Prompt Compression for Large Language Models: A Survey

Zongqian Li*, **Yinhong Liu***, **Yixuan Su, Nigel Collier**
University of Cambridge
{zl510, yl535, ys484, nhc30}@cam.ac.uk

Abstract

Leveraging large language models (LLMs) for complex natural language tasks typically requires long-form prompts to convey detailed requirements and information, which results in increased memory usage and inference costs. To mitigate these challenges, multiple efficient methods have been proposed, with prompt compression gaining significant research interest. This survey provides an overview of prompt compression techniques, categorized into hard prompt methods and soft prompt methods. First, the technical approaches of these methods are compared, followed by an exploration of various ways to understand their mechanisms, including the perspectives of attention optimization, Parameter-Efficient Fine-Tuning (PEFT), modality integration, and new synthetic language. We also examine the downstream adaptations of various prompt compression techniques. Finally, the limitations of current prompt compression methods are analyzed, and several future directions are outlined, such as optimizing the compression encoder, combining hard and soft prompts methods, and leveraging insights from multimodality. [1]

1 Introduction

As task complexity increases, prompts become longer for LLMs to accommodate more detailed requirements, contextual information, and in-context learning (ICL) examples. Lengthy prompts reduce inference speed, increase memory costs, and negatively impact user experience. Current methods for improving LLM efficiency can be broadly categorized into model-centric and prompt-centric methods (Wan et al., 2024). Model-centric approaches, such as parameter pruning (Ma et al., 2023) and quantization (Dettmers et al., 2022), focus on optimizing the model itself. In contrast, prompt-centric methods, including prompt compression (Li et al., 2023) and prompt design (Shin et al., 2020), aim to improve the efficiency of LLM by lowering the complexity of input. Prompt-centric methods typically introduce minimal or no changes to the parameters of the LLM, allowing them to be used in a modular way. Thus, these methods, especially

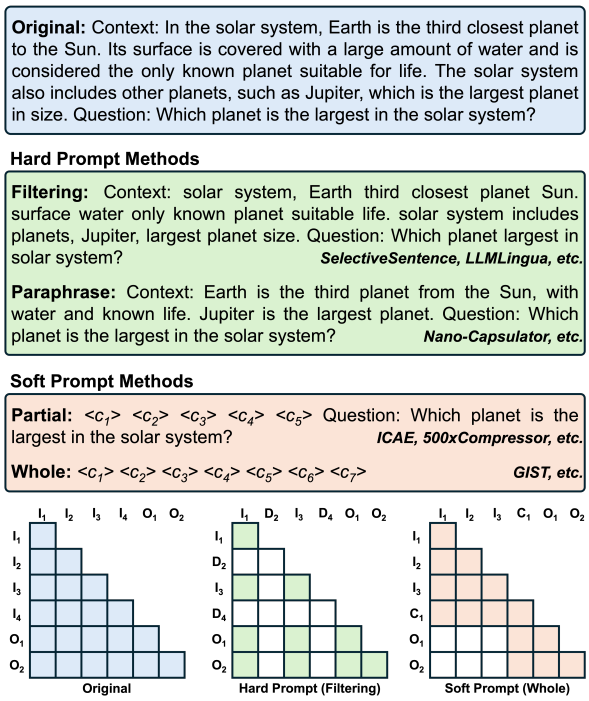

Figure 1: Illustrative examples of prompt compression methods. Hard prompt methods remove low-information tokens or paraphrase for conciseness. Soft prompt methods compress text into a smaller number of special tokens, $< c_n >$. The grids below visualize attentions, where the y-axis represents the sequence of tokens, and the x-axis shows the tokens they attend to. (Bottom left) Original prompt: Each token attends to all previous tokens. (Bottom middle) Hard prompt (filtering): Each token cannot attend to previous deleted tokens (D_i). (Bottom right) Soft prompt (whole): After the compression token (C_i) attends to all prior input tokens (I_i), subsequent output tokens (O_i) cannot attend to tokens before the compression token.

*Co-first authors.

[1] https://github.com/ZongqianLi/Prompt-Compression-Survey

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
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

for prompt compression as shown in Figure 1, have gained increasing attention. However, the optimal architectures and underlying mechanisms of prompt compression remain unclear, highlighting an important area for further investigation.

This survey aims to introduce LLM prompt compression to people with prior knowledge of attention mechanisms, transformers, and LLMs. Unlike previous surveys on general efficient prompting (Chang et al., 2024), this paper specifically focuses on prompt compression. The preliminary knowledge is introduced in Section 2, covering prompting and efficiency. This is followed by a detailed discussion of hard prompt methods in Section 3 and soft prompt methods in Section 4. The key elements driving the evolution of prompt compression models are examined, and insights into various methods are provided. Their downstream adaptations are discussed in Section 5 as well. Finally, Section 6 analyzes the challenges of current prompt compression techniques and proposes several potential solutions. An overview of prompt compression methods and their downstream adaptations is shown in Figure 2.

This survey highlights that:

- In addition to the exploration of existing prompt compression methods, we provide our interpretation of their mechanisms: filtering or paraphrasing in hard prompt methods, and attention modification, PEFT, modality integration, or a new synthetic language in soft prompt methods.
- We identify challenges in current methods, including fine-tuning problems such as overfitting and performance reduction, long compression time, and the need for comparison with attention optimization techniques.
- We propose several future directions to solve current challenges, including optimizing the compression encoder, combining hard and soft prompts, and integrating insights from multimodality.

2 Preliminary

Prompts are important for LLMs, broadly classified into hard and soft prompts (Liu et al., 2023b). They serve as input instructions to guide LLMs in performing tasks such as summarization, classification, translation, and question answering without the need for fine-tuning (Vatsal and Dubey, 2024). The flexibility and effectiveness of prompts leverage the generalization abilities of LLMs, influenced by elements such as wording, in-context examples, clarity, and accuracy, making prompt design an important area (Schulhoff et al., 2024).

Prompt structures in this survey consist of two main components: an instruction or context and an input or question, depending on the task. The instruction, input, output format is commonly used in instruction fine-tuning datasets such as Alpaca (Taori et al., 2023) and PwC (Ge et al., 2024) for tasks such as content creation and language translation. The context, question, answer format is prevalent in QA and reading comprehension datasets such as SQuAD (Rajpurkar et al., 2016), HotpotQA (Yang et al., 2018), and RACE (Lai et al., 2017). In few-shot scenarios, several demonstrations with these formats are prepended at the beginning of the prompt, which serves as additional instruction or context to guide the behavior of the LLM. Prompts can vary significantly in length, with different prompt compression methods targeting some or all of these components.

Hard prompts are natural language prompts made up of tokens from the vocabulary set of the LLM, corresponding to specific words or subwords (Sennrich et al., 2016). These prompts can be generated by either humans or models. While natural language prompts are interpretable and provide transparency, their inherent ambiguity can make it difficult to fully capture intent in a concise manner. This limitation reduces the utility of hard prompts in diverse contexts and scenarios. Additionally, creating effective and precise hard prompts requires considerable human effort, and can involve training a model to refine or optimize these prompts. Furthermore, variations in hard prompts can lead to differences in LLM performance for the same task.

Soft prompts are trainable, continuous vectors that share the same dimensions as token embeddings in the dictionary of the LLM (Zhao et al., 2023). Different from hard prompts, the vectors in soft prompts are trained to convey nuanced meanings that cannot be captured by discrete tokens in the predefined vocabulary. When tuned on diverse datasets, soft prompts are expected to help the LLM perform well across different tasks. However, as the dataset size grows, the computational resources needed increase as well. Additionally, soft prompts are less explainable by humans compared to hard prompts, as their tokens are not directly readable.

Prompt compression aims to reduce the length of prompts, thereby improving the efficiency of processing LLM inputs (Wan et al., 2024). There are

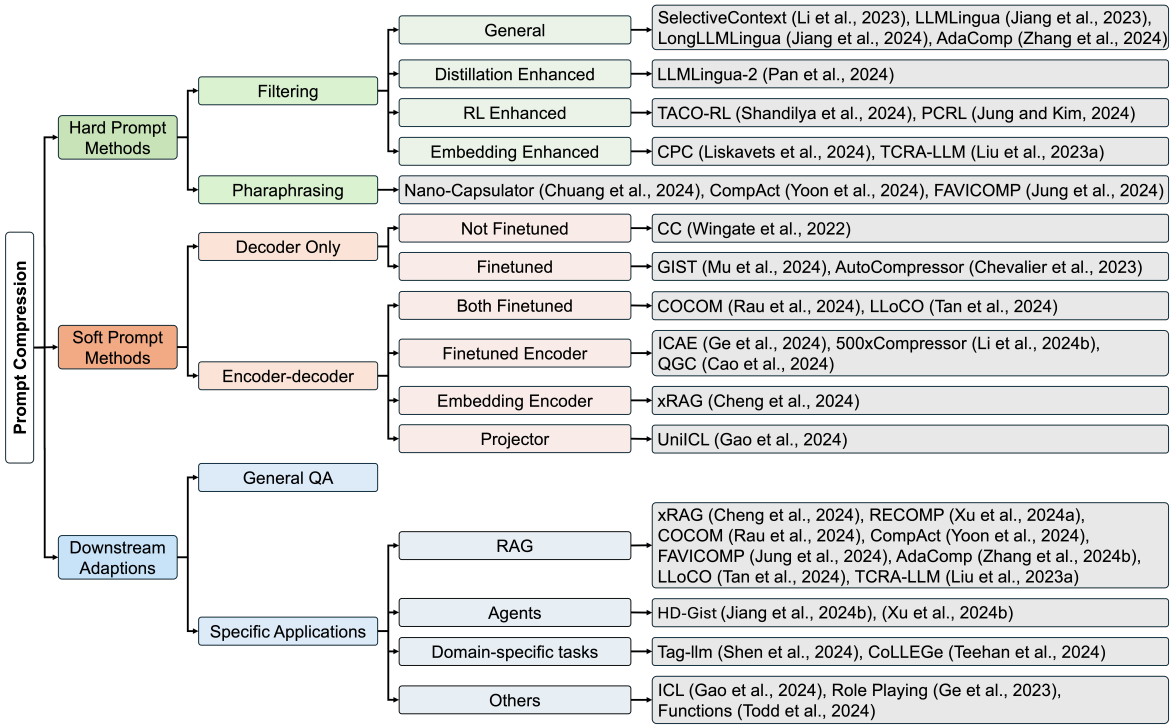

Figure 2: Tree overview of prompt compression methods and their downstream adaptions. For downstream adaptations, compression methods not belonging to specific categories can be classified into general QA.

two primary approaches to prompt compression: removing unnecessary or low-information content (hard prompt methods) and learning continuous prompt vectors of the prompt information in the embedding space (soft prompt methods) (Chang et al., 2024). Hard prompt methods act as a form of filtering, still using natural language tokens, although the resulting prompts may be less fluent and grammatically correct, and can potentially generalize to LLMs with different embeddings. Soft prompt methods, on the other hand, employ encoding to convert prompt information into continuous prompt vectors, resulting in special embeddings that cannot be understood by humans. Different from KV compression, prompt compression only manipulates on the input tokens, but not changes the KV values for the input tokens during encoding. Thus, the compression ratio is calculated by the number of original input tokens divided by the number of compression tokens. The evaluation of prompt compression methods typically follows two approaches: comparing their performance under the same compression ratio, or examining their achievable compression ratios while maintaining comparable performance levels.

3 Hard Prompt Methods

Hard prompt methods remove unnecessary tokens from the original prompt while maintaining the use of natural language words or sub-words (Reynolds and McDonell, 2021). These methods are particularly useful for LLMs that only accept natural language inputs, such as close API models, rather than word embeddings. There are three representative hard prompt methods: SelectiveContext (Li et al., 2023) and LLMLingua (Jiang et al., 2023) for filtering, and Nano-Capsulator (Chuang et al., 2024) for paraphrasing, as shown in Figure 3.

SelectiveContext identifies and deletes redundant or less informative parts of an input prompt by quantifying the informativeness of lexical units using self-information (Li et al., 2023). To maintain text coherence, the syntactic parsing capabilities of SpaCy[2] are used to group individual tokens into noun phrases based on dependency parsing. SelectiveContext does not rely on external models or additional parameters, and can be applied to any model architecture. However, there are two main disadvantages: (1) It relies on accurate phrase boundary detection using SpaCy. (2) Currently, there are no methods for merging verb phrases.

[2] https://spacy.io/api/pipeline-functions/#merge_noun_chunks

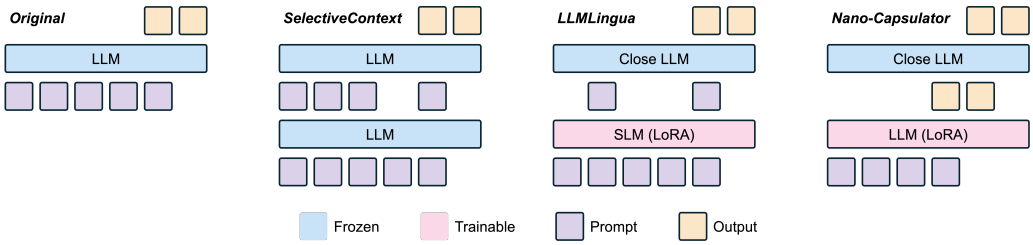

Figure 3: Architectures for various prompt compression models by hard prompt methods. For SelectiveContext and LLMLingua, the bottom language models filter the prompt tokens without modifying them, serving as selection mechanisms. In Nano-Capsulator, the bottom LLM generates a paraphrased version of the input prompt which then serves as input for the LLM above. "SLM" means "small language model". "Close LLM" refers to closed-source language models that only accept natural language inputs through API calls.

LLMLingua employs a smaller language model, such as GPT-2 (Radford et al., 2019), to calculate the self-information of content, similar to SelectiveContext, and removes redundant tokens from the natural language prompt before it is fed to the LLM (Jiang et al., 2023). LLMLingua operates on prompts structured as {Instruction, Input (Demonstrations), Question}, initially selecting key demonstrations based on self-information. It then applies token-level filtering across the prompt, allowing for breaking words into sub-word units and avoiding noun phrase merging by SpaCy. For key elements such as numbers and units, LLMLingua incorporates token preservation algorithms that prioritize these elements within instructions and questions. Achieving compression ratios up to 20x, the compression process of LLMLingua is managed by the smaller external language model, allowing it to work with close LLMs. However, there are two limitations: (1) The smaller language model requires additional memory and may use a different tokenizer than the larger LLM. (2) Since not all prompts contain a large portion of in-context examples, it is important to differentiate between prompt compression and in-context example selection.

Nano-Capsulator summarizes the original prompt into a concise natural language version, which is then input into the LLM (Chuang et al., 2024). This process removes irrelevant information and makes the prompt into fluent sentences. The compression model, a fine-tuned Vicuna-7B, operates independently of the LLM. Different from standard summarization, Nano-Capsulator includes a semantic preservation loss to retain key meanings important for downstream tasks and a reward function to optimize the utility of the prompt for the LLM. This targeted approach yields better task performance by enhancing semantic relevance. However, the memory cost of the compression model is not negligible. Additionally, the compression process is akin to a pre-generation step, requiring more computational resources due to the need for additional inference rather than simple encoding.

In addition to the methods mentioned above, general hard prompt methods include LongLLMLingua, which has a longer compression window than LLMLingua by applying document reordering and subsequence recovery (Jiang et al., 2024a), and AdaComp, an adaptive compression method that dynamically selects relevant documents based on query difficulty and retrieval quality (Zhang et al., 2024b). These general methods have been further improved in different ways. For example, LLMLingua-2 uses data distillation to create a compressed dataset and trains a classifier to retain essential tokens (Pan et al., 2024). Both PCRL (Jung and Kim, 2024) and TACO-RL (Shandilya et al., 2024) apply RL for token selection: PCRL is model-agnostic, while TACO-RL is task-specific. CPC (Liskavets et al., 2024) and TCRA-LLM (Liu et al., 2023a) reduce tokens by leveraging embeddings: CPC ranks sentence relevance with context-aware embeddings, while TCRA-LLM uses embeddings for summarization and semantic compression.

4 Soft Prompt Methods

4.1 Architectures

Soft prompt compression methods typically consist of two main components: an encoder that compresses the prompts into a shorter sequence of continuous special tokens and a decoder that processes the compressed prompts to generate corresponding responses. As these models improve, they show better generalization abilities, require fewer additional trainable parameters in the original LLMs, allow for longer prompt lengths to be

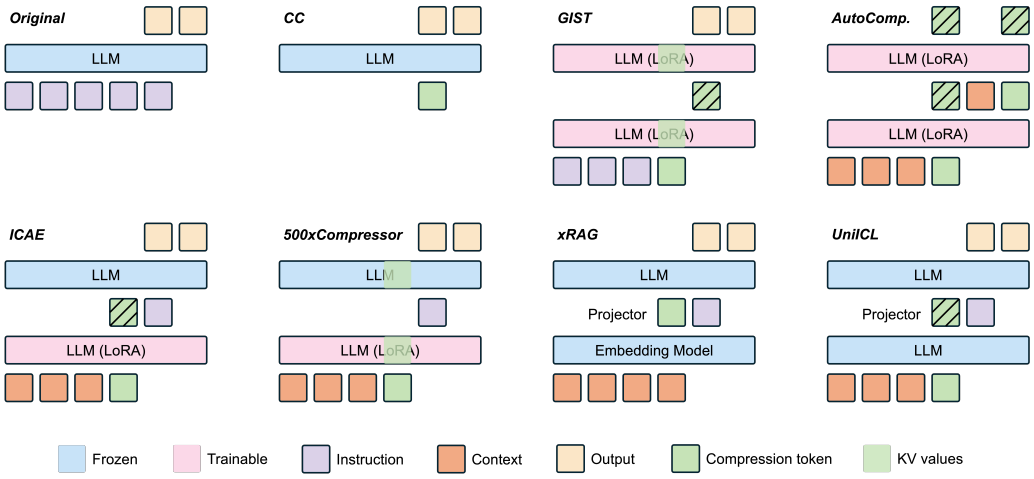

Figure 4: Architectures for various prompt compression models by soft prompt methods. Tokens with diagonal stripes represent the output tokens processed by the language models. Different from hard prompt methods, the bottom LLMs in soft prompt methods process the input tokens, and their outputs (tokens with diagonal stripes) serve as input for the LLMs above.

compressed, and achieve higher compression ratios. Several common architectures for these compression models are illustrated in Figure 4, including contrastive conditioning (CC) (Wingate et al., 2022), gist tokens (GIST) (Mu et al., 2024), AutoCompressor (Chevalier et al., 2023), in-context autoencoder (ICAE) (Ge et al., 2024), 500xCompressor (Li et al., 2024), xRAG (Cheng et al., 2024), and UniICL (Gao et al., 2024)

CC is a decoder-only method that trains a shorter soft prompt to approximate the output distribution of a natural language prompt by minimizing the Kullback-Leibler (KL) divergence across token sequences, thereby aligning with desired responses (Wingate et al., 2022). The output distributions are estimated through repeated sequence sampling, with both the natural and soft prompts serving as conditional inputs for token generation. By applying various contrastive contexts, soft prompts can be trained to produce specific attributes, such as enhanced sentiment, allowing for effective content control. However, each soft prompt is uniquely trained for a specific natural language prompt, which limits the generalization capabilities of CC, as new prompts require retraining from scratch.

GIST modifies the attention mechanism of the LLM (Mu et al., 2024). Compression tokens, a series of a new extended vocabulary token initialized by specific trained values, are appended after the original prompt tokens. While these tokens can attend to the original prompt, newly generated tokens can only attend to the compression tokens, enforcing a separation in attention. This setup can be viewed as an encoder-decoder architecture: an LLM fine-tuned on Alpaca+ functions as the encoder, compressing the original prompt into a smaller set of KV values. These KV values then serve as input to the decoder (the same fine-tuned LLM), which generates corresponding responses. Different from CC, GIST can compress unseen prompts without additional fine-tuning, achieving a compression ratio of up to 26x, though it is limited by the maximum compressible prompt length used in its fine-tuning dataset. Additionally, the compression tokens cannot be used with the original untuned LLM, restricting its broader applicability.

AutoCompressor shares a similar architecture with GIST and can handle long-context prompt compression up to 30,720 tokens (Chevalier et al., 2023). The whole process is recursive, with the original prompt divided into several sub-prompts. In each iteration, the sub-prompt is compressed into a small set of continuous prompt vectors, which are then passed to the next iteration along with a new sub-prompt for further compression. This continues until all the encoded embeddings are collected, representing the information from the entire original prompt. While AutoCompressor increases the maximum prompt length that can be compressed, the training process is time-consuming, and the compression tokens cannot be used by the original untuned LLM.

ICAE increases the compression length and uses the frozen LLM as the decoder (Ge et al., 2024). It compresses long, information-rich contexts into a small number of continuous prompt vectors, which are then used for QA. Unlike GIST, which compresses low-information texts of around 30 tokens

focused on questions and instructions, ICAE can handle detailed contexts. The question itself remains uncompressed, and the answer is generated based on the compressed context and the uncompressed question. Since the decoder is frozen, the compression tokens can be used directly with the original LLM without fine-tuning. ICAE compresses up to 512 tokens into 32, 64, or 128 continuous vectors, achieving compression ratios between 4x and 16x. By concatenating multiple groups of encoded embeddings, ICAE can handle up to 5,120 tokens. However, its compression ratio decreases compared to GIST. In addition, ICAE is trained and tested on the Pile dataset (Gao et al., 2020b), which may overlap with the training corpus of the LLM, raising concerns about potential data leakage and the possibility of retrieving answers from the memory of the LLM.

500xCompressor explores prompt compression at high compression ratios while maintaining the compression length limit of ICAE (Li et al., 2024). Similar to ICAE, 500xCompressor employs trainable LoRA parameters (Gao et al., 2020a) in the encoder, while keeping the original LLM frozen in the decoder. However, different from ICAE, 500xCompressor feeds the decoder with the KV pairs of the compression tokens rather than the continous vectors themselves. These KV pairs keep more detailed information than embedding vectors, especially under high compression ratios. 500xCompressor uses a small number of tokens (1-16) to compress longer sequences (up to 480 tokens), achieving compression ratios from 6x to 480x, while retaining more than 60-70% of the capabilities of the uncompressed prompts. Moreover, the test set ArxivQA, generated from arXiv abstracts from January to April 2024, ensures evaluation on strictly unseen data, mitigating potential data leakage. Though 500xCompressor collects the KV pairs for the compression tokens, it does not modify them, thus 500xCompressor is a prompt compression method not a KV compression method.

xRAG uses a frozen embedding model as the encoder, with only an adapter, positioned between the encoder and the decoder LLM, containing trainable parameters (Cheng et al., 2024). Although primarily designed for Retrieval-Augmented Generation (RAG) (Lewis et al., 2020), the task of xRAG remains a QA task, making it applicable as a general prompt compression method. xRAG proves that current embedding models can compress information into a single token for QA tasks. While several embedding models were tested in the original paper, the final choice was SFR-Embedding-Mistral (Meng et al., 2024), which is still based on the LLM and requires substantial memory. As a result, xRAG needs to load two LLMs and a projector, whereas ICAE and 500xCompressor only need to load a single LLM and a set of LoRA parameters.

UniICL focuses on compressing the demonstrations component of the input prompt, leaving the query unchanged (Gao et al., 2024). The only trainable component in UniICL is a projector placed between the encoder and decoder. Unlike previous soft prompt methods, both the encoder and decoder in UniICL are frozen and utilize the same LLM, reducing gradient computation during training and conserving memory for loading the LLM. In ICL, the quality and relevance of the examples influence model performance. The encoded continuous vectors in UniICL can be considered as embeddings for various in-context examples, eliminating the need for additional embedding processes during in-context example selection.

Besides the seven methods introduced above, several other similar approaches exist, such as COCOM (Rau et al., 2024), LLoCO (Tan et al., 2024), and QGC (Cao et al., 2024). COCOM and LLoCO use fine-tuned encoder-decoder setups, while QGC employs a frozen decoder. Both COCOM and LLoCO are designed for the task of RAG: COCOM compresses multiple documents into groups of context embeddings and inputs them together into the decoder, while LLoCO stores and retrieves specific LoRA parameters for the decoder to adapt to particular text types and tasks. Different architectures possess unique characteristics and advantages (Jha et al., 2024), warranting a comprehensive comparative analysis.

4.2 Insights

Soft prompt methods for prompt compression can be understood from several perspectives, including attention mechanism optimization, analogies with prompt and prefix tuning, encoding natural language into a new modality, and viewing compressed prompts as a new language for LLMs.

Attention mechanism. In the standard self-attention of transformer, each newly generated token attends to all previous tokens, which increases computational costs along with the sequence length. Soft prompt compression reduces the input length in two stages: first, a small number of special to-

kens attend to the full input, storing key information in these tokens; second, new tokens are generated based solely on the encoded continuous vectors. This process blocks the need for the full input during generation, limiting attention to the encoded continuous vectors and reducing computation. This can be seen as a form of attention optimization, though it differs in that the KV pairs generated in the first stage are not identical to those in the original LLM (as LoRA parameters are added). There are other attention optimization methods as well, including sliding window attention (Beltagy et al., 2020) and sparse attention (Child et al., 2019). In sliding window attention, for example, all input tokens are kept, however, each token can only attend to a limited number of preceding tokens.

Prompt and prefix tuning. Prompt tuning (Lester et al., 2021) and prefix tuning (Li and Liang, 2021) are PEFT methods for language models. In prompt tuning, a set of trainable embeddings is added to the beginning of the input. In prefix tuning, both the added embeddings and their corresponding KV pairs in each layer of the language model are trainable as well. While these methods are useful for fine-tuning models for specific tasks, their significance has decreased with the appearance of LLMs capable of handling downstream tasks through simple prompting. ICAE and related methods are similar to prompt tuning. Instead of manually training the embeddings, these embeddings are generated by the encoder (a fine-tuned LLM) through compressing the natural language input. Each natural language prompt is encoded into a unique set of embeddings, with different prompts corresponding to different sets. This process provides a zero-shot way of determining the parameters typically trained in prompt tuning. In contrast, 500xCompressor is more akin to prefix tuning. Here, the inputs for the decoder are not embeddings, but KV pairs, which are determined by the fine-tuned LLM encoder. These KV pairs contain more parameters than embeddings and are assumed to store richer details (Li and Liang, 2021). 500xCompressor proves that KV pairs can compress more detailed information at high compression ratios and outperform embeddings in terms of data retention (Li et al., 2024). Since both embeddings and KV pairs are generated simultaneously by the encoder, they offer similar improvements in LLM efficiency, including higher inference speed and reduced computational costs.

Modality integration. Compressed texts can be considered a new modality, similar to vision and speech. In vision-language models, for example, an image encoder converts an image into a list of embeddings, which are then used by an LLM decoder for downstream tasks such as image captioning or QA (Zhang et al., 2024a). This architecture parallels that of prompt compression models, where an encoder compresses natural language into special embeddings and an LLM decoder utilizes these continuous vectors. The training processes for both are similar as well: image encoders are paired with decoders trained to predict missing parts of an image and align with textual descriptions, while prompt compression encoders are coupled with LLM decoders trained to regenerate original texts from compressed prompt vectors. Thus, encoded continuous vectors from text can be seen as a new, rich-information modality derived from natural language. However, it is important to note that texts contain more specific details and have a higher information density than images. As a result, text compression requires greater precision than image encoding, and the potential for information loss during prompt compression is more significant than during image encoding.

New synthetic language. A new language can be defined by three key characteristics: (1) it can encode information and convert thoughts, concepts, or data into embeddings; (2) this encoded information can be transmitted between entities; and (3) the entity can dynamically adjust its interpretation or output based on the input it receives, a process known as adaptive evaluation (Mansilla, 2004). In the context of prompt compression, the encoded embeddings represent natural language texts as continuous vectors, which can be understood by LLMs. These continuous vectors can be saved, transferred between different LLMs, and facilitate knowledge transfer. Moreover, when the encoded information changes, LLMs can dynamically adjust their responses based on the new input. Therefore, the encoded embeddings generated by prompt compression models can be considered a new, more efficient language for LLMs. Work is ongoing in the definition and evaluation of this emerging LLM-specific language (Guo et al., 2020).

5 Downstream Adaptions

Prompt compression has a wide range of applications in domains such as general QA, RAG (Cheng et al., 2024; Xu et al., 2024a; Rau et al., 2024; Yoon

et al., 2024; Jung et al., 2024; Zhang et al., 2024b; Tan et al., 2024; Liu et al., 2023a), ICL (Gao et al., 2024), role playing (Ge et al., 2023), agent-based (Jiang et al., 2024c; Xu et al., 2024b), and interdisciplinary tasks (Shen et al., 2024; Teehan et al., 2024). In general QA, prompt compression methods are generally used to compress instructions (Mu et al., 2024) or contexts (Li et al., 2024) to perform various instruction-following tasks (Ge et al., 2024). In RAG, xRAG exemplifies a method that answers questions based on text embeddings encoded from retrieved documents rather than processing the entire text (Cheng et al., 2024). For ICL, UniICL is an example method that compresses in-context examples to a smaller number of embedding tokens, helping with the selection of relevant examples (Gao et al., 2024). In agent, API documentation can be compressed to enable more efficient tool use (Jiang et al., 2024c).

6 Challenges and Future Work

6.1 Current Challenges

Current prompt compression methods face several challenges, including information loss, reduced model capability, and marginal improvements in efficiency, due to fine-tuning limitations and inefficient compression processes.

Finetuning problems. Although some soft prompt methods, such as ICAE, 500xCompressor, xRAG, and UniICL, do not need to fine-tune the decoder LLM, the input soft prompt functions similarly to prompt tuning or prefix tuning for the decoder, leading to problems related to fine-tuning. Previous work has shown that fine-tuning base models can result in forgetting, overfitting, and model drift, which can decrease the generalization performance of the base LLM (Gururangan et al., 2020). To avoid these problems, prompt compression models must be trained on large, diverse datasets, which is computationally expensive. Furthermore, the trained encoders are specific to the corresponding decoder LLMs, meaning that when the LLM is updated, for example, from LLaMA-2 (Touvron et al., 2023) to LLaMA-3 (Dubey et al., 2024), the encoder must be re-trained. Hard prompts face challenges as well: filtered hard prompts may decrease grammatical correctness and provide an unfamiliar input distribution to the LLM, potentially influencing its performance.

Limited efficiency improvements. Although prompt length can be reduced by dozens or even hundreds of times, the time required for compression and the memory needed to store the compressor remain under-optimized. Current encoders in soft prompt methods can be finetuned using either full-parameter training or PEFT methods, such as LoRA. Fully fine-tuned LLMs are the most resource-expensive, while the additional parameters in LoRA-based encoders are modular, allowing for separate loading. However, larger encoders result in longer compression times. Hardware variability can lead to fluctuations in time for the same task as well (Hennessy and Patterson, 2011), introducing large standard deviations and relative errors, making it essential to perform multiple evaluations for accuracy. Moreover, if the encoder and decoder are of equal size in soft prompt methods, the computations required for prompt compression are nearly the same or even slightly higher than those for encoding the original prompt. As a result, efficiency gains are only realized during the generation of new tokens. For tasks with short outputs, these improvements may not be substantial. Hard prompt methods also face issues: additional models may be needed to determine which tokens to delete, and the filtered prompts still need to be re-encoded by the LLM, further influencing efficiency.

Comparison with attention optimization methods. As discussed in Section 4.2, soft prompt methods can be regarded as special attention modifications. However, current prompt compression methods have not been compared to traditional attention optimization methods, such as sliding window attention (Beltagy et al., 2020) and sparse attention (Child et al., 2019). Unlike soft prompt methods, attention modifications do not need an encoder model, which eliminates additional memory costs (Ren et al., 2021). Additionally, in soft prompt methods, the input and generated tokens rely on different attention mechanisms, whereas attention optimization methods apply the same mechanism to both the input and the generated tokens, resulting in greater stability and scalability (Tay et al., 2022). Therefore, it is important to determine the compression ratio when both methods have equivalent computational efficiency and compare prompt compression methods with attention modification methods.

6.2 Future Directions

To address current challenges and improve prompt compression methods, several future directions are proposed to reduce information loss while increas-

ing compression ratio and speed.

Encoder Optimization. In current soft prompt methods, encoders are similar in size to decoders. For instance, if a decoder has 8 billion parameters, the encoder may add tens of millions of trainable parameters on top of that. As a result, the time used for compression is comparable to the time it takes for the original LLM to process inputs, meaning that soft prompt methods only improve efficiency during the generation of new tokens. In theory, large encoders that are similar to the decoders can compress information well from the original text into continuous vectors for the LLM. However, smaller, well-trained models such as BERT, which have fewer parameters (as least 10 times smaller than LLMs), are capable of encoding semantic information effectively (Devlin et al., 2019). This will substantially increase compression speed. Besides LoRA, other PEFT methods, such as QLoRA (Dettmers et al., 2023), DoRA (yang Liu et al., 2024), and MoRA (Jiang et al., 2024b) are worth trying for fine-tuning the compression encoder as well.

Combination of hard and soft prompts. Hard prompt methods increase information density by filtering out unnecessary tokens. Soft prompt methods represent the original text using a small number of special tokens. Since hard and soft prompt methods operate through orthogonal mechanisms, their combination can further enhance compression ratios. However, when combining these methods, their compression times add up sequentially, leading to increased overall processing time for input.

Insights from multimodality. As discussed in Section 4.2, soft prompt methods can be understood as a form of modality integration between natural language and compressed language. This opens the possibility of applying insights from multimodality to benefit prompt compression models. There are mainly two ways to encode images into embeddings: self-attention and cross-attention (Jin et al., 2024). In self-attention, images and query vectors are input to the image encoder together, whereas in cross-attention, only query vectors are input to the encoder, and they attend to external image embeddings in each layer. Current soft prompt methods rely on self-attention to transfer information from natural language prompts to compression tokens, however, cross-attention remains unexplored. Trying other multimodal architectures, such as those using cross-attention, may offer new ways to leverage compression tokens. In image-text integration, image encoders are trained to align image embeddings with natural language embeddings, a process that can be adapted for prompt compression models (Radford et al., 2021).

7 Conclusions

This survey provides a comprehensive overview of the previous prompt compression methods, from the perspectives of hard and soft prompt approaches. In addition to discussing the technical approaches of these models, different perspectives on understanding the compression process and their applications are explored. Furthermore, we also discuss the challenges of the existing prompt compression methods and suggest the potential future development directions. We hope our survey offers a comprehensive overview of existing methods, providing deeper insights into their motivations. We also aim for our suggested future directions to inspire the community and support future developments in the field.

Limitations

This paper focuses specifically on prompt compression and does not provide an overview of all efficient prompting methods or other efficiency-related LLM techniques. Rather than addressing a broad topic, it offers a detailed explanation and insights into the specific area of prompt compression. It should be noted that the use of any prompt compression methods should follow their guidelines and copyright restrictions.

Ethics Statement

No ethical approval was required for this study. No ethical concerns are present.

Availability Statement

The list of papers and updates related to this survey are uploaded to the open-source community at https://github.com/ZongqianLi/Prompt-Compression-Survey.

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

Zhiwei Cao, Qian Cao, Yu Lu, Ningxin Peng, Luyang Huang, Shanbo Cheng, and Jinsong Su. 2024. Retaining key information under high compression ratios:

Query-guided compressor for LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12685–12695, Bangkok, Thailand. Association for Computational Linguistics.

Kaiyan Chang, Songcheng Xu, Chenglong Wang, Yingfeng Luo, Tong Xiao, and Jingbo Zhu. 2024. Efficient prompting methods for large language models: A survey. *Preprint*, arXiv:2404.01077.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. 2024. xrag: Extreme context compression for retrieval-augmented generation with one token. *Preprint*, arXiv:2405.13792.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

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

Yu-Neng Chuang, Tianwei Xing, Chia-Yuan Chang, Zirui Liu, Xun Chen, and Xia Hu. 2024. Learning to compress prompt in natural language formats. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7756–7767, Mexico City, Mexico. Association for Computational Linguistics.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, volume 35, pages 30318–30332. Curran Associates, Inc.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems*, volume 36, pages 10088–10115. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, ..., and Zhiwei Zhao. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Jun Gao, Ziqiang Cao, and Wenjie Li. 2024. Unifying demonstration selection and compression for in-context learning. *Preprint*, arXiv:2405.17062.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020a. The pile: An 800gb dataset of diverse text for language modeling. *Preprint*, arXiv:2101.00027.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020b. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Tao Ge, Hu Jing, Li Dong, Shaoguang Mao, Yan Xia, Xun Wang, Si-Qing Chen, and Furu Wei. 2023. Extensible prompts for language models on zero-shot language style customization. In *Advances in Neural Information Processing Systems*, volume 36, pages 35576–35591. Curran Associates, Inc.

Tao Ge, Hu Jing, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2024. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations*.

Shangmin Guo, Yi Ren, Agnieszka Słowik, and Kory Mathewson. 2020. Inductive bias and language expressivity in emergent communication. *4th NeurIPS Workshop on Emergent Communication*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

John L Hennessy and David A Patterson. 2011. *Computer architecture: a quantitative approach*. Elsevier.

Siddharth Jha, Lutfi Eren Erdogan, Sehoon Kim, Kurt Keutzer, and Amir Gholami. 2024. Characterizing prompt compression methods for long context inference. In *Workshop on Efficient Systems for Foundation Models II @ ICML2024*.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376, Singapore. Association for Computational Linguistics.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024a. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1658–1677, Bangkok, Thailand. Association for Computational Linguistics.

Ting Jiang, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, et al. 2024b. Mora: High-rank updating for parameter-efficient fine-tuning. *arXiv preprint arXiv:2405.12130*.

Yichen Jiang, Marco Vecchio, Mohit Bansal, and Anders Johannsen. 2024c. Hierarchical and dynamic prompt compression for efficient zero-shot API usage. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 2162–2174, St. Julian's, Malta. Association for Computational Linguistics.

Yizhang Jin, Jian Li, Yexin Liu, Tianjun Gu, Kai Wu, Zhengkai Jiang, Muyang He, Bo Zhao, Xin Tan, Zhenye Gan, Yabiao Wang, Chengjie Wang, and Lizhuang Ma. 2024. Efficient multimodal large language models: A survey. *Preprint*, arXiv:2405.10739.

Dongwon Jung, Qin Liu, Tenghao Huang, Ben Zhou, and Muhao Chen. 2024. Familiarity-aware evidence compression for retrieval augmented generation. *Preprint*, arXiv:2409.12468.

Hoyoun Jung and Kyung-Joong Kim. 2024. Discrete prompt compression with reinforcement learning. *IEEE Access*, 12:72578–72587.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. 2023. Compressing context to enhance inference efficiency of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6342–6353, Singapore. Association for Computational Linguistics.

Zongqian Li, Yixuan Su, and Nigel Collier. 2024. 500xcompressor: Generalized prompt compression for large language models. *Preprint*, arXiv:2408.03094.

Barys Liskavets, Maxim Ushakov, Shuvendu Roy, Mark Klibanov, Ali Etemad, and Shane Luke. 2024. Prompt compression with context-aware sentence encoding for fast and improved llm inference. *Preprint*, arXiv:2409.01227.

Junyi Liu, Liangzhi Li, Tong Xiang, Bowen Wang, and Yiming Qian. 2023a. TCRA-LLM: Token compression retrieval augmented large language model for inference cost reduction. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9796–9810, Singapore. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023b. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9).

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 21702–21720. Curran Associates, Inc.

Paloma Ubeda Mansilla. 2004. Foundations of language (brain, meaning, grammar, evolution), by ray jackendoff. *Ibérica: Revista de la Asociación Europea de Lenguas para Fines Específicos (AELFE)*, (7):150–152.

Rui Meng, Ye Liu, Shafiq Rayhan Joty, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. 2024. Sfr-embedding-mistral:enhance text retrieval with transfer learning. Salesforce AI Research Blog.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2024. Learning to compress prompts with gist tokens. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 963–981, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark,

Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

David Rau, Shuai Wang, Hervé Déjean, and Stéphane Clinchant. 2024. Context embeddings for efficient answer generation in rag. *Preprint*, arXiv:2407.09252.

Hongyu Ren, Hanjun Dai, Zihang Dai, Mengjiao Yang, Jure Leskovec, Dale Schuurmans, and Bo Dai. 2021. Combiner: Full attention transformer with sparse computation cost. In *Advances in Neural Information Processing Systems*, volume 34, pages 22470–22482. Curran Associates, Inc.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI EA '21, New York, NY, USA. Association for Computing Machinery.

Sander Schulhoff, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, Chenglei Si, Yinheng Li, Aayush Gupta, HyoJung Han, Sevien Schulhoff, Pranav Sandeep Dulepet, Saurav Vidyadhara, Dayeon Ki, Sweta Agrawal, Chau Pham, Gerson Kroiz, Feileen Li, Hudson Tao, Ashay Srivastava, Hevander Da Costa, Saloni Gupta, Megan L. Rogers, Inna Goncearenco, Giuseppe Sarli, Igor Galynker, Denis Peskoff, Marine Carpuat, Jules White, Shyamal Anadkat, Alexander Hoyle, and Philip Resnik. 2024. The prompt report: A systematic survey of prompting techniques. *Preprint*, arXiv:2406.06608.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Shivam Shandilya, Menglin Xia, Supriyo Ghosh, Huiqiang Jiang, Jue Zhang, Qianhui Wu, and Victor Rühle. 2024. Taco-rl: Task aware prompt compression optimization with reinforcement learning. *Preprint*, arXiv:2409.13035.

Junhong Shen, Neil Tenenholtz, James Brian Hall, David Alvarez-Melis, and Nicolo Fusi. 2024. Tag-llm: Repurposing general-purpose llms for specialized domains. *arXiv preprint arXiv:2402.05140*.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Sijun Tan, Xiuyu Li, Shishir Patil, Ziyang Wu, Tianjun Zhang, Kurt Keutzer, Joseph E. Gonzalez, and Raluca Ada Popa. 2024. Lloco: Learning long contexts offline. *Preprint*, arXiv:2404.07979.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *ACM Comput. Surv.*, 55(6).

Ryan Teehan, Brenden Lake, and Mengye Ren. 2024. CoLLEGe: Concept embedding generation for large language models. In *First Conference on Language Modeling*.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. 2024. Function vectors in large language models. In *The Twelfth International Conference on Learning Representations*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Shubham Vatsal and Harsh Dubey. 2024. A survey of prompt engineering methods in large language models for different nlp tasks. *Preprint*, arXiv:2407.12994.

Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, and Mi Zhang. 2024. Efficient large language models: A survey. *Transactions on Machine Learning Research*. Survey Certification.

David Wingate, Mohammad Shoeybi, and Taylor Sorensen. 2022. Prompt compression and contrastive conditioning for controllability and toxicity reduction in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5621–5634, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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

Yang Xu, Yunlong Feng, Honglin Mu, Yutai Hou, Yitong Li, Xinghao Wang, Wanjun Zhong, Zhongyang Li, Dandan Tu, Qingfu Zhu, Min Zhang, and Wanxiang Che. 2024b. Concise and precise context compression for tool-using language models. *Preprint*, arXiv:2407.02043.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Shih yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. DoRA: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*.

Chanwoong Yoon, Taewhoo Lee, Hyeon Hwang, Minbyul Jeong, and Jaewoo Kang. 2024. Compact: Compressing retrieved documents actively for question answering. *Preprint*, arXiv:2407.09014.

Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. 2024a. Vision-language models for vision tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5625–5644.

Qianchi Zhang, Hainan Zhang, Liang Pang, Hongwei Zheng, and Zhiming Zheng. 2024b. Adacomp: Extractive context compression with adaptive predictor for retrieval-augmented large language models. *Preprint*, arXiv:2409.01579.

Wenbo Zhao, Arpit Gupta, Tagyoung Chung, and Jing Huang. 2023. SPC: Soft prompt construction for cross domain generalization. In *Proceedings of the 8th Workshop on Representation Learning for NLP (RepL4NLP 2023)*, pages 118–130, Toronto, Canada. Association for Computational Linguistics.

A Appendix

To provide readers with a way to locate reference papers on prompt compression methods, Figure 2 has been translated into Table 1. Data examples for prompt structures in Section 2 are presented in Table 2.

This paper was refined with ChatGPT.

Hard Prompt Methods
Filtering:
General: SelectiveContext (Li et al., 2023), LLMLingua (Jiang et al., 2023), LongLLMLingua (Jiang et al., 2024a), AdaComp (Zhang et al., 2024b)
Distillation Enhanced: LLMLingua-2 (Pan et al., 2024)
RL Enhanced: TACO-RL (Shandilya et al., 2024), PCRL (Jung and Kim, 2024)
Embedding Enhanced: CPC (Liskavets et al., 2024), TCRA-LLM (Liu et al., 2023a)
Paraphrasing:
Nano-Capsulator (Chuang et al., 2024), CompAct (Yoon et al., 2024), FAVICOMP (Jung et al., 2024)
Soft Prompt Methods
Decoder Only:
Not Finetuned: CC (Wingate et al., 2022)
Finetuned: GIST (Mu et al., 2024), AutoCompressor (Chevalier et al., 2023)
Encoder-decoder:
Both Finetuned: COCOM (Rau et al., 2024), LLoCO (Tan et al., 2024)
Finetuned Encoder: ICAE (Ge et al., 2024), 500xCompressor (Li et al., 2024), QGC (Cao et al., 2024)
Embedding Encoder: xRAG (Cheng et al., 2024)
Projector: UniICL (Gao et al., 2024)
Applications:
RAG:
RAG: xRAG (Cheng et al., 2024), RECOMP (Xu et al., 2024a), COCOM (Rau et al., 2024), CompAct (Yoon et al., 2024), FAVICOMP (Jung et al., 2024), AdaComp (Zhang et al., 2024b), LLoCO (Tan et al., 2024), TCRA-LLM (Liu et al., 2023a)
Agents: HD-Gist (Jiang et al., 2024c), (Xu et al., 2024b)
Domain-specific tasks: Tag-llm (Shen et al., 2024), CoLLEGe (Teehan et al., 2024)
Others: ICL (Gao et al., 2024), Role Playing (Ge et al., 2023), Functions (Todd et al., 2024)

Table 1: Summary of methods and applications in prompt compression.

{Instruction, Input, Output}
Example in Alpaca:
Instruction: Evaluate this sentence for spelling and grammar mistakes
Input: He finnished his meal and left the resturant
Output: There are two spelling errors in the sentence. The corrected sentence should be: "He finished his meal and left the restaurant."
{Context, Question, Answer}
Example in SQuAD:
Context: Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California. As this was the 50th Super Bowl, the league emphasized the "golden anniversary" with various gold-themed initiatives, as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as "Super Bowl L"), so that the logo could prominently feature the Arabic numerals 50.
Question: Where did Super Bowl 50 take place?
Answer: Santa Clara, California

Table 2: Examples for the QA items with formats of {Instruction, Input, Output} and {Context, Question, Answer}.