

ICPC: In-context Prompt Compression with Faster Inference

Anonymous ACL submission

Abstract

Despite the recent success of Large Language Models (LLMs), it remains challenging to feed LLMs with long prompts due to fixed size of LLM inputs. As a remedy, prompt compression becomes a promising solution by removing redundant tokens in the prompt. However, the usage of LLM in the existing works requires additional computation resources and leads to memory overheads. To address it, we propose ICPC (In-context Prompt Compression), a novel and scalable prompt compression method that reduces the length of prompt adaptively. The key idea of ICPC is to calculate the probability of each word appears in the prompt using encoders and calculate information carried by each word through information function, which effectively reduces the information loss during prompt compression and increases the speed of compression. Empirically, we demonstrate that ICPC can effectively compress long texts of different categories and thus achieve better performance and speed on different types of NLP tasks.

1 Introduction

Large Language Models (LLMs) have revolutionized natural language processing, demonstrating remarkable capabilities across a wide array of tasks such as text generation, question answering, and semantic understanding. However, LLMs encounter significant challenges when processing long prompts or extended contexts, as their computational cost scales quadratically with sequence length due to the attention mechanism. This limitation hinders their ability to handle real-world applications requiring long-context understanding, such as document summarization, multi-turn conversations, and knowledge-intensive reasoning. To address these scalability issues, researchers have explored efficient attention mechanisms, sparse representations, and distributed processing approaches,

enabling LLMs to manage long contexts more effectively and maintain their performance on extended inputs. However, existing approach often utilize LLM to compress texts, which leads to the challenge of memory overhead.

There are numerous existing approaches aimed at addressing the memory overhead of large language models (LLMs) during inference, including LoRA (Hu et al., 2021) and Sparse Attention (Child et al., 2019). Additionally, several prompt compression methods, such as Selective Context (Li et al., 2023), have demonstrated strong performance with limited memory requirements. Unlike these existing methods, which primarily focus on utilizing large language models to compress prompts, we propose a novel approach to prompt compression leveraging language models with millions of parameters. Our approach not only offers faster inference compared to existing methods but also achieves excellent compression performance.

The motivation for the proposed approach stems from the structure of transformer encoders. First, the pretraining of transformer encoders enables them to effectively capture and understand the context of words. Second, transformer encoders typically have significantly fewer parameters compared to large language models, resulting in a 10x to 100x increase in inference speed over existing compression methods, which is shown in Appendix 5.

In this paper, we introduce a novel framework named **In-Context Prompt Compression** (ICPC). ICPC operates by first segmenting sentences or paragraphs at both the phrase and clause levels. It then uses a pretrained transformer encoder to calculate the loss associated with each word in the sentence, removing words to make paragraph concise without loss of essential meaning. We conducted experiments across various encoder architectures to demonstrate the generalizability and superiority of our method over existing approaches.

2 Preliminaries

2.1 Entropy

Entropy, rooted in information theory, measures the average level of uncertainty or surprise in a probability distribution over linguistic units. In Natural Language Processing (NLP), each token $t \in T$ (e.g., word or subword) is associated with a probability $p(t)$ reflecting how frequently t appears in a given context. Formally, the Shannon entropy of p is defined as

$$H(p) = - \sum_{t \in T} p(t) \log p(t),$$

where $p(t)$ captures the likelihood of observing token t . Entropy thus provides a principled way of quantifying how “spread out” or “concentrated” the distribution is. In language modeling, for example, a lower entropy typically indicates a model that makes confident predictions about the next token, whereas higher entropy signals more uncertainty. Extensions like cross-entropy and KL divergence further leverage this concept to compare model-generated distributions against ground-truth or target distributions. By minimizing these metrics during training, NLP models are encouraged to produce more accurate and reliable predictions, ultimately improving language understanding and generation.

2.2 Transformer Encoder

The transformer encoder is a fundamental building block in transformer-based architectures, leveraging self-attention to capture global contextual information in sequences. Each encoder layer consists of two core components: a Multi-Head Self-Attention (MHSA) module and a position-wise Feed-Forward Network (FFN). Formally, given input embeddings \mathbf{X} , MHSA computes attention scores among all positions in the sequence, yielding contextualized representations. Residual connections and layer normalization are then applied, preserving the flow of information and stabilizing training. The result is passed through the FFN, which comprises two linear transformations with an activation function in between. Stacking multiple layers allows the transformer encoder to refine representations progressively, making it particularly effective for NLP tasks such as language modeling, classification, and machine translation.

3 Method

Our method compresses the input text for LLM by removing redundant words and phrases to increase the ability of LLM understanding on long context understanding and reduce the computational cost without extra large language models (e.g., GPT-4, Llama-3) needed.

3.1 Participle

If participle-based filtering is directly applied at the word level, it may fail to capture the nuanced structure of linguistic patterns. Therefore, beyond word-level processing, we implement participle-based filtering at both phrase and clause levels. In our framework, a participle unit serves as a fundamental building block, encompassing words, phrases, or clauses depending on the granularity required. To support this, we group tokens with their contextual embeddings into participle units, enabling the model to retain richer semantic and syntactic information during the filtering process.

3.2 Loss Computation

Given a list of words $C = (x_{i-k}, \dots, x_{i+k})$, the loss function by removing x_i is defined as

$$L(x_i) = \alpha * \text{sim}(\mathbf{x}_{i,k}, \mathbf{x}_i) - H(x_i | \mathbf{x}_{i,k}) \quad (1)$$

where $\mathbf{x}_{i,k}$ is defined as

$$(x_{i-k}, \dots, x_{i-1}, x_{i+1}, \dots, x_{i+k}) \quad (2)$$

This loss function provides a mechanism to balance the trade-off between compression and the preservation of information.

3.3 Redundant Words Removal

To minimize the loss of information while retain the original key information during during compression, we rank the units based on their calculated loss in descending order and compute the p -th percentile of loss among all units.

$$L_p = np.\text{percentile}([L(x_0), \dots, L(x_n)], p) \quad (3)$$

Then, we remove all the lexical units will loss greater or equal than L_p and merge the remaining words together as the output C' :

$$C' = \{x_i | L(x_i) < L_p\} \quad (4)$$

This adaptive filtering strategy provides a more flexible mechanism to discard redundant units

while retaining the most essential content. By dynamically adjusting the threshold, the method ensures that the selection process accounts for variations in the loss distribution.

4 Experiments

In this section, we present the performance of our method against other state-of-the-art approaches with both quantitative and qualitative analysis. For all experiments, we simulate a evaluation environment using an EC2 p4d.24xlarge virtual machine (VM) instance on AWS, which has 8 NVIDIA A100 GPUs, 96 vCPUs, and 320 GB main memory. Other important information including operation system version, Linux kernel version, and CUDA version is summarized in Table ??.

4.1 Experimental Settings

For a fair comparison, we adopt the same input format (tokens, phrases, or sentences), and inference settings for all experimental conditions. For parameters specific to our method, such as the compression ratio and lexical unit granularity, we tune these to optimize efficiency without degrading performance. Baseline methods such as random compression and full context usage retain their respective default configurations. All metrics (e.g., BLEU) are computed under identical evaluation protocols to ensure consistency across tasks, with multiple runs for stochastic outputs to mitigate randomness.

4.1.1 Datasets

Our method reduces redundancy in the input context, enabling efficient processing of very long contexts for LLMs. However, existing benchmarks such as SQuAD (Rajpurkar et al., 2018) and Piqa (Bisk et al., 2020) are mostly single-round question-answer datasets with short question length, which is not appropriate to evaluate our proposed method. Therefore, we compile four datasets with long context and conversations to demonstrate the effectiveness of our method. The statistics and compilation details are presented in the appendix.

Wikipedia: A dataset containing articles from Wikipedia, a free online encyclopedia covering an extensive range of topics across numerous domains, including history, science, arts, and culture. For our experiments, we utilize the introductory sections of each article, which provide concise and informative summaries of the topics.

arXiv Papers: A dataset comprising academic papers from arXiv, spanning diverse fields such as

computer science, physics, mathematics, and biology. Due to the length of many arXiv papers, we focus on processing the abstract and the introduction sections for our experiments, ensuring a balance between content depth and computational efficiency.

Reddit: A dataset derived from user-generated posts and comments on Reddit, a social media platform organized into communities covering a wide variety of topics, from technology and science to hobbies and entertainment. For our experiments, we use a curated subset of posts and their associated top-level comments to capture meaningful discussions and interactions.

4.1.2 Models

To demonstrate the generalization of our method on different settings, we test the method on different kinds of language models. We evaluate our method on BERT, RoBERTa, XLNet, ALBERT, T5, DEBERTA. The detailed description of these encoders is shown in Appendix A.2.

4.1.3 Metrics

We evaluate our method using BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), TF-IDF Similarity (Sparck Jones, 1972), Jaccard Similarity (Jaccard, 1912), BERTScore (Zhang et al., 2019), compression Rate (Roberts, 2003), and Flesch-Kincaid readability score (Kincaid, 1975). The detailed description of these encoders is shown in Appendix A.3.

4.2 ICPC Performance Evaluation

In this section, we offer an in-depth evaluation of the performance enhancements attributed to our In-context Prompt Compression (ICPC). Specifically, we evaluate ICPC against 4 baseline methods, including Original, Random Deletion, Selective Context and LLM-Lingua. These baseline methods do not require model training.

As shown in Table 1, it’s evident that the ICPC has led to marked improvements across all metrics. Notably, ICPC saw the BLEU score increase with ratio 0.6 on metric BLEU from 42.6 to 43.4, highlighting ICPC’s ability in understanding the in-context information of prompt. Random deletion underperforms mainly due to its limited importance understanding for words in the prompt, which led developers to make selection on prompt according to the importance score of different words. LLM-Lingua performs well on different metrics using

Model	Ratio	METEOR	BLEU	ROUGE			BERTScore		
				rouge1	rouge2	rougeL	Precision	Recall	F1
Original	-	49.3	44.1	66.9	49.8	56.6	86.7	90.9	88.8
Random Deletion	0.8	43.9	40.8	61.9	43.1	52.1	81.9	81.3	81.5
	0.6	41.6	36.1	58.1	37.5	47.5	78.1	79.5	79.2
	0.4	39.2	31.8	53.9	32.4	43.2	75.0	75.1	75.4
Selective Context	0.8	45.1	41.7	62.7	43.8	52.7	82.9	82.9	82.9
	0.6	42.6	37.2	58.8	38.2	48.2	79.2	80.4	79.8
	0.4	39.9	32.6	54.3	33.2	44.3	76.1	76.0	76.0
LLMLingua	0.8	45.2	42.1	61.8	43.1	52.1	83.4	83.1	83.2
	0.6	43.1	37.8	58.4	37.9	47.9	78.0	79.1	78.5
	0.4	40.1	32.9	55.0	33.4	44.8	76.3	75.6	75.9
ICPC	0.8	45.3	42.7	63.1	44.2	52.8	83.6	83.5	83.5
	0.6	43.4	38.0	59.1	38.4	48.5	79.1	80.6	79.8
	0.4	40.6	33.1	55.2	33.6	45.1	76.4	76.2	76.3

Table 1: Performance comparison of baseline methods. The ICPC utilize BERT as encoder. ICPC can boost the performance compared to traditional methods using large language models.

Model	Ratio	Training time (ms)
Selective Context	0.8	46.3
	0.6	49.5
	0.4	52.2
LLMLingua	0.8	45.2
	0.6	43.1
	0.4	40.1
ICPC	0.8	10.3
	0.6	13.4
	0.4	16.6

Table 2: Average compression time comparison of baseline methods. The ICPC utilize BERT as encoder. ICPC can significantly reduce compression time compared to traditional methods.

token-level iterative compression and distribution alignment between models, though it faces slowdowns due to the usage of large language models.

4.3 Compression Speed Analysis

As shown in Table 2, the ICPC achieves fastest compression speed compared to Selective Context and LLMLingua. It is obvious that the ICPC reduce compression time multiple times by using models of smaller size. It is noted that the performance does not degrade as our method utilizes context information instead of previous information, such as GPT and Llama, etc.

4.4 Readability Analysis

As is shown in Figure 2, the ICPC compress the prompt through calculation of the importance of each words in the sentence and remove the unnecessary words to make the prompts shorter and more concise without loss of information. The compressed text also preserves good readability and make it easier for people to grasp the meaning of the long prompt.

4.5 Scalability on Very-Long Texts

To align with the input constraints of language models, the prompts are segmented into fixed-length

token chunks. This segmentation ensures compatibility while enabling the model to process information effectively. During our experiments, we observed that the input limit of 512 tokens, as defined by BERT, is sufficient to capture the necessary contextual information for downstream tasks. This token limit ensures that the most important context is preserved while allowing the model to make informed decisions about the selection of relevant tokens. Moreover, this approach balances efficiency and performance, avoiding unnecessary computational overhead while maintaining the integrity of the contextual information. We found that even within this constraint, BERT’s representation capabilities are robust enough to handle a wide range of tasks, providing reliable outputs that meet our experimental objectives.

4.6 Comparison of Different Encoder Configurations

We conducted experiments of our method using five different encoder architectures: BERT, RoBERTa, XLNet, ALBERT, T5, and DeBERTa. As presented in Table 3, the performance across these encoders shows minimal variation, with different models demonstrating distinct strengths across various evaluation metrics. The detailed information

Basketball is a team sport in which two teams, most commonly of five players each, opposing one another on a rectangular court, compete with the primary objective of shooting a basketball (approximately 9.4 inches (24 cm) in diameter) through the defender's hoop (a basket 18 inches (46 cm) in diameter mounted 10 feet (3.048 m) high to a backboard at each end of the court), while preventing the opposing team from shooting through their own hoop. A field goal is worth two points, unless made from behind the three-point line, when it is worth three. After a foul, timed play stops and the player fouled or designated to shoot a technical foul is given one, two or three one-point free throws. The team with the most points at the end of the game wins, but if regulation play expires with the score tied, an additional period of play (overtime) is mandated.

Basketball team sport two teams five players each opposing rectangular court objective shooting basketball defender's hoop basket backboard end court preventing opposing team shooting through own hoop field goal two points behind three-point line After foul timed play stops player fouled designated shoot technical foul given one, two or three one-point free throws most points end game wins regulation play expires score tied period of play (overtime) is mandated.

Figure 1: Texts before and after compression. Yellow represents words with higher importance. up: text before compression. Down: text after compression.

Model	Ratio	METEOR	BLEU	ROUGE			BERTScore		
				rouge1	rouge2	rougeL	Precision	Recall	F1
Original	-	49.3	44.1	66.9	49.8	56.6	86.7	90.9	88.8
BERT	0.8	45.3	42.7	63.1	44.2	52.8	83.6	83.5	83.5
	0.6	43.4	38.0	59.1	38.4	48.5	79.1	80.6	79.8
	0.4	40.6	33.1	55.2	33.6	45.1	76.4	76.2	76.3
RoBERTa	0.8	45.1	42.2	63.0	44.5	52.4	83.3	83.6	83.4
	0.6	43.3	38.1	59.3	38.6	48.5	79.6	80.4	80.0
	0.4	40.5	33.6	55.1	33.3	45.3	76.2	76.3	76.2
XLNet	0.8	45.5	42.4	63.1	44.3	52.1	83.6	83.2	83.4
	0.6	43.3	38.4	59.4	38.2	48.2	79.7	80.2	79.9
	0.4	40.7	33.2	55.7	33.5	45.5	76.4	76.9	76.6
ALBERT	0.8	44.9	42.4	62.8	44.1	52.4	83.2	83.2	83.2
	0.6	42.9	38.5	59.5	38.6	48.1	79.5	80.3	79.9
	0.4	40.8	33.2	55.6	33.3	45.6	76.9	76.1	76.5
T5	0.8	45.5	42.2	61.9	43.5	52.3	83.8	83.5	83.4
	0.6	43.2	37.3	58.2	38.0	47.2	78.4	79.2	78.8
	0.4	40.6	32.5	55.9	33.8	44.9	76.2	75.7	75.9
DeBERTa	0.8	45.9	42.9	61.2	43.8	52.7	83.3	83.6	83.4
	0.6	43.4	37.5	58.6	38.9	47.3	78.5	79.2	78.8
	0.4	40.2	32.3	55.7	33.6	44.6	76.1	75.5	75.8

Table 3: Performance comparison of ICPC using different encoders. ICPC utilizes 6 different types of models as encoder: BERT, RoBERTa, XLNet, ALBERT, T5 and DeBERTa.

Model	Year	Pretraining Objective	Parameters	Training Data Size
BERT	2018	Masked Language Modeling (MLM) + NSP	110M (Base)	16 GB (BooksCorpus, Wiki)
RoBERTa	2019	MLM (Improved)	125M (Base)	160 GB (OpenWebText, etc.)
XLNet	2019	Permuted Language Modeling	117M (Base)	32 GB (BooksCorpus, Wiki)
ALBERT	2019	MLM + Sentence Order Prediction	12M (Base)	16 GB (BooksCorpus, Wiki)
T5	2019	Text-to-Text Generation	220M (Base)	750 GB (C4 dataset)
DeBERTa	2020	MLM + Replace Token Detection	140M (Base)	160 GB (similar to RoBERTa)

Table 4: Information of different encoders

of each encoders is shown in Figure 1.

5 Conclusion

In-context Prompt Compression is a good improvement with regarding to problems presented by large language models, such as memory overhead and

computation speed. In this paper, we present ICPC (In-context Prompt Compression), a novel and scalable prompt compression method that improves the performance without utilisation of large language models. We formulate the important tokens selection task as an information calculation task. Extensive experiments over various comparison methods

on multiple benchmarks with different encoders demonstrate that our proposed ICPC can greatly boost the performance of existing hard prompt compression methods and also achieve faster convergence speed.

6 Ethics Statement

This research did not involve any studies with human participants or animals performed by any of the authors. Therefore, no ethical approval was required for this study. All data and materials were collected in a manner consistent with ethical guidelines, ensuring no ethical concerns are present.

References

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Paul Jaccard. 1912. The distribution of the flora in the alpine zone. 1. *New phytologist*, 11(2):37–50.

JP Kincaid. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. *Chief of Naval Technical Training*.

Z Lan. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Yucheng Li, Bo Dong, Chenghua Lin, and Frank Guerin. 2023. Compressing context to enhance inference efficiency of large language models. *arXiv preprint arXiv:2310.06201*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don’t know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Martyn Roberts. 2003. Benefits and challenges of variable compression ratio (vcr). Technical report, SAE Technical Paper.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

Zhilin Yang. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

A Appendix

In this appendix, we describe the detailed comparison between LLM and LM, detailed descriptions of encoders and metrics, etc.

A.1 Comparison between LLM and LM

To show the superior of using encoder to compress prompts, we take BERT Base and GPT-3 as example, which is shown in Table 5.

A.2 Detailed description of encoders

BERT: BERT (Devlin, 2018) introduces bidirectional context into NLP tasks using a transformer architecture. It is pretrained on large corpora and fine-tuned for tasks like question answering. BERT’s widespread influence and robust baseline performance make it essential for comparison in experiments.

RoBERTa: RoBERTa (Liu, 2019) optimizes BERT by removing the next-sentence prediction objective and using larger datasets and longer training. Its superior performance on multiple benchmarks makes it a strong candidate for evaluating enhanced pre-training techniques.

XLNet: XLNet (Yang, 2019) employs permutation-based training to capture bidirectional context without masking. Its ability to outperform BERT on tasks like GLUE demonstrates the advantages of its innovative pretraining objective.

ALBERT: ALBERT (Lan, 2019) reduces memory and computation costs via parameter sharing and embedding factorization while maintaining strong benchmark performance. Its efficiency and scalability make it ideal for resource-constrained settings.

T5: T5 (Raffel et al., 2020) frames all NLP tasks as text-to-text problems using a unified transformer architecture. Its state-of-the-art results across diverse benchmarks highlight its flexibility and generalization capabilities.

DeBERTa: DeBERTa (He et al., 2020) enhances BERT with disentangled attention and improved position encoding. Its strong performance on GLUE and SQuAD makes it valuable for evaluating innovative attention mechanisms.

A.3 Detailed description of metrics

BLEU: BLEU evaluates machine translation by measuring n-gram overlap between machine and reference translations, emphasizing precision. Its consistency and simplicity make it a standard for translation benchmarks.

ROUGE: ROUGE measures overlap between predicted and reference summaries using recall-based metrics like n-grams and longest common subsequence. It is widely adopted for summarization due to its focus on content retention.

TF-IDF Similarity: TF-IDF computes text similarity by balancing term frequency against inverse document frequency, highlighting distinctive terms. Its interpretability makes it useful for document comparison.

Jaccard Similarity: Jaccard similarity compares sets by their intersection-over-union ratio, often used for token or n-gram overlap. Its simplicity makes it effective for assessing basic textual similarity.

BERTScore: BERTScore uses contextual embeddings to evaluate semantic alignment between texts. Its ability to capture deep contextual meaning makes it ideal for tasks requiring nuanced comparisons.

Compression Rate: Compression Rate evaluates text conciseness by comparing original and compressed sizes. It is effective for gauging redundancy and assessing information density.

Flesch-Kincaid Readability Score: This score evaluates readability based on sentence length and word complexity. It is essential for analyzing the accessibility of generated or written content.

METEOR: METEOR combines precision and recall with synonyms and stemming for flexible text alignment. Its nuanced approach is valuable for evaluating natural language generation and translation.

Feature	Transformer Encoder (BERT Base)	LLM (GPT-3)
Model Type	Encoder-only Transformer	Decoder-only Transformer
Parameter Count	~ 110 million	~ 175 billion
Number of Layers	12	96
Hidden Size	768	12288
Inference Time*	~ 20 ms / 128 tokens	~ 2 s / 128 tokens

Table 5: Comparison between a typical Transformer encoder model (BERT Base) and a large language model (GPT-3). *Inference times are approximate and depend on hardware (e.g., single GPU vs. multi-GPU) and implementation details.