4: Space-saving rates of different models on **English** texts in four domains ($M = 1024$).

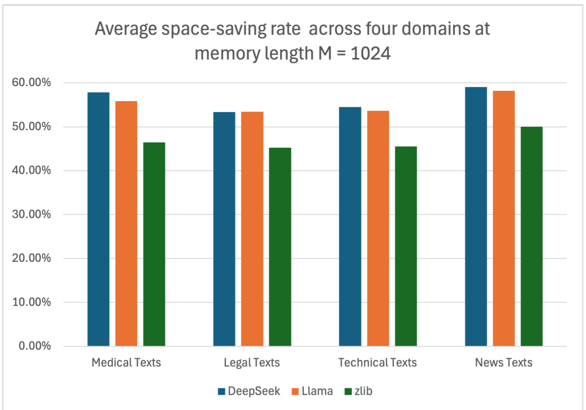

Figure 5: Performance differences of each model in different fields at memory length $M = 1024$.

(2) English Text Analysis DeepSeekZip and LlamaZip perform nearly identically on English texts, with average space-saving rates of 59.66% and 59.73% respectively. Both models significantly outperform zlib (52.59%), confirming the advantage of LLMs in modeling English semantics and capturing long-range dependencies.

(3) Summary DeepSeekZip shows a clear advantage over LlamaZip when processing Chinese texts, while both methods perform similarly on English texts. Both LLM-based approaches significantly outperform zlib in all instances, particularly with Chinese data. These results underscore the robustness of DeepSeekZip and LlamaZip across various languages and domains, showcasing notable improvements in space-saving efficiency.

4.3.4 Domain-Specific Space-Saving Performance (M = 1024)

To compare the performance of **DeepSeekZip**, **LlamaZip**, and **zlib** across different domains. The average space-saving rate is computed for each model within each domain.

Analysis:

(1) **Medical Texts:** DeepSeekZip achieves the highest space-saving rate at 57.85%, outperforming LlamaZip (55.84%) and zlib (46.41%). This indicates DeepSeekZip's superior capability in capturing medical expressions.

(2) **Legal Texts:** DeepSeekZip and LlamaZip perform nearly equally with rates of 53.36% and 53.40%, respectively—both significantly better than zlib's 45.20%. This suggests both LLM-based methods handle the formal and complex syntax of legal documents more effectively than traditional compression methods.

(3) **Technical Texts:** DeepSeekZip slightly outperforms LlamaZip (54.50% vs. 53.67%), and both surpass zlib (45.54%), reflecting strong performance in compressing structured and terminology-rich technical content.

(4) **News Texts:** DeepSeekZip shows superior performance with a 59.05% space-saving rate, ahead of LlamaZip (58.16%) and zlib (49.98%), highlighting its strength in modeling the variability and redundancy present in journalistic writing.

Model	Medical	Legal	Technical	News
DeepSeekZip	57.85%	53.36%	54.50%	59.05%
LlamaZip	55.84%	53.40%	53.67%	58.16%
zlib	46.41%	45.20%	45.54%	49.98%

Table 5: Average space-saving rate (%) across four domains at memory length $M = 1024$.

Overall, DeepSeekZip demonstrates consistent superiority across all domains, particularly in medical and news texts. LlamaZip follows closely and also significantly outperforms the traditional zlib approach, which demonstrates the weakest results across all categories. This reinforces the advantages of LLM-based methods in domain-sensitive semantic modeling for compression.

5 Conclusion

Text compression is evolving toward LLM-based systems that leverage deep semantics. We propose **DeepSeekZip** and **LlamaZip**, combining LLM prediction with entropy coding. Results show that longer memory lengths (e.g., $\mathbf{M = 512}$) improve compression by capturing semantic redundancy. Our methods outperform traditional baselines while remaining lossless. Future work includes structure-aware modeling and efficient decoding like *FineZip* (Mittu et al., 2024).

Limitations

While our framework demonstrates strong performance across languages and domains, several limitations remain. First, LLM-based compression methods are computationally expensive compared to traditional algorithms like zlib, especially during inference. This restricts their deployment in real-time or resource-constrained environments. Second, although we evaluated multilingual settings, our non-English experiments relied on machine-translated corpora, which may not fully capture the complexities of native-language structures. Third, the current approach does not incorporate adaptive memory or dynamic context resizing, which could be important for handling variable-length documents more efficiently. Lastly, our evaluation focuses on average space-saving rates, without measuring decoding latency or memory overhead, which are also critical in practical applications. Future work could address these issues through model optimization, native multilingual pretraining, and efficiency-aware benchmarks.

Ethics Statement

Our work builds upon publicly available large language models (DeepSeek and LLaMA) and standard compression libraries (zlib). All datasets used, including the `text8` corpus and its domain-specific subdivisions, are derived from open-access sources. To enable multilingual evaluation, we generate corresponding Chinese texts from the English corpus using GPT-4o machine translation, followed by basic validation to ensure semantic consistency. These translations are used for research purposes only, and no personally identifiable or sensitive user data is involved.

We recognize that automatically translated data may not fully capture the linguistic richness and diversity of native Chinese corpora, which could limit generalizability and introduce subtle biases. Future work should explore more authentic and diverse Chinese datasets to support broader language fairness.

Moreover, while LLM-based compression methods offer significant gains in efficiency, they come with non-negligible environmental costs due to the computational demands of large-scale models. Our experiments are conducted using existing pre-trained models to minimize additional carbon footprint. No human subjects were involved in this study, and no privacy or safety concerns arise from our methodology.

We advocate for continued efforts toward sustainable, inclusive, and responsible NLP research.

References

Joaquín Adiego, Gonzalo Navarro, and Pablo de la Fuente. 2007. Using structural contexts to compress semistructured text collections. *Information Processing & Management*, 43(3):769–790.

DeepSeek AI. 2024a. Deepseek-r1-distill-llama-8b. https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-8B. Accessed: 2024-05-19.

Meta AI. 2024b. Meta-llama-3-8b. https://huggingface.co/meta-llama/Meta-Llama-3-8B. Accessed: 2024-05-19.

David Chen, Enoch Peserico, and Larry Rudolph. 2003. A dynamically partitionable compressed cache. In *Proceedings of the Singapore-MIT Alliance Symposium*. Technical Report, MIT Laboratory for Computer Science.

Michał Gańczorz. 2018. Entropy bounds for grammar compression. *arXiv preprint arXiv:1804.08547*.

Mohit Goyal, Kedar Tatwawadi, Shubham Chandak, and Idoia Ochoa. 2018. Deepzip: Lossless data compression using recurrent neural networks. *arXiv preprint arXiv:1811.08162*.

David A Huffman. 1952. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. 2021. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR.

Zuchao Li, Zhuosheng Zhang, Hai Zhao, Rui Wang, Kehai Chen, Masao Utiyama, and Eiichiro Sumita. 2021. Text compression-aided transformer encoding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3840–3857.

Fazal Mittu, Yihuan Bu, Akshat Gupta, Ashok Devireddy, Alp Eren Ozdarendeli, Anant Singh, and Gopala Anumanchipalli. 2024. Finezip: Pushing the limits of large language models for practical lossless text compression. *arXiv preprint arXiv:2409.17141*.

Erika Nishi, Hayato Mizutani, and Takuya Hashimoto. 2023. Gptzip: Lossless text compression using gpt. https://github.com/erika-n/GPTzip. Accessed: 2025-05-08.

Emir Öztürk and Altan Mesut. 2024. Learning-based short text compression using bert models. *PeerJ Computer Science*, 10:e2423.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, and 1 others. 2023. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*.

Claude E Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Chandra Shekhara Kaushik Valmeekam, Krishna Narayanan, Dileep Kalathil, Jean-Francois Chamberland, and Srinivas Shakkottai. 2023. Llmzip: Lossless text compression using large language models. *arXiv preprint arXiv:2306.04050*.

Ian H Witten, Radford M Neal, and John G Cleary. 1987. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540.

Junxuan Zhang, Zhengxue Cheng, Yan Zhao, Shihao Wang, Dajiang Zhou, Guo Lu, and Li Song. 2025. L3tc: Leveraging rwkv for learned lossless low-complexity text compression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13251–13259.

Jacob Ziv and Abraham Lempel. 1977. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343.

A Full Results and Compression Pipeline

A.1 Experimental Results by Domain and Memory Length

Table 6 presents the full results of our compression experiments. We report the average space-saving rates (%) across four domains—Medical, Legal, Technical, and News—in both Chinese and English, using memory lengths $M \in \{128, 256, 512, 1024\}$. Each score is averaged over five samples per configuration.

A.2 Visualization

To visually complement the compression process, we include a diagram illustrating the semantic-to-entropy pipeline (Figure 1). The figure shows how the LLM module predicts token-level probability rankings, which are then transformed into rank sequences and further compressed using entropy-based algorithms such as `zlib`.

A.3 Experimental environment

We used DeepSeek-8B and LLaMA3-8B as frozen pre-trained models, each with approximately 8B parameters. Experiments were conducted on A100 (80GB) GPUs. No additional fine-tuning or training was performed.

B Licenses of Used Artifacts

We summarize the licenses of all third-party artifacts used in this work:

- **DeepSeek-R1-Distill-Llama-8B**: Released under the **Apache 2.0 License**, available at `https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-8B`.
- **Meta-Llama-3-8B**: Released under Meta's custom license for research use, available at `https://huggingface.co/meta-llama/Meta-Llama-3-8B`.
- **zlib**: A classical entropy coding library released under the **Zlib License**.

All licenses permit the use of these models and tools for non-commercial research purposes. No modifications were made to the original released artifacts.

Model	Medical		Legal		Technical		News		Avg.
	zh	en	zh	en	zh	en	zh	en	
M = 128									
DeepSeek	35.73	47.84	33.79	50.18	35.47	49.50	37.09	47.15	42.09
Llama	34.05	47.66	32.34	50.06	32.42	48.51	35.33	46.92	40.91
zlib	40.31	52.51	38.76	51.64	38.28	52.80	46.55	53.41	46.78
M = 256									
DeepSeek	47.56	54.11	41.04	53.86	41.09	54.10	47.12	54.67	49.19
Llama	44.35	54.17	43.18	54.27	39.77	54.42	45.47	54.15	48.72
zlib	40.31	52.51	38.76	51.64	38.28	52.80	46.55	53.41	46.78
M = 512									
DeepSeek	56.03	59.67	48.64	58.08	48.36	60.63	57.83	60.27	56.19
Llama	51.83	59.85	48.07	58.72	46.88	60.45	56.41	59.90	55.26
zlib	40.31	52.51	38.76	51.64	38.28	52.80	46.55	53.41	46.78
M = 1024									
DeepSeek	56.03	59.67	48.64	58.08	48.36	60.63	57.83	60.27	56.19
Llama	51.83	59.85	48.07	58.72	46.88	60.45	56.41	59.90	55.26
zlib	40.31	52.51	38.76	51.64	38.28	52.80	46.55	53.41	46.78

Table 6: Space-saving rates across domains, languages, and memory lengths.

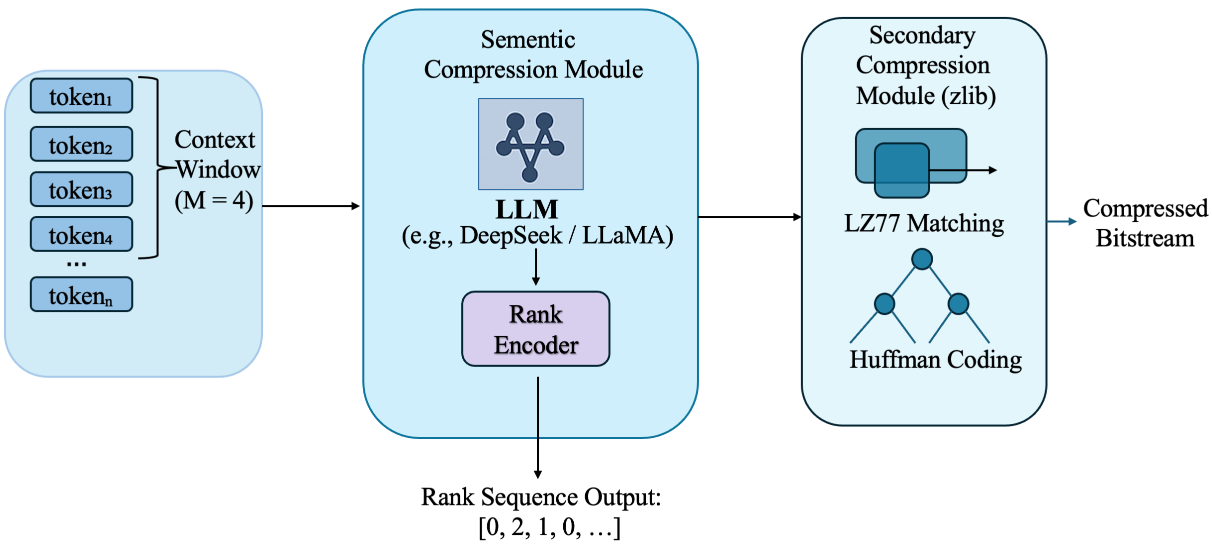

Figure 6: Semantic-to-Entropy Compression Pipeline. The LLM module outputs token rankings, which are subsequently encoded into compact sequences and compressed via entropy coding.