

LLMLingua: Compressing Prompts for Accelerated Inference of Large Language Models

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, Lili Qiu

Microsoft Corporation

{hjiang, qianhuiwu, cyl, yuqing.yang, liliqui}@microsoft.com

Abstract

Large language models (LLMs) have been applied in various applications due to their astonishing capabilities. With advancements in technologies such as chain-of-thought (CoT) prompting and in-context learning (ICL), the prompts fed to LLMs are becoming increasingly lengthy, even exceeding tens of thousands of tokens. To accelerate model inference and reduce cost, this paper presents *LLMLingua*, a coarse-to-fine prompt compression method that involves a budget controller to maintain semantic integrity under high compression ratios, a token-level iterative compression algorithm to better model the interdependence between compressed contents, and an instruction tuning based method for distribution alignment between language models. We conduct experiments and analysis over four datasets from different scenarios, *i.e.*, GSM8K, BBH, ShareGPT, and Arxiv-March23; showing that the proposed approach yields state-of-the-art performance and allows for up to 20x compression with little performance loss.¹

1 Introduction

The widespread adoption of ChatGPT has transformed numerous scenarios by harnessing the powerful generalization and reasoning capabilities of large language models (LLMs). In practical applications, crafting suitable prompts is crucial and usually involves techniques such as chain-of-thought, in-context learning, and retrieving related documents or historical conversations (Wei et al., 2022; Chase, 2022). While these methods can elicit highly effective generations by activating LLMs’ domain-specific knowledge, they often require longer prompts. Therefore, striking a balance between the massive computational demands of LLMs and the need for longer prompts has become an urgent issue. Some studies attempt to accelerate model inference by modifying the parameters of

LLMs through quantization (Dettmers et al., 2022; Xiao et al., 2023), compression (Frantar and Alishtarh, 2023), etc. However, these approaches may be not suitable when the LLMs can be accessed via APIs only.

Approaches that attempt to reduce the length of original prompts while preserving essential information have emerged lately. These approaches are grounded in the concept that natural language is inherently redundant (Shannon, 1951) and thus can be compressed. Gilbert et al. (2023) also indicate that LLMs can effectively reconstruct source code from compressed text descriptions while maintaining a high level of functional accuracy. Therefore, we follow this line of studies to compress a long prompt into a shorter one without any gradient flow through the LLMs to support applications based on a larger range of LLMs.

In terms of information entropy, tokens with lower perplexity (PPL) contribute less to the overall entropy gains of the language model. In other words, removing tokens with lower perplexity has a relatively minor impact on the LLM’s comprehension of the context. Motivated by this, Li (2023) propose Selective-Context, which first employs a small language model to compute the self-information of each lexical unit (such as sentences, phrases, or tokens) in original prompts, and then drops the less informative content for prompt compression. However, this method not only ignores the interdependence between the compressed contents but also neglects the correspondence between the LLM being targeted and the small language model used for prompt compression.

This paper proposes *LLMLingua*, a coarse-to-fine prompt compression method, to address the aforementioned issues. Specifically, we first present a budget controller to dynamically allocate different compression ratios to various components in original prompts such as the instruction, demonstrations, and the question, and meanwhile,

¹Our code is available at <https://aka.ms/LLMLingua>.

LLMLingua: 压缩提示以加速大型语言模型的推理

江辉强, 吴倩慧, 林振耀, 杨宇清, 邱莉莉 微软公司 {hjiang, qianh
uiwu, cyl, yuqing.yang, liliqui}@microsoft.com

摘要

大型语言模型 (LLMs) 因其惊人的能力而被应用于各种应用中。随着链式思维 (CoT) 提示和上下文学习 (ICL) 等技术的进步, 输入到 LLMs 的提示变得越来越冗长, 甚至超过数十万个标记。为了加速模型推理并降低成本, 本文提出了 *LLMLingua*, 一种粗到细的提示压缩方法, 该方法涉及一个预算控制器, 以在高压比下保持语义完整性, 一个令牌级迭代压缩算法, 以更好地建模压缩内容之间的相互依赖关系, 以及一种基于指令调优的方法, 用于语言模型之间的分布对齐。我们在来自不同场景的四个数据集上进行实验和分析, *i.e.*、GSM8K、BBH、ShareGPT 和 Arxiv-March23; 结果表明, 所提出的方法实现了最先进的性能, 并允许在几乎没有性能损失的情况下实现高达 20 倍的压缩。¹

1 引言

ChatGPT 的广泛应用通过利用大型语言模型 (LLMs) 强大的泛化和推理能力, 改变了许多场景。在实际应用中, 制作合适的提示至关重要, 通常涉及诸如思维链、上下文学习和检索相关文档或历史对话等技术 (Wei et al., 2022; Chase, 2022)。虽然这些方法可以通过激活 LLMs 的领域特定知识来引发高效的生成, 但它们通常需要更长的提示。因此, 在 LLMs 巨大的计算需求和对更长提示的需求之间取得平衡已成为一个紧迫的问题。一些研究试图通过修改 $\{v^*\}$ 的参数来加速模型推理。

通过量化 (Dettmers et al., 2022; Xiao et al., 2023)、压缩 (Frantar 和 Alistarh, 2023) 等方法进行 LLMs。然而, 当 LLMs 只能通过 API 访问时, 这些方法可能不适用。

最近出现了一些试图在保留基本信息的同时减少原始提示长度的方法。这些方法基于自然语言本质上是冗余的这一概念 (Shannon, 1951), 因此可以进行压缩。Gilbert 等人 (2023) 还指出, LLMs 可以有效地从压缩的文本描述中重建源代码, 同时保持较高的功能准确性。因此, 我们遵循这一研究方向, 将长提示压缩为更短的提示, 而不通过 LLMs 进行任何梯度流, 以支持基于更大范围 LLMs 的应用。

在信息熵方面, 具有较低困惑度 (PPL) 的标记对语言模型的整体熵增贡献较小。换句话说, 去除具有较低困惑度的标记对 LLM 对上下文的理解影响相对较小。基于此, Li (2023) 提出了选择性上下文, 首先使用一个小型语言模型计算原始提示中每个词汇单元 (如句子、短语或标记) 的自信息, 然后去掉信息量较少的内容以进行提示压缩。然而, 这种方法不仅忽视了压缩内容之间的相互依赖性, 还忽略了目标 LLM 与用于提示压缩的小型语言模型之间的对应关系。

本文提出了 *LLMLingua*, 一种粗到细的提示压缩方法, 以解决上述问题。具体而言, 我们首先提出了一个预算控制器, 以动态分配不同的压缩比给原始提示中的各个组件, 如指令、示例和问题, 同时,

¹Our code is available at <https://aka.ms/LLMLingua>.

perform coarse-grained, demonstration-level compression to maintain semantic integrity under high compression ratios. We further introduce a token-level iterative algorithm for fine-grained prompt compression. Compared with *Selective Context*, it can better preserve the key information within the prompt by taking into account the conditional dependencies between tokens. Additionally, we pose the challenge of distribution discrepancy between the target LLM and the small language model used for prompt compression, and further propose an instruction tuning based method to align the distribution of both language models.

We validate the effectiveness of our approach on four datasets from different domains, *i.e.*, GSM8K and BBH for reasoning and ICL, ShareGPT for conversation, and Arxiv-March23 for summarization. The results show that our method yields state-of-the-art performance across the board. Furthermore, we conduct extensive experiments and discussions to analyze why our approach attains superior performance. To our best knowledge, we are the first to evaluate reasoning and ICL capabilities in the domain of efficient LLMs.

2 Related Work

2.1 Efficient LLMs

Efficient large language models have gained significant attention in recent research community, especially with the growing prominence of ChatGPT. Most of these methods aim to reduce the costs of inference and fine-tuning by modifying the model parameters through quantization (Dettmers et al., 2022; Frantar et al., 2023; Xiao et al., 2023), compression (Frantar and Alistarh, 2023), instruct tuning (Taori et al., 2023; Chiang et al., 2023; Xu et al., 2023), or delta tuning (Hu et al., 2022).

A line of studies attempt to optimize inference costs from the perspective of the input prompts. Motivated by the observation of the abundance of identical text spans between the input and the generated result, Yang et al. (2023) directly copy tokens from prompts for decoding to accelerate the inference process of LLMs. Some approaches focus on compressing prompts, specifically, learning special tokens via prompt tuning of LLMs to reduce the number of tokens to be processed during inference (Mu et al., 2023; Ge et al., 2022; Wingate et al., 2022; Chevalier et al., 2023; Ge et al., 2023). Unfortunately, these methods are usually tailored to particular tasks and some of them (Mu et al., 2023;

Chevalier et al., 2023) even require to fine-tune the whole language model, which severely limits their application scenarios. Furthermore, there are some studies (Chase, 2022; Zhang et al., 2023) that attempt to utilize LLMs to summarize dialog or data, thereby forming memory and knowledge. However, these approaches require multiple invocations of LLMs, which are quite costly.

Some methods reduce the prompt length by selecting a subset of demonstrations. For example, Zhou et al. (2023) introduces a reinforcement learning based algorithm to allocate a specific number of demonstrations for each question. Some other methods focus on token pruning (Goyal et al., 2020; Kim and Cho, 2021; Kim et al., 2022; Rao et al., 2021; Modarressi et al., 2022) and token merging (Bolya et al., 2023). However, these approaches are proposed for smaller models such as BERT, ViT. Moreover, they depend on fine-tuning the models or obtaining intermediate results during inference.

The most similar work to this paper is *Selective-Context* (Li, 2023), which evaluates the informativeness of lexical units by computing self-information with a small language model, and drops the less informative content for prompt compression. This paper is inspired by *Selective-Context* and further proposes a coarse-to-fine framework to address its limitations.

2.2 Out-of-Distribution (OoD) Detection

Recently, a series of studies have been proposed for unsupervised OoD detection. With only in-distribution texts available for learning, these methods either fine-tune a pre-trained language model (Arora et al., 2021) or train a language model from scratch (Mai et al., 2022). Wu et al. (2023) analyze the characteristics of these methods and leverage multi-level knowledge distillation to integrate their strengths while mitigating their limitations. Finally, perplexity output by the resulting language model is used as the indication of an example being OoD.

This paper also regards perplexity as a measurement of how well a language model predicts a sample. In contrast to out-of-distribution detection, which identifies examples with high perplexities as indicative of unreliable predictions, we consider tokens with higher perplexity to be more influential during the inference process of language models.

2.3 LLMs as a Compressor

Recently, many perspectives have interpreted large language models and unsupervised learn-

进行粗粒度的演示级压缩，以在高压比下保持语义完整性。我们进一步引入了一种基于令牌的迭代算法，用于细粒度的提示压缩。与 *Selective Context* 相比，它通过考虑令牌之间的条件依赖关系，更好地保留提示中的关键信息。此外，我们提出了目标 LLM 与用于提示压缩的小语言模型之间的分布差异挑战，并进一步提出了一种基于指令调优的方法，以对齐两个语言模型的分布。

我们在来自不同领域的四个数据集上验证了我们方法的有效性，*i.e.*、GSM8K 和 BBH 用于推理和 ICL，ShareGPT 用于对话，Arxiv-March23 用于摘要。结果表明，我们的方法在各个方面都达到了最先进的性能。此外，我们进行了广泛的实验和讨论，以分析为什么我们的方法能够获得优越的性能。据我们所知，我们是第一个在高效 LLM 领域评估推理和 ICL 能力的研究。

2 相关工作

2.1 高效的 LLMs

高效的大型语言模型在最近的研究界获得了显著关注，尤其是在 ChatGPT 日益突出的背景下。这些方法大多数旨在通过量化 (Dettmers et al., 2022; Frantar et al., 2023; Xiao et al., 2023)、压缩 (Frantar 和 Alistarh, 2023)、指令调优 (Taori et al., 2023; Chiang et al., 2023; Xu et al., 2023) 或增量调优 (Hu et al., 2022) 来降低推理和微调的成本。

一系列研究试图从输入提示的角度优化推理成本。受到输入和生成结果之间相同文本片段丰富性的观察的启发，Yang 等人 (2023) 直接从提示中复制标记进行解码，以加速 LLM 的推理过程。一些方法专注于压缩提示，具体来说，通过对 LLM 的提示调优学习特殊标记，以减少推理过程中需要处理的标记数量 (Mu 等人, 2023; Ge 等人, 2022; Wingate 等人, 2022; Chevalier 等人, 2023; Ge 等人, 2023)。不幸的是，这些方法通常是针对特定任务量身定制的，其中一些 (Mu 等人, 2023;

Chevalier 等人 (2023) 甚至要求对整个语言模型进行微调，这严重限制了它们的应用场景。此外，还有一些研究 (Chase, 2022; Zhang 等人, 2023) 试图利用 LLM 来总结对话或数据，从而形成记忆和知识。然而，这些方法需要多次调用 LLM，这非常昂贵。

一些方法通过选择演示的子集来减少提示长度。例如，Zhou 等人 (2023) 提出了一种基于强化学习的算法，为每个问题分配特定数量的演示。其他一些方法则专注于令牌修剪 (Goyal 等人, 2020; Kim 和 Cho, 2021; Kim 等人, 2022; Rao 等人, 2021; Modarressi 等人, 2022) 和令牌合并 (Bolya 等人, 2023)。然而，这些方法是为较小的模型如 BERT、ViT 提出的。此外，它们依赖于微调模型或在推理过程中获得中间结果。

与本文最相似的工作是选择性上下文 (Li, 2023)，该工作通过使用小型语言模型计算自信息来评估词汇单元的信息量，并删除不太信息丰富的内容以进行提示压缩。本文受到选择性上下文的启发，进一步提出了一种粗到细的框架来解决其局限性。

2.2 分布外 (OoD) 检测

最近，提出了一系列用于无监督 OoD 检测的研究。仅使用可用于学习的分布内文本，这些方法要么微调预训练的语言模型 (Arora 等, 2021)，要么从头开始训练语言模型 (Mai 等, 2022)。Wu 等 (2023) 分析了这些方法的特征，并利用多层次知识蒸馏来整合它们的优点，同时减轻它们的局限性。最后，结果语言模型输出的困惑度被用作示例为 OoD 的指示。

本文还将困惑度视为语言模型预测样本的衡量标准。与识别高困惑度示例作为不可靠预测的指示的分布外检测相反，我们认为在语言模型的推理过程中，具有更高困惑度的标记更具影响力。

2.3 LLMs 作为压缩器

最近，许多观点对大型语言模型和无监督学习进行了阐释。

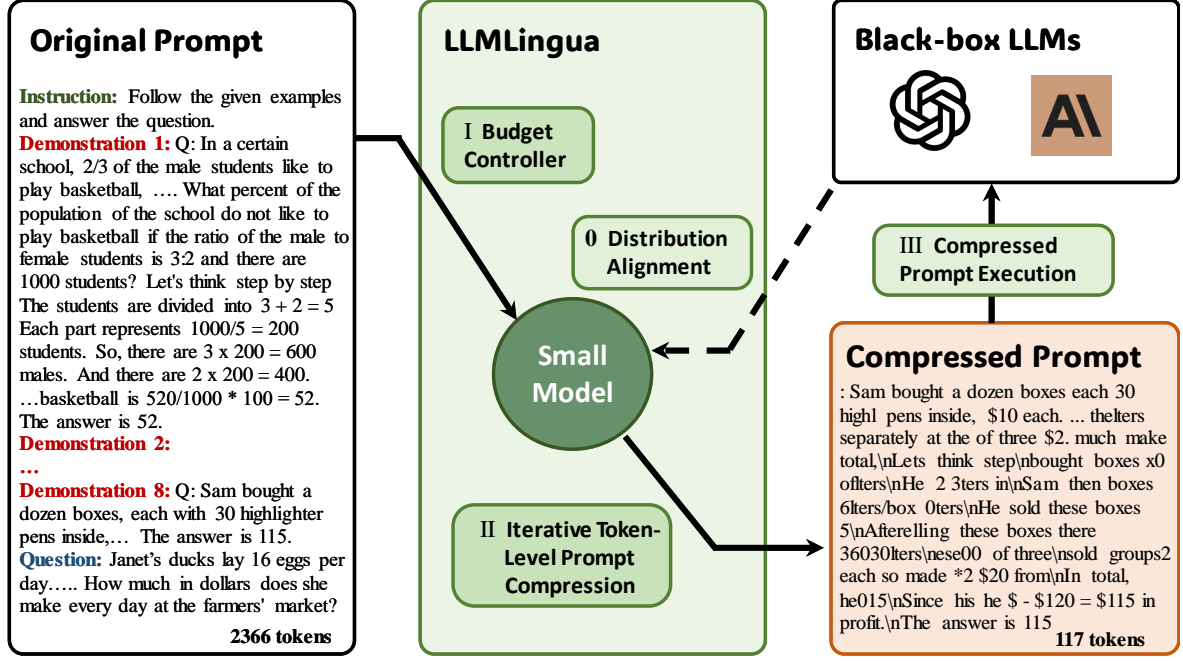


Figure 1: Framework of the proposed approach *LLMLingua*.

ing as a kind of compressor for world knowledge (Sutskever, 2023; Delétang et al., 2023), by using arithmetic coding (Rissanen, 1976; Pasco, 1976). Our research can be viewed as an endeavor to further compress information within prompts by capitalizing on the compression-like characteristics of large language models.

3 Problem Formulation

A prompt compression system is designed to generate a compressed prompt $\tilde{x} = \{\tilde{x}_i\}_{i=1}^{\tilde{L}}$ from a given original prompt $x = (x^{\text{ins}}, x^{\text{dems}}, x^{\text{que}})$, where $x^{\text{ins}} = \{x_i^{\text{ins}}\}_{i=1}^{L^{\text{ins}}}$, $x^{\text{dems}} = \{x_i^{\text{dems}}\}_{i=1}^{L^{\text{dems}}}$, and $x^{\text{que}} = \{x_i^{\text{que}}\}_{i=1}^{L^{\text{que}}}$ denote the instruction, demonstrations, and the question in the original prompt x . \tilde{L} , L^{ins} , L^{dems} , and L^{que} represent the numbers of tokens in \tilde{x} , x^{ins} , x^{dems} , and x^{que} , respectively. Let $L = L^{\text{ins}} + L^{\text{dems}} + L^{\text{que}}$ denote the total sequence length of x , the compression rate is defined as $\tau = \tilde{L}/L$, $\tau \in [0, 1]$, and the compression ratio is $1/\tau$. A smaller value of τ implies a lower inference cost, which is preferable. Let \tilde{x}_G represent the LLM-generated results derived by \tilde{x} and x_G denotes the tokens derived by x , the distribution of \tilde{x}_G is expected to be as similar to x_G as possible. This can be formulated as:

$$\min_{\tilde{x}, \tau} \text{KL}(P(\tilde{x}_G|\tilde{x}), P(x_G|x)), \quad (1)$$

4 Methodology

In this section, we elaborate on the proposed coarse-to-fine prompt compression approach, *LLMLingua*. First, we introduce a budget controller to dynamically allocate different compression ratios to various components in prompts and meanwhile, perform coarse-grained, demonstration-level compression to maintain semantic integrity under high compression ratios. Next, we describe the proposed iterative prompt algorithm designed to retain knowledge from the prompt while compressing. Finally, we introduce alignment to address the distribution gap between the small model and black-box large models. Figure 1 show the framework.

4.1 Budget Controller

The budget controller here is designed to allocate different budgets, *i.e.*, compression ratio, to different components in a prompt such as instructions, demonstrations, and questions, at the sentence or demonstration level. There are two considerations:

(i) In general, the instruction and the question in a prompt have a direct influence on the generated results, as they should contain all the necessary knowledge to generate the following answer. On the contrary, if there are multiple demonstrations in the original prompt, the conveyed information may be redundant. Therefore, a tailored budget controller is required to allocate more budget (*i.e.*,

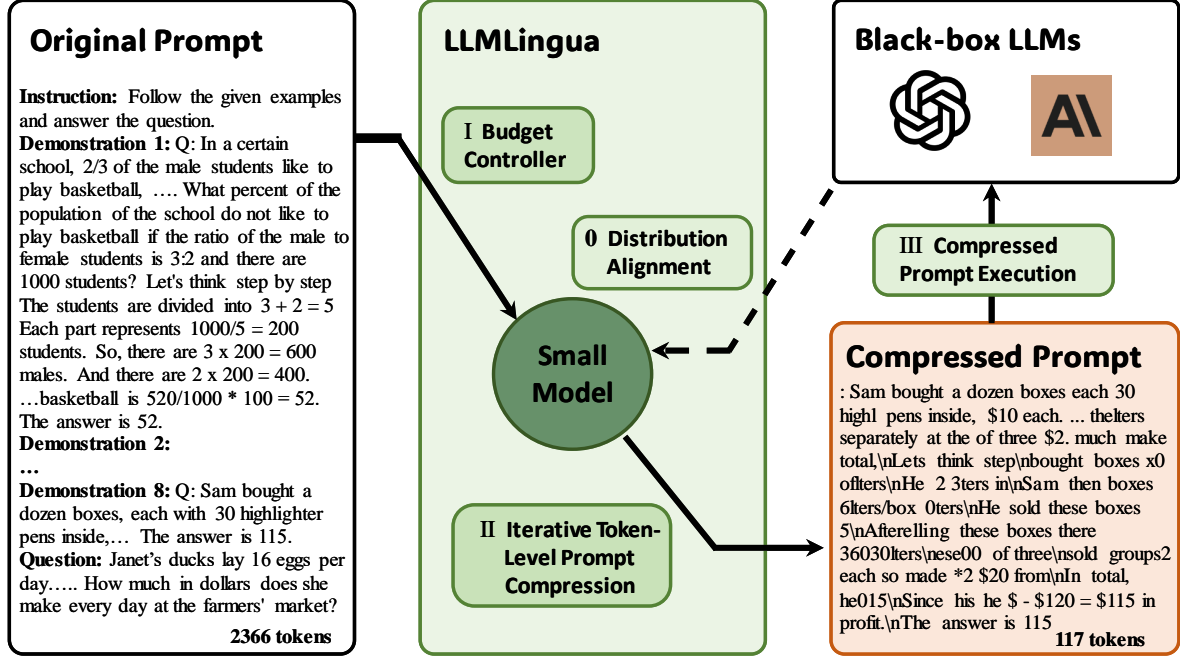


图 1: 所提方法的框架 *LLMLingua*。

作为一种世界知识的压缩器 (Sutskever, 2023; Delétang et al., 2023), 通过使用算术编码 (Rissanen, 1976; Pasco, 1976)。我们的研究可以被视为一种努力, 通过利用大型语言模型的压缩特性, 进一步压缩提示中的信息。

3 问题表述

一个提示压缩系统旨在从给定的原始提示 $x = (x^{\text{ins}}, x^{\text{dems}}, x^{\text{que}})$ 生成一个压缩提示 $\tilde{x} = \{\tilde{x}_i\}_{i=1}^{\tilde{L}}$, 其中 $x^{\text{ins}} = \{x_i^{\text{ins}}\}_{i=1}^{L^{\text{ins}}}$, $x^{\text{dems}} = \{x_i^{\text{dems}}\}_{i=1}^{L^{\text{dems}}}$ 和 $x^{\text{que}} = \{x_i^{\text{que}}\}_{i=1}^{L^{\text{que}}}$ 分别表示原始提示 x 中的指令、示例和问题。 \tilde{x}^{ins} 、 \tilde{x}^{dems} 和 \tilde{x}^{que} 分别表示 \tilde{x} 、 x^{ins} 、 x^{dems} 和 x^{que} 中的标记数量。设 $\tilde{L} = \tilde{L}^{\text{ins}} + \tilde{L}^{\text{dems}} + \tilde{L}^{\text{que}}$ 表示 \tilde{x} 的总序列长度, 压缩率定义为 $\tau = \tilde{L} / L$, $\tau \in [0, 1]$, 压缩比为 $1/\tau$ 。较小的 τ 值意味着较低的推理成本, 这是更可取的。设 \tilde{x}_G 表示由 \tilde{x} 生成的 LLM 结果, x_G 表示由 x 生成的标记, 期望 \tilde{x}_G 的分布尽可能与 x_G 相似。这可以表述为:

$$\min_{\tilde{x}, \tau} \text{KL}(P(\tilde{x}_G|\tilde{x}), P(x_G|x)), \quad (1)$$

4 方法论

在本节中, 我们详细阐述了提出的粗到细提示压缩方法 *LLMLingua*。首先, 我们引入了一个预算控制器, 以动态分配不同的压缩比给提示中的各个组件, 同时进行粗粒度的演示级压缩, 以在高压缩比下保持语义完整性。接下来, 我们描述了旨在在压缩的同时保留提示知识的迭代提示算法。最后, 我们引入对齐, 以解决小模型与黑箱大模型之间的分布差距。图1展示了框架。

4.1 预算控制器

这里的预算控制器旨在将不同的预算 *i.e.*、压缩比分配给提示中的不同组件, 例如指令、演示和问题, 具体到句子或演示级别。有两个考虑因素:

(i) 一般来说, 提示中的指令和问题对生成的结果有直接影响, 因为它们应该包含生成以下答案所需的所有知识。相反, 如果原始提示中有多个示例, 传达的信息可能是多余的。因此, 需要一个量身定制的预算控制器来分配更多预算 (*i.e.*,

Algorithm 1 Pseudo code of Budget Controller.

Input: A small language model M_s ; the original prompt $\mathbf{x} = (\mathbf{x}^{\text{ins}}, \mathbf{x}^{\text{dems}}, \mathbf{x}^{\text{que}})$.

- 1: Set the selected demonstration set $\mathcal{D} = \phi$.
- 2: Get demonstration compression rate τ_{dems} by Eq.(2).
- 3: Calculate the perplexity of each demonstration via M_s .
- 4: Rank all demonstrations in descending order of their perplexity as a list $(\mathbf{x}_{(1)}^{\text{dems}}, \dots, \mathbf{x}_{(N)}^{\text{dems}})$, where N is the number of demonstrations, $\mathbf{x}_{(i)}^{\text{dems}}$ is the i -th demonstration.
- 5: **for** $i = 1$ **do**
- 6: **if** $\tilde{L}_{\mathcal{D}} > k \cdot \tau_{\text{dems}} L_{\text{dems}}$ **then**
- 7: Break.
- 8: **end if**
- 9: Append $\mathbf{x}_{(i)}^{\text{dems}}$ to \mathcal{D} .
- 10: $i = i + 1$
- 11: **end for**
- 12: Allocate remaining budget to \mathbf{x}^{ins} and \mathbf{x}^{que} via Eq. (3).

Output: The subset of demonstrations \mathcal{D} obtained from coarse-grained compression; Additional budget $\Delta\tau_{\text{ins,que}}$ for the instruction and the question.

smaller compression ratios) for instructions and questions, and less budget for demonstrations.

(ii) When a high compression ratio is required, token-level dropout as in Li (2023) might make the compressed prompts too trivial and thus lose vital information from the original prompt. Consequently, sentence-level dropout should be employed instead to preserve a certain degree of linguistic integrity. Especially in the case of multiple redundant demonstrations, we can even perform demonstration-level control to meet the compression requirement.

Algorithm 1 illustrates the overall procedure of the budget controller.

Derive compression ratio for demonstrations.

We first compute the compression rate for demonstrations τ_{dems} according to the target overall compression rate τ and the pre-defined compression rate for instructions and questions, *i.e.*, τ_{ins} and τ_{que} , respectively.

$$\tau_{\text{dems}} = \frac{\tau L - (\tau_{\text{ins}} L_{\text{ins}} + \tau_{\text{que}} L_{\text{que}})}{L_{\text{dems}}}. \quad (2)$$

Demonstration-level prompt compression.

With the derived τ_{dems} for demonstrations, we then perform a coarse-grained demonstration-level prompt compression: we construct \mathcal{D} , a subset of demonstrations from \mathbf{x}^{dems} .

Specifically, we first employ a small language model M_s , such as GPT-2 or LLaMA, to compute the perplexity of each demonstration in \mathbf{x}^{dems} . Then, we select demonstrations in descending order of their perplexity values, until adding one more

Algorithm 2 Pseudo code of Iterative Token-level Prompt Compression (ITPC).

Input: A small language model M_s ; the prompt from budget controller $\mathbf{x}' = (\mathbf{x}^{\text{ins}}, \mathbf{x}^{\mathcal{D}}, \mathbf{x}^{\text{que}})$; target compression rate τ , adjusted compression rate $\Delta\tau_{\text{ins,que}}$.

- 1: Set the selected token set $\mathcal{T} = \phi$
- 2: Get segment set \mathcal{S} .
- 3: **for** $i = 1, 2, \dots, m$ **do**
- 4: Get the conditional probabilities $p(s_i)$ via Eq.(5)
- 5: Get the compression threshold γ_i with Eq. (6).
- 6: Append the compressed token to \mathcal{T} via Eq.(7).
- 7: **end for**
- 8: Concatenate all tokens in \mathcal{T} as $\tilde{\mathbf{x}}$.

Output: The compressed prompt $\tilde{\mathbf{x}}$.

demonstration to \mathcal{D} will make the total number of tokens in \mathcal{D} exceed maximum tokens $k \cdot \tau_{\text{dems}} L_{\text{dems}}$, where k is the granular control coefficient.

Adjust compression ratios for instruction and question.

After obtaining the coarse-grained compression result $\mathcal{D} = \{x_i\}_{i=1}^{\tilde{L}_{\mathcal{D}}}$, we allocate the remaining budget to the instruction and the question:

$$\Delta\tau = \frac{k \cdot \tau_{\text{dems}} L_{\text{dems}} - \tilde{L}_{\mathcal{D}}}{L_{\text{ins}} + L_{\text{que}}}, \quad (3)$$

where $\tilde{L}_{\mathcal{D}}$ denote the total number of tokens in \mathcal{D} .

4.2 Iterative Token-level Prompt Compression

Utilizing perplexity for prompt compression encounters the intrinsic limitation, *i.e.*, the independence assumption, similar to the shortcomings of the Mask Language Model (Yang et al., 2019) as:

$$\begin{aligned} p(\tilde{\mathbf{x}}) &= \prod_{i=1}^{\tilde{L}} p(\tilde{x}_i | \tilde{x}_{<i}) \\ &\approx p(\mathbf{x}') = \prod_{i=1}^{L'} p(x_i | \tilde{x}_{<i}, \bar{x}_{<i}), \end{aligned} \quad (4)$$

where $\mathbf{x}' = (\mathbf{x}^{\text{ins}}, \mathbf{x}^{\mathcal{D}}, \mathbf{x}^{\text{que}})$ is the original prompt after demonstration-level compression; $\mathbf{x}^{\mathcal{D}}$ is the concatenation of all demonstrations in \mathcal{D} ; $\tilde{\mathbf{x}}$ is the final compressed prompt; $\tilde{x}_{<i}$ and $\bar{x}_{<i}$ denote the preserved and compressed tokens before the i -th token x_i ; L' and \tilde{L} denote the numbers of all tokens in \mathbf{x}' and $\tilde{\mathbf{x}}$, respectively.

Here we propose an iterative token-level prompt compression (ITPC) algorithm to mitigate the inaccuracy introduced by the conditional independence assumption. Algorithm 2 shows the pseudo codes.

Specifically, we first divide the target prompt \mathbf{x}' into several segments $\mathcal{S} = \{s_1, s_2, \dots, s_m\}$. And

Algorithm 1 Pseudo code of Budget Controller.

Input: A small language model M_s ; the original prompt $x = (x^{\text{ins}}, x^{\text{dems}}, x^{\text{que}})$.

- 1: Set the selected demonstration set $\mathcal{D} = \phi$.
- 2: Get demonstration compression rate τ_{dems} by Eq.(2).
- 3: Calculate the perplexity of each demonstration via M_s .
- 4: Rank all demonstrations in descending order of their perplexity as a list $(x_{(1)}^{\text{dems}}, \dots, x_{(N)}^{\text{dems}})$, where N is the number of demonstrations, $x_{(i)}^{\text{dems}}$ is the i -th demonstration.
- 5: **for** $i = 1$ **do**
- 6: **if** $\tilde{L}_{\mathcal{D}} > k \cdot \tau_{\text{dems}} L_{\text{dems}}$ **then**
- 7: Break.
- 8: **end if**
- 9: Append $x_{(i)}^{\text{dems}}$ to \mathcal{D} .
- 10: $i = i + 1$
- 11: **end for**
- 12: Allocate remaining budget to x^{ins} and x^{que} via Eq. (3).

Output: The subset of demonstrations \mathcal{D} obtained from coarse-grained compression; Additional budget $\Delta\tau_{\text{ins,que}}$ for the instruction and the question.

较小的压缩比)用于指令和问题,并且用于演示的预算更少。

(ii) 当需要高压缩比时,像Li (2023)中所述的令牌级丢弃可能会使压缩后的提示过于简单,从而失去原始提示中的重要信息。因此,应采用句子级丢弃来保持一定程度的语言完整性。特别是在多个冗余示例的情况下,我们甚至可以进行示例级控制以满足压缩要求。

算法 1 说明了预算控制器的整体过程。

推导演示的压缩比。

我们首先根据目标整体压缩率 τ 和预定义的指令和问题的压缩率 $i.e.$, τ_{ins} 和 τ_{que} , 计算演示 τ_{dems} 的压缩率。

$$\tau_{\text{dems}} = \frac{\tau L - (\tau_{\text{ins}} L_{\text{ins}} + \tau_{\text{que}} L_{\text{que}})}{L_{\text{dems}}}. \quad (2)$$

演示级提示压缩。通过推导出的 τ_{dems} 进行演示后,我们进行粗粒度的演示级提示压缩:我们构建 \mathcal{D} , 这是来自 x^{dems} 的演示子集。

具体来说,我们首先使用一个小型语言模型 M_s , 例如 GPT-2 或 LLaMA, 来计算 x^{dems} 中每个示例的困惑度。然后,我们按困惑度值的降序选择示例,直到再添加一个。

Algorithm 2 Pseudo code of Iterative Token-level Prompt Compression (ITPC).

Input: A small language model M_s ; the prompt from budget controller $x' = (x^{\text{ins}}, x^{\mathcal{D}}, x^{\text{que}})$; target compression rate τ , adjusted compression rate $\Delta\tau_{\text{ins,que}}$.

- 1: Set the selected token set $\mathcal{T} = \phi$
- 2: Get segment set \mathcal{S} .
- 3: **for** $i = 1, 2, \dots, m$ **do**
- 4: Get the conditional probabilities $p(s_i)$ via Eq.(5)
- 5: Get the compression threshold γ_i with Eq. (6).
- 6: Append the compressed token to \mathcal{T} via Eq.(7).
- 7: **end for**
- 8: Concatenate all tokens in \mathcal{T} as \tilde{x} .

Output: The compressed prompt \tilde{x} .

对 \mathcal{D} 的演示将使 \mathcal{D} 中的令牌总数超过最大令牌数 $k \cdot \tau_{\text{dems}} L_{\text{dems}}$, 其中 k 是粒度控制系数。

调整指令和问题的压缩比。在获得粗粒度压缩结果 $\mathcal{D} = \{x_i\}_{i=1}^{\tilde{L}_{\mathcal{D}}}$ 后,我们将剩余预算分配给指令和问题:

$$\Delta\tau = \frac{k \cdot \tau_{\text{dems}} L_{\text{dems}} - \tilde{L}_{\mathcal{D}}}{L_{\text{ins}} + L_{\text{que}}}, \quad (3)$$

其中 $\tilde{L}_{\mathcal{D}}$ 表示 \mathcal{D} 中的总令牌数。

4.2 迭代的令牌级提示压缩

利用困惑度进行提示压缩克服了固有的限制 $i.e.$, 即独立假设,这与掩码语言模型 (Yang et al., 2019) 的缺点类似:

$$\begin{aligned} p(\tilde{x}) &= \prod_{i=1}^{\tilde{L}} p(\tilde{x}_i | \tilde{x}_{<i}) \\ &\approx p(x') = \prod_{i=1}^{L'} p(x_i | \tilde{x}_{<i}, \bar{x}_{<i}), \end{aligned} \quad (4)$$

其中 $x' = (x^{\text{ins}}, x^{\mathcal{D}}, x^{\text{que}})$ 是经过演示级压缩后的原始提示; $x^{\mathcal{D}}$ 是 \mathcal{D} 中所有演示的连接; \tilde{x} 是最终压缩的提示; $\tilde{x}_{<i}$ 和 $\bar{x}_{<i}$ 表示在第 i 个标记之前保留和压缩的标记; L' 和 \tilde{L} 分别表示 x' 和 \tilde{x} 中所有标记的数量。

在这里,我们提出了一种迭代的令牌级提示压缩 (ITPC) 算法,以减轻条件独立假设引入的不准确性。算法 2 显示了伪代码。

具体来说,我们首先将目标提示 x' 分成几个部分 $\mathcal{S} = \{s_1, s_2, \dots, s_m\}$ 。并且

then, we use the smaller model \mathcal{M}_s to obtain the perplexity distribution of all segments. The compressed prompt obtained from each segment is concatenated to the subsequent segment, enabling more accurate estimation of the conditional probability. The corresponding probability estimation function can be formulated as:

$$\begin{aligned} p(\tilde{s}_j) &= \prod_{i=1}^{\sum_k^j \tilde{L}_{s,k}} p(\tilde{s}_{j,i} | \tilde{s}_{j,<i}, \tilde{s}_{<j}) \\ &\approx \prod_{i=1}^{L_{s,j} + \sum_k^{j-1} \tilde{L}_{s,k}} p(s_{j,i} | s_{j,<i}, \tilde{s}_{<j}), \end{aligned} \quad (5)$$

where $s_{j,i}$ denotes the i -th token in the j -th segment, $L_{s,j}$ and $\tilde{L}_{s,j}$ represent the token length of j -th original and compressed segment, respectively.

When the conditional probabilities for each segment $p(s_j)$ are obtained, the compression ratio threshold γ_j w.r.t. s_j are dynamically calculated based on the PPL distribution and the corresponding compression ratio τ_{s_j} , where

$$\tau_{s_j} = \begin{cases} \tau_{\text{ins}} + \Delta\tau, & \text{if } s_j \text{ from } \mathbf{x}^{\text{ins}}, \\ \tau_{\text{dems}}, & \text{if } s_j \text{ from } \mathbf{x}^{\mathcal{D}}, \\ \tau_{\text{que}} + \Delta\tau, & \text{if } s_j \text{ from } \mathbf{x}^{\text{que}}. \end{cases} \quad (6)$$

Finally, tokens in each s_j with the PPL greater than γ_j are retained in the compressed prompt.

$$\tilde{s}_j = \{s_{j,i} | p(s_{j,i}) > \gamma_j\} \quad (7)$$

4.3 Distribution Alignment

To narrow the gap between the distribution of the LLM and that of the small language model used for prompt compression, here we align the two distributions via instruction tuning.

Specifically, we start from a pre-trained small language model \mathcal{M}_s and use the data generated by the LLM to perform instruction tuning on \mathcal{M}_s . The optimization of \mathcal{M}_s can be formulated as:

$$\min_{\theta_s} \mathbb{E} \left[\frac{1}{N} \sum_{i=1}^N \mathcal{L}(\mathbf{x}_i, \mathbf{y}_{i,\text{LLM}}; \theta_{\mathcal{M}_s}) \right], \quad (8)$$

where $\theta_{\mathcal{M}_s}$ denotes the parameters of \mathcal{M}_s , $(x_i, \mathbf{y}_{i,\text{LLM}}^{\text{LLM}})$ denotes the pair of instruction x_i and the LLM generated texts $\mathbf{y}_{i,\text{LLM}}^{\text{LLM}}$, N is the number of all examples used for instruction tuning.

5 Experiments

5.1 Settings

Datasets To comprehensively assess the effectiveness of compressed prompts in retaining LLM abilities, we evaluated their performance across four datasets. For reasoning and in-context learning (ICL), we use **GSM8K** (Cobbe et al., 2021) and **BBH** (Suzgun et al., 2022). As for contextual understanding, we use **ShareGPT** (sha, 2023) for conversation and **Arxiv-March23** (Li, 2023) for summarization. It’s worth noting that neither the small LM nor the target LLMs used in this paper have seen any of the evaluation datasets, especially the last two which were newly collected this year. We followed the experimental setup of previous work (Fu et al., 2023a; Li, 2023) for the usage of these datasets. Please refer to Appendix A.1 for detailed information.

Evaluation Following Cobbe et al. (2021), Fu et al. (2023a), and Li (2023), we utilize the Exact Match as the evaluation metric for GSM8K and BBH. We use BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), and BERTScore (Zhang et al., 2020) as the evaluation metrics for ShareGPT and Arxiv-March23.

Implementation Details In this paper, we employ the GPT-3.5-Turbo-0301 and the Claude-v1.3 as the target LLMs, which can be accessed via OpenAI² and Claude API³. To improve the stability of outputs produced by LLMs we apply greedy decoding with a temperature of 0 across all experiments. The Alpaca dataset (Taori et al., 2023) is exclusively employed for aligning small language models with black-box LLMs, and is not utilized in the evaluation process. In our experiments, we utilize either Alpaca-7B⁴ or GPT2-Alpaca as the small pre-trained language model \mathcal{M}_s for compression. We implement our approach based on PyTorch 1.12.0⁵ and Huggingface’s Transformers⁶. We set the granular control coefficient k to 2. We use the pre-defined compression rates $\tau_{\text{ins}} = 0.85$ and $\tau_{\text{que}} = 0.9$ for instructions and questions. The segment size used in the iterative token-level compression is set to 100.

²<https://platform.openai.com/>

³<https://anthropic.com/>

⁴https://github.com/tatsu-lab/stanford_alpaca

⁵<https://pytorch.org/>

⁶<https://github.com/huggingface/transformers>

然后，我们使用较小的模型 M_s 来获得所有片段的困惑度分布。从每个片段获得的压缩提示被连接到后续片段，从而能够更准确地估计条件概率。相应的概率估计函数可以表述为：

$$p(\tilde{s}_j) = \prod_{i=1}^{\sum_k \tilde{L}_{s,k}} p(\tilde{s}_{j,i} | \tilde{s}_{j,<i}, \tilde{s}_{<j}) \quad (5)$$

$$\approx \prod_{i=1}^{L_{s,j} + \sum_k^{j-1} \tilde{L}_{s,k}} p(s_{j,i} | s_{j,<i}, \tilde{s}_{<j}),$$

其中 $s_{j,i}$ 表示第 j 段中的第 i 个标记， $s_{s,j}$ 和 $\tilde{s}_{s,j}$ 分别表示第 s 段原始和压缩段的标记长度。

当每个段落的条件概率 (s_j) 被获得时，压缩比阈值 γ_j w.r.t. s_j 会根据 PPL 分布和相应的压缩比 s_j 动态计算，其中

$$\tau_{s_j} = \begin{cases} \tau_{\text{ins}} + \Delta\tau, & \text{if } s_j \text{ from } \mathbf{x}^{\text{ins}}, \\ \tau_{\text{dems}}, & \text{if } s_j \text{ from } \mathbf{x}^{\mathcal{D}}, \\ \tau_{\text{que}} + \Delta\tau, & \text{if } s_j \text{ from } \mathbf{x}^{\text{que}}. \end{cases} \quad (6)$$

最后， s_j 中 PPL 大于 γ_j 的标记被保留在压缩提示中。

$$\tilde{s}_j = \{s_{j,i} | p(s_{j,i}) > \gamma_j\} \quad (7)$$

4.3 分布对齐

为了缩小 LLM 的分布与用于提示压缩的小语言模型的分布之间的差距，我们通过指令调优对这两个分布进行对齐。

具体来说，我们从一个预训练的小型语言模型 M_s 开始，并使用 LLM 生成的数据对 M_s 进行指令调优。 M_s 的优化可以表述为：

$$\min_{\theta_s} \mathbb{E} \left[\frac{1}{N} \sum_{i=1}^N \mathcal{L}(\mathbf{x}_i, \mathbf{y}_i^{\text{LLM}}; \theta_{M_s}) \right], \quad (8)$$

其中 θ_{M_s} 表示 M_s 的参数， $(\mathbf{x}_i, \mathbf{y}_i^{\text{LLM}})$ 表示指令 \mathbf{x}_i 的一对和 LLM 生成的文本 $\mathbf{y}_i^{\text{LLM}}$ ， N 是用于指令调优的所有示例的数量。

5 个实验

5.1 设置

数据集 为了全面评估压缩提示在保留 LLM 能力方面的有效性，我们在四个数据集上评估了它们的表现。对于推理和上下文学习 (ICL)，我们使用 GSM8K (Cobbe 等, 2021) 和 BBH (Suzgun 等, 2022)。至于上下文理解，我们使用 ShareGPT (sha, 2023) 进行对话，使用 Arxiv-March23 (Li, 2023) 进行摘要。值得注意的是，本文中使用的目标 LM 和小型 LM 都没有见过任何评估数据集，尤其是最后两个数据集是今年新收集的。我们遵循了之前工作的实验设置 (Fu 等, 2023a; Li, 2023) 来使用这些数据集。有关详细信息，请参阅附录 A.1。

评估 根据 Cobbe 等人 (2021)、Fu 等人 (2023a) 和 Li (2023) 的研究，我们将精确匹配作为 GSM8K 和 BBH 的评估指标。我们使用 BLEU (Papineni 等人, 2002)、ROUGE (Lin, 2004) 和 BERTScore (Zhang 等人, 2020) 作为 ShareGPT 和 Arxiv-March23 的评估指标。

实现细节 在本文中，我们采用 GPT-3.5-Turbo-0301 和 Claude-v1.3 作为目标 LLM，可以通过 OpenAI² 和 Claude API³ 访问。为了提高 LLM 生成输出的稳定性，我们在所有实验中应用温度为 0 的贪婪解码。Alpaca 数据集 (Taori 等, 2023) 专门用于将小型语言模型与黑箱 LLM 对齐，并未在评估过程中使用。在我们的实验中，我们使用 Alpaca-7B⁴ 或 GPT2-Alpaca 作为小型预训练语言模型 M_s 进行压缩。我们基于 PyTorch 1.12.0⁵ 和 Huggingface 的 Transformers⁶ 实现我们的方法。我们将细粒度控制系数 α 设置为 0。我们对指令和问题使用预定义的压缩率 $\tau_{\text{ins}} = 0.85$ 和 $\tau_{\text{que}} = 0.9$ 。在迭代的令牌级压缩中使用的段大小设置为 100。

²<https://platform.openai.com/>

³<https://anthropic.com/>

⁴https://github.com/tatsu-lab/stanford_alpaca

⁵<https://pytorch.org/>

⁶<https://github.com/huggingface/transformers>

Methods	ShareGPT								Arxiv-March23							
	BLEU	Rouge1	Rouge2	RougeL	BS F1	Tokens	1/ τ		BLEU	Rouge1	Rouge2	RougeL	BS F1	Tokens	1/ τ	
Constraint I	<i>2x constraint</i>								<i>350 tokens constraint</i>							
Sentence Selection	28.59	46.11	31.07	37.94	88.64	388	1.5x		22.77	50.1	25.93	33.63	88.21	379	4x	
Selective-Context	25.42	46.47	29.09	36.99	88.92	307	1.9x		21.41	51.3	27.94	36.73	89.60	356	4x	
Ours	27.36	48.87	30.32	38.55	89.52	304	1.9x		23.15	54.21	32.66	42.74	90.33	345	4x	
Constraint II	<i>3x constraint</i>								<i>175 tokens constraint</i>							
Sentence Selection	18.94	35.17	18.96	26.75	85.63	255	2.3x		12.41	38.91	14.25	26.72	87.09	229	7x	
Selective-Context	15.79	38.42	20.55	28.89	87.12	180	3.3x		12.23	42.47	19.48	29.47	88.16	185	8x	
Ours	19.55	40.81	22.68	30.98	87.70	177	3.3x		13.45	44.36	24.86	34.94	89.03	176	9x	

Table 1: Performance of different methods under different target compression ratios on the conversation (ShareGPT) and summarization (Arxiv-March23) task.

Methods	GSM8K			BBH		
	EM	Tokens	1/ τ	EM	Tokens	1/ τ
Full-shot	78.85	2,366	-	70.07	774	-
1-shot constraint						
1-shot	77.10	422	6x	69.60	284	3x
Selective-Context	53.98	452	5x	54.27	276	3x
GPT4 Generation	71.87	496	5x	27.13	260	3x
Ours	79.08	446	5x	70.11	288	3x
half-shot constraint						
Sentence Selection	72.33	230	10x	39.56	175	4x
Selective-Context	52.99	218	11x	54.02	155	5x
GPT4 Generation	68.61	223	11x	27.09	161	5x
Ours	77.41	171	14x	61.60	171	5x
quarter-shot constraint						
Sentence Selection	66.67	195	12x	46.00	109	7x
Selective-Context	44.20	157	15x	47.37	108	7x
GPT4 Generation	56.33	188	20x	26.81	101	8x
Ours	77.33	117	20x	56.85	110	7x
zero-shot	48.75 [†]	11	215x	32.32	16	48x
Simple Prompt	74.9	691	3x	-	-	-

Table 2: Performance of different methods under different target compression ratios on the GSM8K mathematical reasoning and Big-bench Hard (BBH) datasets. [†]We also include the instruction of the prompt in zero-shot experiments for a vertical comparison.

Baselines We consider the following baselines:

- *GPT4-Generation*: Instruct GPT-4 to compress the original prompt. We used ten sets of instructions here and reported the best results. Appendix C displays the instructions we employed.
- *Random Selection*: Random select the demonstrations or sentences of the original prompt.
- *Selective-Context* (Li, 2023): Use the phrase-level self-information from a small language model to filter out less informative content. We use the same small LM, *i.e.*, Alpaca-7B for a fair comparison.

5.2 Main Results

Table 1 and 2 report the results of our approach alongside those baseline methods on GSM8K,

BBH, ShareGPT, and Arxiv-March23. It can be seen that our proposed method consistently outperforms the prior methods by a large margin in almost all experiments.

Specifically, on GSM8K and BBH, the reasoning and in-context learning-related benchmark, our method even achieves slightly higher results than the full-shot approach, while also delivering impressive compression ratios ($1/\tau$) of 5x and 3x respectively, with the 1-shot constraint. This well demonstrates that our compressed prompts effectively retain the reasoning information contained in the original prompt. As the compression ratio increases, *i.e.*, under the half-shot and quarter-shot constraints, the performance experiences a slight decline. For instance, on GSM8K, the EM scores will decrease by 1.44 and 1.52, respectively, despite compression ratios as high as 14x and 20x. On BBH, our approach achieves compression ratios of 5x and 7x with the EM score decreasing by 8.5 and 13.2 points, respectively. In fact, this performance is already quite satisfactory, as it approaches the score of 62.0 achieved by PaLM-540B in half-shot constraint. Our case study reveals that this declined performance on BBH is mainly due to challenging reasoning tasks, such as tracking_shuffled_objects_seven_objects.

Moreover, on ShareGPT and Arxiv-March23, two contextual understanding benchmarks, we can see that our approach achieves acceleration ratios of 9x and 3.3x with a high BERTScore F1, indicating that our approach successfully retains the semantic information of the initial prompts.

5.3 Analysis on Reasoning & ICL Tasks.

Here we analyze the performance of our approach and baseline methods on the difficult reasoning and in-context learning (ICL) benchmarks GSM8K and BBH.

We notice that our approach shows significant

Methods	ShareGPT								Arxiv-March23							
	BLEU	Rouge1	Rouge2	RougeL	BS F1	Tokens	1/ τ		BLEU	Rouge1	Rouge2	RougeL	BS F1	Tokens	1/ τ	
Constraint I	<i>2x constraint</i>								<i>350 tokens constraint</i>							
Sentence Selection	28.59	46.11	31.07	37.94	88.64	388	1.5x		22.77	50.1	25.93	33.63	88.21	379	4x	
Selective-Context	25.42	46.47	29.09	36.99	88.92	307	1.9x		21.41	51.3	27.94	36.73	89.60	356	4x	
Ours	27.36	48.87	30.32	38.55	89.52	304	1.9x		23.15	54.21	32.66	42.74	90.33	345	4x	
Constraint II	<i>3x constraint</i>								<i>175 tokens constraint</i>							
Sentence Selection	18.94	35.17	18.96	26.75	85.63	255	2.3x		12.41	38.91	14.25	26.72	87.09	229	7x	
Selective-Context	15.79	38.42	20.55	28.89	87.12	180	3.3x		12.23	42.47	19.48	29.47	88.16	185	8x	
Ours	19.55	40.81	22.68	30.98	87.70	177	3.3x		13.45	44.36	24.86	34.94	89.03	176	9x	

表1: 在对话 (ShareGPT) 和摘要 (Arxiv-March23) 任务中, 不同方法在不同目标压缩比下的性能。

Methods	GSM8K			BBH		
	EM	Tokens	1/ τ	EM	Tokens	1/ τ
Full-shot	78.85	2,366	-	70.07	774	-
1-shot constraint						
1-shot	77.10	422	6x	69.60	284	3x
Selective-Context	53.98	452	5x	54.27	276	3x
GPT4 Generation	71.87	496	5x	27.13	260	3x
Ours	79.08	446	5x	70.11	288	3x
half-shot constraint						
Sentence Selection	72.33	230	10x	39.56	175	4x
Selective-Context	52.99	218	11x	54.02	155	5x
GPT4 Generation	68.61	223	11x	27.09	161	5x
Ours	77.41	171	14x	61.60	171	5x
quarter-shot constraint						
Sentence Selection	66.67	195	12x	46.00	109	7x
Selective-Context	44.20	157	15x	47.37	108	7x
GPT4 Generation	56.33	188	20x	26.81	101	8x
Ours	77.33	117	20x	56.85	110	7x
zero-shot	48.75 [†]	11	215x	32.32	16	48x
Simple Prompt	74.9	691	3x	-	-	-

表2: 在GSM8K数学推理和Big-bench Hard (BBH) 数据集上, 不同目标压缩比下不同方法的性能。[†] 我们还在零-shot实验中包含了提示的指令, 以便进行纵向比较。

基准 我们考虑以下基准:

- *GPT4-Generation* 指示 GPT-4 压缩原始提示。我们在这里使用了十组指令, 并报告了最佳结果。附录 C 显示了我们使用的指令。
- *Random Selection* 随机选择原始提示的演示或句子。
- *Selective-Context* (李, 2023): 使用小型语言模型的短语级自信息来过滤掉信息量较少的内容。我们使用相同的小型语言模型 *i.e.*, Alpaca-7B, 以便进行公平比较。

5.2 主要结果

表1和表2报告了我们的方法与这些基线方法在GSM8K上的结果,

BBH、ShareGPT 和 Arxiv-March23。可以看出, 我们提出的方法在几乎所有实验中都大幅超越了之前的方法。

具体来说, 在GSM8K和BBH这两个与推理和上下文学习相关的基准上, 我们的方法甚至在1-shot约束下取得了比全样本方法略高的结果, 同时也实现了令人印象深刻的压缩比 (1/) 分别为5倍和3倍。这很好地表明我们的压缩提示有效地保留了原始提示中包含的推理信息。随着压缩比的增加, *i.e.*, 在半样本和四分之一样本的约束下, 性能略有下降。例如, 在GSM8K上, EM分数将分别下降1.44和1.52, 尽管压缩比高达14倍和20倍。在BBH上, 我们的方法实现了5倍和7倍的压缩比, EM分数分别下降了8.5和13.2分。事实上, 这一表现已经相当令人满意, 因为它接近于PaLM-540B在半样本约束下取得的62.0分。我们的案例研究表明, BBH上性能下降主要是由于具有挑战性的推理任务, 例如tracking_shuffled_objects_seven_objects。

此外, 在ShareGPT和Arxiv-March23这两个上下文理解基准上, 我们可以看到我们的方法实现了9倍和3.3倍的加速比, 并且具有较高的BERTScore F1, 这表明我们的方法成功保留了初始提示的语义信息。

5.3 关于推理和 ICL 任务的分析。

在这里, 我们分析了我们的方法和基线方法在困难的推理和上下文学习 (ICL) 基准测试GSM8K和BBH上的表现。

我们注意到我们的方法显示出显著的

performance improvements over the strong baseline Selective-Context under all settings. We conjecture that, as relying on phrase-level self-information, Selective-Context is prone to lose critical reasoning information during the chain-of-thought process. Especially on GSM8K, its performance is lower than ours by 33.10 points at a compression ratio of 20x. The inferior performance of Sentence Selection suggests that it may face similar issues of fragmentary reasoning logic. Surprisingly, though GPT-4 has demonstrated its strong text generation capability, the suboptimal performance on prompt compression indicates that the generated prompts may omit crucial details from the original prompt, particularly reasoning steps.

In addition to the findings mentioned above, the experiments also demonstrate that our method can preserve the ICL capacity of prompts for LLMs. Compared to the zero-shot results, our approach exhibits significant performance improvements of 51.55 and 24.53 even with the largest compression ratios. Notably, on GSM8K, our 20x compressed prompt outperforms the 8-shot 3-step CoT by 2.43, further suggesting that our method can effectively retain the reasoning information.

5.4 Ablation

To validate the contributions of different components in our approach, we introduce five variants of our model for ablation study: i) *Ours w/o Iterative Token-level Compression*, which performs token-level compression in a single inference rather than iteratively. ii) *Ours w/o Budget Controller*, which directly employs ITPC with the same compression ratio for all components. iii) *Ours w/o Dynamic Compression Ratio*, which uses the same compression ratio for all components. iv) *Ours w/ Random Selection in Budget Controller*, which randomly selects demonstrations or sentences for demonstration-level prompt compression. v) *Ours w/o Distribution Alignment*, which removes the distribution alignment module of our approach and directly use the pre-trained LLaMA-7B as the small language model. vi) *Ours w/ Remove Stop Words*, which removes the stop words in original prompts using NLTK⁷. Table 3 shows the results.

Comparing Ours with w/o Iterative Token-level Prompt Compression, we observe a significant decline in Exact Match when the conditional dependence between compressed tokens is not consid-

	EM	Tokens	1/ τ
Ours	79.08	439	5x
- w/o Iterative Token-level Prompt Compression	72.93	453	5x
- w/o Budget Controller	73.62	486	5x
- w/o Dynamic Compression Ratio	77.26	457	5x
- w/ Random Selection in Budget Controller	72.78	477	5x
- w/o Distribution Alignment	78.62	452	5x
- w/ Remove Stop Words	76.27	1,882	1.3x

Table 3: Ablation study on GSM8K in 1-shot constraint.

ered. We conjecture this variant may lose essential information in the prompt, especially for low-frequency keywords that frequently appear in the given prompt. When comparing Ours with w/o Dynamic Compression Ratio and with w/o Budget Controller, it reveals that different components of the prompt exhibit varying sensitivity. Instructions and questions necessitate a lower compression ratio. To balance the relationship between compression ratio and language integrity, introducing a demonstration or sentence-level filter better preserves sufficient linguistic information, even at higher compression ratios. Ours w/ Random Selection in Budget Controller indicates that selecting sentences or demonstrations based on perplexity can better identify information-rich sentences for target LLMs. Distribution Alignment allows small LMs to generate distributions that more closely resemble those of target LLMs, resulting in a further improvement of 0.56 on GSM8K.

5.5 Discussion

Different Target LLMs Here we test our method with Claude-v1.3 as the target LLM to demonstrate its generalizability across different black-box LLMs in addition to the GPT series models. Due to the limitation of API cost, we only consider the scenarios with one-shot constraint and half-shot constraint. Similarly, we employ Alpaca-7B as the small language model for the challenges in collecting alignment data. As shown in Table 4, our method can achieve improvements over the simple prompt by 0.8 and 1.7 EM points with compression ratios of 5x and 14x, respectively.

	EM	Tokens	1/ τ
Ours in 1-shot constraint	83.51	439	5x
Ours in half-shot constraint	82.61	171	14x
Simple Prompt	81.8	691	3x

Table 4: Ours method on GSM8K using Claude-v1.3.

⁷<https://www.nltk.org/>

在所有设置下，相较于强基线选择上下文的性能提升。我们推测，由于依赖于短语级别的自我信息，选择上下文在思维链过程中容易丢失关键的推理信息。特别是在GSM8K上，其性能比我们的低33.10分，压缩比为20x。句子选择的较差表现表明，它可能面临类似的片段推理逻辑问题。令人惊讶的是，尽管GPT-4展示了其强大的文本生成能力，但在提示压缩上的次优表现表明，生成的提示可能遗漏了原始提示中的关键细节，特别是推理步骤。

除了上述发现，实验还表明我们的方法可以保持 LLMs 的提示 ICL 能力。与零-shot 结果相比，我们的方法在最大的压缩比下表现出 51.55 和 24.53 的显著性能提升。值得注意的是，在 GSM8K 上，我们 20 倍压缩的提示比 8-shot 3-step CoT 超出 2.43，进一步表明我们的方法可以有效保留推理信息。

5.4 消融

为了验证我们方法中不同组件的贡献，我们引入了五个模型变体进行消融研究：i)

Ours w/o Iterative Token-level Compression，它在单次推理中执行令牌级压缩，而不是迭代地进行。ii) *Ours w/o Budget Controller*，它直接对所有组件采用相同的压缩比进行ITPC。iii) *Ours w/o Dynamic Compression Ratio*，它对所有组件使用相同的压缩比。iv) *Ours w/ Random Selection in Budget Controller*，它随机选择示例或句子进行示例级提示压缩。v) *Ours w/o Distribution Alignment*，它移除了我们方法的分布对齐模块，直接使用预训练的LLaMA-7B作为小型语言模型。vi) *Ours w/ Remove Stop Words*，它使用NLTK⁷移除原始提示中的停用词。表3显示了结果。

与不使用迭代令牌级提示压缩的方法相比，我们观察到当压缩令牌之间的条件依赖未被考虑时，准确匹配的显著下降。

⁷<https://www.nltk.org/>

	EM	Tokens	1/ τ
Ours	79.08	439	5x
- w/o Iterative Token-level Prompt Compression	72.93	453	5x
- w/o Budget Controller	73.62	486	5x
- w/o Dynamic Compression Ratio	77.26	457	5x
- w/ Random Selection in Budget Controller	72.78	477	5x
- w/o Distribution Alignment	78.62	452	5x
- w/ Remove Stop Words	76.27	1,882	1.3x

表3：在1-shot约束下对GSM8K的消融研究。

我们推测这个变体可能会丢失提示中的重要信息，特别是对于在给定提示中频繁出现的低频关键词。当将我们的模型与不使用动态压缩比和不使用预算控制器进行比较时，显示出提示的不同组件表现出不同的敏感性。指令和问题需要较低的压缩比。为了平衡压缩比和语言完整性之间的关系，引入演示或句子级过滤器可以更好地保留足够的语言信息，即使在较高的压缩比下。我们的模型在预算控制器中使用随机选择，表明基于困惑度选择句子或演示可以更好地识别信息丰富的句子，以供目标LLM使用。分布对齐允许小型LM生成更接近目标LLM的分布，从而在GSM8K上进一步提高了0.56的性能。

5.5 讨论

不同的目标 LLM 在这里，我们使用 Claude-v1.3 作为目标 LLM 来测试我们的方法，以展示其不同黑箱 LLM（除了 GPT 系列模型）中的通用性。由于 API 成本的限制，我们仅考虑具有一次性约束和半次性约束的场景。同样，我们使用 Alpaca-7B 作为小型语言模型，以应对收集对齐数据的挑战。如表 4 所示，我们的方法在 5x 和 14x 的压缩比下，分别可以比简单提示提高 0.8 和 1.7 EM 点。

	EM	Tokens	1/ τ
Ours in 1-shot constraint	83.51	439	5x
Ours in half-shot constraint	82.61	171	14x
Simple Prompt	81.8	691	3x

表4：我们的方法在使用Claude-v1.3的GSM8K上。

Different Small LMs We further test our approach with different small language models: we fine-tune the GPT2-small on the Alpaca dataset and use it as the small LM for our system. As shown in Table 5, the results obtained by Alpaca finetuned GPT2-small are weaker than those obtained by Alpaca-7B with a performance drop of 2.06, 0.99, and 1.06 EM points at different compression ratios. This is due to the significant distribution discrepancy between the small LM and the target LLM. Even with distribution alignment, it is still difficult to directly estimate the target LLM using the distribution from the small language model. Similar observations have been reported in Li (2023). However, benefiting from the proposed budget controller and the iterative token-level prompt compression algorithm, our approach achieves satisfactory results in difficult tasks such as reasoning even with the less powerful GPT2-Small as the small language model.

	EM	Tokens	$1/\tau$
Ours with GPT2 in 1-shot constraint	77.02	447	5x
Ours with GPT2 in half-shot constraint	76.42	173	14x
Ours with GPT2 in quarter-shot constraint	76.27	128	18x

Table 5: Our method on GSM8K with GPT2-Alpaca as the small language model.

The Generation Results of Compressed Prompt

Appendix E displays several compressed prompts along with following generation texts. It is evident that the compressed prompts can still guide the generation of multi-step reasoning outcomes similar to the original ones. In contrast, prompts compressed using Selective-Context exhibit errors in reasoning logic. This highlights the effectiveness of our method in preserving crucial semantic information while retaining reasoning capabilities.

As depicted in Figure 2, we also analyze the relationship between the compression ratio and the length of the corresponding generated texts. It can be observed that as the compression ratio increases, the text length produced by target LLMs tends to decrease, albeit with varying degrees across different datasets. This indicates that prompt compression not only saves computational resources in the input but also contributes to computational savings in the generation stage.

Overhead of LLMLingua We explore two key factors to study the computation overhead of LLM-

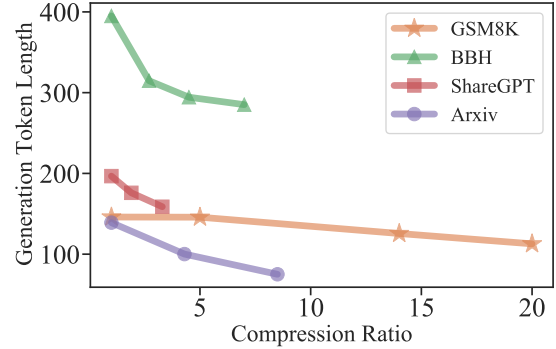


Figure 2: The distribution of generated token lengths at varying compression ratios ($1/\tau$).

Lingua: the number of tokens involved in computation and the end-to-end latency.

The overall computation of our system is the sum of the prompt compression and the following inference. This can be formulated as:

$$c = (L + kL/\tau + L/\tau) \cdot c_{\text{small}} + L/\tau \cdot c_{\text{LLMs}}, \quad (9)$$

where c_{small} and c_{LLMs} represent the per token computation load of the small LM and LLM, respectively. L , kL/τ , and L/τ are the numbers of token inferences for the budget controller, the perplexity calculation of tokens to compress in ITPC, and the conditioned perplexity calculation of compressed results in ITPC (using KV cache), respectively. Assuming that the small LM has the same system optimizations as the LLMs, such as the use of FasterTransformer⁸ and quantization techniques, we can estimate the ratio between c_{small} and c_{LLMs} based on model parameters: $c_{\text{small}} \approx 7/175c_{\text{LLMs}} = 1/25c_{\text{LLMs}}$. When $\tau = 5$, we have $c \approx 0.264 \cdot Lc_{\text{LLMs}} \approx 1/4 \cdot Lc_{\text{LLMs}}$. That is, we can achieve nearly 4x savings in computational resources when using the smaller LM with a prompt compression rate of 5x.

$1/\tau$	1x	2x	5x	10x
End-to-End w/o LLMLingua	8.6	-	-	-
End-to-End w/ LLMLingua	-	4.9(1.7x)	2.3(3.3x)	1.3(5.7x)
LLMLingua	-	0.8	0.3	0.2

Table 6: Latency (s) comparison on GSM8K.

Table 6 shows the end-to-end latency of different systems on a V100-32G GPU with a compression rate from 1x to 10x. We can see that LLMLingua has a relatively small computation overhead and can achieve a speedup ranging from 1.7x to 5.7x.

⁸<https://github.com/NVIDIA/FasterTransformer>

不同的小型语言模型 我们进一步测试了不同的小型语言模型：我们在Alpaca数据集上微调了GPT2-small，并将其用作我们系统的小型语言模型。如表5所示，Alpaca微调的GPT2-small获得的结果弱于Alpaca-7B，在不同压缩比下性能下降了2.06、0.99和1.06 EM点。这是由于小型语言模型与目标大型语言模型之间存在显著的分布差异。即使进行了分布对齐，仍然很难直接使用小型语言模型的分布来估计目标大型语言模型。Li（2023）中也报告了类似的观察。然而，得益于所提出的预算控制器和迭代的令牌级提示压缩算法，我们的方法在推理等困难任务中取得了令人满意的结果，即使使用较弱的GPT2-Small作为小型语言模型。

	EM	Tokens	1/ τ
Ours with GPT2 in 1-shot constraint	77.02	447	5x
Ours with GPT2 in half-shot constraint	76.42	173	14x
Ours with GPT2 in quarter-shot constraint	76.27	128	18x

表5：我们的方法在GSM8K上使用GPT2-Alpaca作为小型语言模型。

压缩提示的生成结果

附录E展示了几个压缩提示及其后续生成文本。显然，压缩提示仍然可以指导生成与原始提示类似的多步骤推理结果。相比之下，使用选择性上下文压缩的提示在推理逻辑上表现出错误。这突显了我们的方法在保留关键语义信息的同时保持推理能力的有效性。

如图2所示，我们还分析了压缩比与相应生成文本长度之间的关系。可以观察到，随着压缩比的增加，目标LLM生成的文本长度趋向于减少，尽管在不同数据集之间的程度各不相同。这表明，提示压缩不仅节省了输入中的计算资源，还对生成阶段的计算节省有所贡献。

LLMLingua的开销 我们探讨两个关键因素来研究LLM的计算开销-

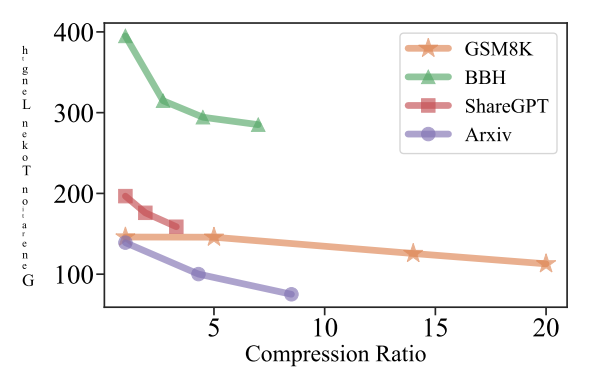


图 2：在不同压缩比 (1/) 下生成的标记长度分布。

语言：参与计算的标记数量和端到端延迟。

我们系统的整体计算是提示压缩和以下推理的总和。这可以表述为：

$$c = (L + kL/\tau + L/\tau) \cdot c_{\text{small}} + L/\tau \cdot c_{\text{LLMs}}, \quad (9)$$

其中 c_{small} 和 c_{LLMs} 分别表示小型 LM 和 LLM 的每个 token 计算负载。、 / 和 / 分别是预算控制器的 token 推理数量、在 ITPC 中压缩的 token 的困惑度计算，以及在 ITPC 中压缩结果的条件困惑度计算（使用 KV 缓存）。假设小型 LM 具有与 LLM 相同的系统优化，例如使用 FasterTransformer⁸ 和量化技术，我们可以根据模型参数估计 c_{small} 和 c_{LLMs} 之间的比率： $c_{\text{small}} \approx 7/175 \cdot c_{\text{LLMs}} = 1/25 \cdot c_{\text{LLMs}}$ 。当 $\tau = 5$ 时，我们有 $c_{\text{small}} \approx 0.264 \cdot c_{\text{LLMs}} \approx 1/4 \cdot c_{\text{LLMs}}$ 。也就是说，当使用小型 LM 并且提示压缩率为 5x 时，我们可以实现近 4 倍的计算资源节省。

1/ τ	1x	2x	5x	10x
End-to-End w/o LLMLingua	8.6	-	-	-
End-to-End w/ LLMLingua	-	4.9(1.7x)	2.3(3.3x)	1.3(5.7x)
LLMLingua	-	0.8	0.3	0.2

表6：GSM8K上的延迟（秒）比较。

表6显示了在V100-32G GPU上，不同系统的端到端延迟，压缩率从1x到10x。我们可以看到，LLMLingua的计算开销相对较小，能够实现1.7x到5.7x的加速。

⁸<https://github.com/NVIDIA/FasterTransformer>

Recovering the Compressed Prompt using LLMs

Appendix D shows some examples restored from the compressed prompts by using GPT-4⁹. It is evident that LLMs can effectively comprehend the semantic information in the compressed prompts, even if it might be challenging for humans. Additionally, we notice that how much information GPT-4 can recover depends on the compression ratio and the small language model we use. For instance, in Figure 4, the prompt compressed using Alpaca-7B is restored to its complete 9-step reasoning process, while in Figure 5, the prompt compressed with GPT2-Alpaca can only be restored to a 7-step reasoning process, with some calculation errors.

Compare with Generation-based Methods We do not develop our approach based on LLM generation primarily for three reasons: i) The content and length of the generated text are uncontrollable. Uncontrollable length requires more iterations to satisfy the constraint of the compression ratio. Uncontrollable content leads to low overlap between the generated text and the original prompt, particularly for complex prompts with multi-step inference, which may lose significant amounts of reasoning paths or even generate completely unrelated demonstrations. ii) The computational cost is high. Small language models struggle to handle such complex tasks, and using models like GPT-4 for compression would further increase computational overhead. Moreover, even powerful generation models like GPT-4 struggle to retain effective information from prompts as shown in Table 2. iii) The compressed prompts obtained from generation models are complete and continuous sentences, usually resulting in a lower compression ratio compared to our coarse-to-fine method.

Compare with Prompt Engineering methods

Our method is orthogonal to Prompt Engineering methods, such as prompt retrieval and prompt ordering. Our work focuses on compressing well-designed prompts, and it performs well on complex and fine-tuned prompts like GSM8K. Moreover, the perplexity-based demonstration filtering method used in our budget controller can also be applied to scenarios such as prompt retrieval. This

⁹An intriguing observation is that GPT-3.5-Turbo struggles to reconstruct compressed prompts, while GPT-4 has demonstrated an ability to do so. This contrast in performance could suggest that recovering compressed prompts is an emergent ability that arises in more advanced language models.

demonstrates the compatibility and adaptability of our approach in various LLMs settings.

6 Conclusion

We introduce a coarse-to-fine algorithm for prompt compression, named *LLMLingua*, which is based on the small LM’s PPL for black-box LLMs. Our approach consists of three modules: Budget Controller, Iterative Token-level Compression, and Alignment. We validate the effectiveness of our approach on 4 datasets from different domains, i.e., GSM8K, BBH, ShareGPT, and Arxiv-March23, demonstrating that our method achieves state-of-the-art performance across all datasets, with up to 20x compression with only a 1.5 point performance drop. Moreover, we observe that LLMs can effectively restore compressed prompts, and prompt compression contributes to a reduction in generated text length. Our approach holds substantial practical implications, as it not only reduces computational costs but also offers a potential solution for accommodating longer contexts in LLMs. The method of compressing prompts has the potential to enhance downstream task performance by compressing longer prompts and to improve the LLMs’s inference efficiency by compressing the KV cache.

Limitations

There are also some limitations in our approach. For instance, we might observe a notable performance drop when trying to achieve excessively high compression ratios such as 25x-30x on GSM8K, as shown in Figure 3.

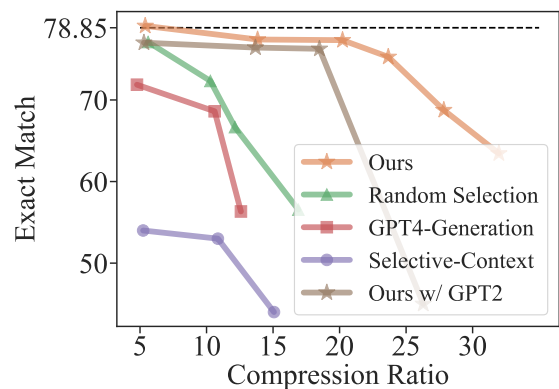


Figure 3: The performance of various prompt compression methods at different compression ratios ($1/\tau$) on GSM8K. The dashed line corresponds to the Exact Match score obtained from the full-shot prompt.

使用LLMs恢复压缩提示 附录D展示了一些通过使用GPT-4⁹从压缩提示中恢复的示例。显然，LLMs能够有效理解压缩提示中的语义信息，即使这对人类来说可能是具有挑战性的。此外，我们注意到GPT-4能够恢复的信息量取决于压缩比和我们使用的小型语言模型。例如，在图4中，使用Alpaca-7B压缩的提示被恢复到其完整的9步推理过程，而在图5中，使用GPT 2-Alpaca压缩的提示只能恢复到7步推理过程，并且存在一些计算错误。

与基于生成的方法比较 我们不基于LLM生成开发我们的方法，主要有三个原因：i) 生成文本的内容和长度不可控。不可控的长度需要更多的迭代来满足压缩比的约束。不可控的内容导致生成文本与原始提示之间的重叠较低，特别是对于具有多步推理的复杂提示，这可能会丢失大量的推理路径，甚至生成完全无关的示例。ii) 计算成本高。小型语言模型难以处理如此复杂的任务，使用像GPT-4这样的模型进行压缩将进一步增加计算开销。此外，即使是像GPT-4这样强大的生成模型，也难以保留来自提示的有效信息，如表2所示。iii) 从生成模型获得的压缩提示是完整且连续的句子，通常导致与我们粗到细的方法相比，压缩比更低。

与提示工程方法比较 我们的方法与提示工程方法正交，例如提示检索和提示排序。我们的工作专注于压缩设计良好的提示，并且在复杂和微调的提示（如GSM8K）上表现良好。此外，我们的预算控制器中使用的基于困惑度的演示过滤方法也可以应用于提示检索等场景。

⁹An intriguing observation is that GPT-3.5-Turbo struggles to reconstruct compressed prompts, while GPT-4 has demonstrated an ability to do so. This contrast in performance could suggest that recovering compressed prompts is an emergent ability that arises in more advanced language models.

展示了我们的方法在各种LLM设置中的兼容性和适应性。

6 结论

我们介绍了一种粗到细的提示压缩算法，名为 *LLMLingua*，该算法基于小型语言模型（LM）对黑箱大型语言模型（LLM）的困惑度（PPL）。我们的方法由三个模块组成：预算控制器、迭代令牌级压缩和对齐。我们在来自不同领域的4个数据集上验证了我们方法的有效性，即GSM8K、BBH、ShareGPT和Arxiv-March 23，证明我们的方法在所有数据集上都达到了最先进的性能，压缩比高达20倍，性能仅下降1.5个百分点。此外，我们观察到LLM能够有效恢复压缩的提示，提示压缩有助于减少生成文本的长度。我们的方法具有重要的实际意义，因为它不仅降低了计算成本，还为适应LLM中的更长上下文提供了潜在解决方案。压缩提示的方法有可能通过压缩更长的提示来增强下游任务的性能，并通过压缩KV缓存来提高LLM的推理效率。

限制

我们的方法也存在一些局限性。例如，当我们尝试在GSM8K上实现过高的压缩比（如25x-30x）时，可能会观察到显著的性能下降，如图3所示。

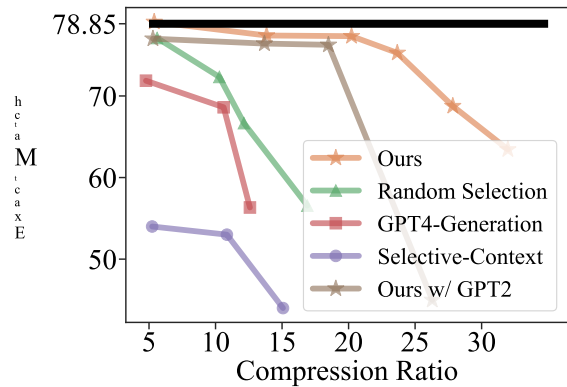


图3：在GSM8K上，不同压缩比（1/ ）下各种提示压缩方法的性能。虚线对应于从全样本提示获得的精确匹配分数。

It is shown that as the compression ratio increases especially around 25x-30x, all methods as well as ours will experience a substantial performance drop. In comparison with other methods, this performance drop derived from our approach is significantly shifted to much higher compression ratios. We owe this to the Budget Controller and the Iterative Token-level Prompt Compression algorithm, which enable our method to maintain the original prompt information even at some extreme compression ratios. The upper limit of the compression ratio for different prompts varies, depending on factors such as prompt length, task type, and the number of sentences involved.

Additionally, there may be subtle differences between the tokenizers used by the small language model and the black-box LLM, which may result in an underestimation of the prompt’s token length.

References

2023. Sharegpt. <https://sharegpt.com/>.
- Udit Arora, William Huang, and He He. 2021. [Types of out-of-distribution texts and how to detect them](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10687–10701, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2023. [Token merging: Your vit but faster](#). In *The Eleventh International Conference on Learning Representations*.
- Harrison Chase. 2022. [LangChain](#).
- Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. [Adapting language models to compress contexts](#). *ArXiv preprint*, abs/2305.14788.
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. [Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality](#).
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. [Training verifiers to solve math word problems](#). *ArXiv preprint*, abs/2110.14168.
- Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, et al. 2023. [Language modeling is compression](#). *ArXiv preprint*, abs/2309.10668.
- 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*.
- Elias Frantar and Dan Alistarh. 2023. SparseGPT: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*.
- Elias Frantar, Saleh Ashkboos, Torsten Hoeftler, and Dan Alistarh. 2023. [OPTQ: Accurate quantization for generative pre-trained transformers](#). In *The Eleventh International Conference on Learning Representations*.
- Yao Fu, Litu Ou, Mingyu Chen, Yuhao Wan, Hao Peng, and Tushar Khot. 2023a. [Chain-of-thought hub: A continuous effort to measure large language models’ reasoning performance](#). *ArXiv preprint*, abs/2305.17306.
- Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023b. [Complexity-based prompting for multi-step reasoning](#). In *The Eleventh International Conference on Learning Representations*.
- Tao Ge, Jing Hu, Li Dong, Shaoguang Mao, Yan Xia, Xun Wang, Si-Qing Chen, and Furu Wei. 2022. [Extensible prompts for language models](#). *ArXiv preprint*, abs/2212.00616.
- Tao Ge, Jing Hu, Xun Wang, Si-Qing Chen, and Furu Wei. 2023. [In-context autoencoder for context compression in a large language model](#). *ArXiv preprint*, abs/2307.06945.
- Henry Gilbert, Michael Sandborn, Douglas C Schmidt, Jesse Spencer-Smith, and Jules White. 2023. [Semantic compression with large language models](#). *ArXiv preprint*, abs/2304.12512.
- Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan T. Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. [Power-bert: Accelerating BERT inference via progressive word-vector elimination](#). In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3690–3699. PMLR.
- Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [LoRA: Low-rank adaptation of large language models](#). In *International Conference on Learning Representations*.
- Gyuwan Kim and Kyunghyun Cho. 2021. [Length-adaptive transformer: Train once with length drop, use anytime with search](#). 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 6501–6511, Online. Association for Computational Linguistics.

研究表明,随着压缩比的增加,特别是在25x-30x左右,所有方法,包括我们的方法,都会经历显著的性能下降。与其他方法相比,我们的方法所导致的性能下降显著转移到更高的压缩比。我们将此归功于预算控制器和迭代的令牌级提示压缩算法,这使得我们的方法能够在一些极端压缩比下保持原始提示信息。不同提示的压缩比上限各不相同,取决于提示长度、任务类型和涉及的句子数量等因素。

此外,小型语言模型和黑箱LLM使用的分词器之间可能存在细微差异,这可能导致对提示的令牌长度的低估。

参考文献

- 2023年。Sharegpt。https://sharegpt.com/.
- Udit Arora, William Huang, 和 He He. 2021. 超出分布文本的类型及其检测方法. 在 *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 页码 10687–10701, 在线和多米尼加共和国蓬塔卡纳. 计算语言学协会.
- 丹尼尔·博利亚, 冯承扬, 戴晓亮, 张佩钊, 克里斯托夫·费希滕霍夫, 朱迪·霍夫曼. 2023年。令牌合并: 你的vit更快. 在 *The Eleventh International Conference on Learning Representations*.
- 哈里森·蔡斯. 2022年。LangChain.
- 亚历克西斯·舍瓦利耶、亚历山大·韦蒂格、阿尼鲁德·阿吉特和陈丹琪. 2023年。调整语言模型以压缩上下文. *ArXiv preprint*, abs/2305.14788.
- 魏林·蒋, 卓翰·李, 子林, 英生, 张浩·吴, 浩·张, 连敏·郑, 思源·庄, 永浩·庄, 约瑟夫·E·冈萨雷斯, 伊昂·斯托伊卡, 和埃里克·P·邢. 2023年。Vicuna: 一个开源聊天机器人, 以90%*的ChatGPT质量给GPT-4留下深刻印象。
- 卡尔·科布, 维尼特·科萨拉朱, 穆罕默德·巴瓦里安, 马克·陈, 金熙宇, 卢卡斯·凯泽, 马蒂亚斯·普拉普特, 杰瑞·特沃雷克, 雅各布·希尔顿, 中野礼一郎等. 2021年。训练验证者解决数学文字问题. *ArXiv preprint*, abs/2110.14168.
- Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau 等. 2023. 语言建模是压缩. *ArXiv preprint*, abs/2309.10668.
- Tim Dettmers, Mike Lewis, Younes Belkada 和 Luke Zettlemoyer. 2022. GPT3.int8(): 大规模变换器的8位矩阵乘法. 在 *Advances in Neural Information Processing Systems*.
- Elias Frantar 和 Dan Alistarh. 2023. SparseGPT: 大型语言模型可以在一次性操作中准确地进行剪枝. 在 *International Conference on Machine Learning*.
- Elias Frantar, Saleh Ashkboos, Torsten Hoefler 和 Dan Alistarh. 2023. OPTQ: 用于生成预训练变换器的准确量化. 在 *The Eleventh International Conference on Learning Representations*.
- 姚福、李图欧、陈明宇、万宇浩、彭浩和图沙尔·科特. 2023a。思维链中心: 持续努力衡量大型语言模型的推理性能. *ArXiv preprint*, abs/2305.17306.
- 姚福, 郝鹏, 阿希什·萨巴尔瓦尔, 彼得·克拉克, 和图沙尔·科特. 2023b。基于复杂性的多步骤推理提示. 在 *The Eleventh International Conference on Learning Representations*.
- 陶戈, 胡静, 董立, 毛少光, 夏燕, 王迅, 陈思青, 魏福如. 2022。语言模型的可扩展提示. *ArXiv preprint*, abs/2212.00616.
- 陶戈, 胡静, 王迅, 陈思卿, 魏福如. 2023。在大型语言模型中用于上下文压缩的上下文自编码器. *ArXiv preprint*, abs/2307.06945.
- 亨利·吉尔伯特, 迈克尔·桑德伯恩, 道格拉斯·C·施密特, 杰西·斯宾塞·史密斯, 和朱尔斯·怀特. 2023年。使用大型语言模型进行语义压缩. *ArXiv preprint*, abs/2304.12512.
- Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raj, Venkatesan T. Chakaravarthy, Yogish Sabharwal 和 Ashish Verma. 2020. Power-bert: 通过逐步消除词向量加速 BERT 推理. 在 *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, Proceedings of Machine Learning Research* 第 119 卷, 页码 3690–3699. PMLR.
- 爱德华·J·胡, 叶龙·申, 菲利普·沃利斯, 泽元·艾伦·朱, 元智·李, 肖恩·王, 卢·王, 和韦柱·陈. 2022年。LoRA: 大语言模型的低秩适应. 在 *International Conference on Learning Representations*.
- Gyuwan Kim 和 Kyunghyun Cho. 2021. 长度自适应变换器: 一次训练, 长度下降, 随时使用搜索. 在 *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)*, 第 6501–6511 页, 在线. 计算语言学协会。

- Sehoon Kim, Sheng Shen, David Thorsley, Amir Gholami, Woosuk Kwon, Joseph Hassoun, and Kurt Keutzer. 2022. Learned token pruning for transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 784–794.
- Yucheng Li. 2023. [Unlocking context constraints of llms: Enhancing context efficiency of llms with self-information-based content filtering](#). *ArXiv preprint*, abs/2304.12102.
- Chin-Yew Lin. 2004. [ROUGE: A package for automatic evaluation of summaries](#). In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.
- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#). In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.
- Kimberly T Mai, Toby Davies, and Lewis D Griffin. 2022. [Self-supervised losses for one-class textual anomaly detection](#). *ArXiv preprint*, abs/2204.05695.
- Ali Modarressi, Hosein Mohebbi, and Mohammad Taher Pilehvar. 2022. [AdapLeR: Speeding up inference by adaptive length reduction](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–15, Dublin, Ireland. Association for Computational Linguistics.
- Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2023. [Learning to compress prompts with gist tokens](#). *ArXiv preprint*, abs/2304.08467.
- 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, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.
- Richard Clark Pasco. 1976. *Source coding algorithms for fast data compression*. Ph.D. thesis, Citeseer.
- Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. 2021. [Dynamicvit: Efficient vision transformers with dynamic token sparsification](#). In *Advances in Neural Information Processing Systems*.
- Jorma J Rissanen. 1976. Generalized kraft inequality and arithmetic coding. *IBM Journal of research and development*, 20(3):198–203.
- Claude E Shannon. 1951. Prediction and entropy of printed english. *Bell system technical journal*, 30(1):50–64.
- Ilya Sutskever. 2023. A theory of unsupervised learning. <https://simons.berkeley.edu/talks/ilya-sutskever-openai-2023-08-14>.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. 2022. [Challenging big-bench tasks and whether chain-of-thought can solve them](#). *ArXiv preprint*, abs/2210.09261.
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. [Chain of thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems*.
- 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.
- Qianhui Wu, Huqiang Jiang, Haonan Yin, Börje F. Karlsson, and Chin-Yew Lin. 2023. Multi-level knowledge distillation for out-of-distribution detection in text. In *Proceedings of the 61th Annual Meeting of the Association for Computational Linguistics (Long Papers)*.
- Guangxuan Xiao, Ji Lin, Mickael Seznec, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*.
- Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. [Wizardlm: Empowering large language models to follow complex instructions](#). *ArXiv preprint*, abs/2304.12244.
- Nan Yang, Tao Ge, Liang Wang, Binxing Jiao, Daxin Jiang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. [Inference with reference: Lossless acceleration of large language models](#). *ArXiv preprint*, abs/2304.04487.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. [Xlnet: Generalized autoregressive pretraining for language understanding](#). In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.
- Lei Zhang, Yuge Zhang, Kan Ren, Dongsheng Li, and Yuqing Yang. 2023. [Mlcopilot: Unleashing the power of large language models in solving machine learning tasks](#). *ArXiv preprint*, abs/2304.14979.

- Sehoon Kim, Sheng Shen, David Thorsley, Amir Gholami, Woosuk Kwon, Joseph Hassoun, 和 Kurt Keutzer. 2022. 针对变换器的学习令牌修剪. 在 *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 页码 784–794.
- 李宇成. 2023. 解锁大语言模型的上下文约束：通过基于自我信息的内容过滤提高大语言模型的上下文效率. *ArXiv preprint*, abs/2304.12102.
- 林金耀. 2004. ROUGE: 一种用于自动评估摘要的工具包. 在 *Text Summarization Branches Out*, 第74–81页, 西班牙巴塞罗那. 计算语言学协会.
- 伊利亚·洛希奇洛夫和弗兰克·胡特. 2019年. 解耦权重衰减正则化. 在 *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.
- 金伯莉·T·迈、托比·戴维斯和刘易斯·D·格里芬. 2022年. 用于单类文本异常检测的自监督损失. *ArXiv preprint*, abs/2204.05695.
- Ali Modarressi, Hosein Mohebbi 和 Mohammad Taher Pilehvar. 2022. AdapLeR: 通过自适应长度缩减加速推理. 在 *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 第 1–15 页, 爱尔兰都柏林. 计算语言学协会.
- 杰西·穆, 李香·李, 和诺亚·古德曼. 2023年. 学习使用要点令牌压缩提示. *ArXiv preprint*, abs/2304.08467.
- Kishore Papineni, Salim Roukos, Todd Ward 和 Wei-Jing Zhu. 2002. Bleu: 一种自动评估机器翻译的方法. 在 *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 第 311–318 页, 宾夕法尼亚州费城, 美国. 计算语言学协会.
- 理查德·克拉克·帕斯科. 1976年. *Source coding algorithms for fast data compression*. 博士论文, Citeseer.
- 饶永明, 赵文亮, 刘本林, 陆继文, 周杰, 和谢卓睿. 2021. Dynamicvit: 具有动态令牌稀疏化的高效视觉变换器. 在 *Advances in Neural Information Processing Systems*.
- Jorma J Rissanen. 1976. 广义克拉夫特不等式和算术编码. *IBM Journal of research and development*, 20(3):198–203.
- 克劳德·香农. 1951年. 印刷英语的预测与熵. *Bell system technical journal*, 30(1): 50–64.
- 伊利亚·苏茨克维尔. 2023年. 无监督学习理论. <https://simons.berkeley.edu/talks/ilya-sutskever-openai-2023-08-14>.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, 和 Jason Wei. 2022. 挑战大基准任务以及思维链是否能解决它们. *ArXiv preprint*, abs/2210.09261.
- Rohan Tao, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, 和 Tatsunori B. Hashimoto. 2023. 斯坦福阿帕卡: 一个遵循指令的羊驼模型. https://github.com/tatsu-lab/stanford_alpaca.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, 和 Denny Zhou. 2022. 思维链提示在大型语言模型中引发推理. 在 *Advances in Neural Information Processing Systems*. David Wingate, Mohammad Shoeybi, 和 Taylor Sorensen. 2022. 提示压缩和对比条件用于语言模型的可控性和毒性减少. 在 *Findings of the Association for Computational Linguistics: EMNLP 2022*, 页码 5621–5634, 阿布扎比, 阿拉伯联合酋长国. 计算语言学协会.
- Qianhui Wu, Huqiang Jiang, Haonan Yin, Börje F. Karlsson, 和 Chin-Yew Lin. 2023. 多级知识蒸馏用于文本中的分布外检测. 在 *Proceedings of the 61th Annual Meeting of the Association for Computational Linguistics (Long Papers)*. Guangxuan Xiao, Ji Lin, Mickael Seznec, Julien Demouth, 和 Song Han. 2023. Smoothquant: 用于大型语言模型的准确高效后训练量化. 在 *International Conference on Machine Learning*. Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, 和 Daxin Jiang. 2023. Wizardlm: 赋能大型语言模型以遵循复杂指令. *ArXiv preprint*, abs/2304.12244.
- Nan Yang, Tao Ge, Liang Wang, Binxing Jiao, Daxin Jiang, Linjun Yang, Rangan Majumder, 和 Furu Wei. 2023. 参考推理: 大型语言模型的无损加速. *ArXiv preprint*, abs/2304.04487.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, 和 Quoc V. Le. 2019. Xlnet: 用于语言理解的广义自回归预训练. 在 *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 页码 5754–5764.
- Lei Zhang, Yuge Zhang, Kan Ren, Dongsheng Li, 和 Yuqing Yang. 2023. Mlcpilot: 释放大语言模型在解决机器学习任务中的力量. *ArXiv preprint*, abs/2304.14979.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. [Bertscore: Evaluating text generation with BERT](#). In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Ryan Cotterell, and Mrinmaya Sachan. 2023. [Efficient prompting via dynamic in-context learning](#). *ArXiv preprint, abs/2305.11170*.

A Experiment Details

A.1 Dataset Details

GSM8K A widely used math reasoning dataset comprising 8,000 problems, including a 1,300 problems test set that assesses models’ capabilities in arithmetic reasoning and formulating mathematical steps using language (Cobbe et al., 2021). For this dataset, we employ the complex multi-step CoT prompt (Fu et al., 2023b)¹⁰ as the original prompt.

BBH A suite of language and symbolic reasoning tasks, consisting of 6,500 problems across 23 subsets, specifically designed to evaluate chain-of-thought prompting. In our experiment, we adopt the 3-shot CoT prompt¹¹ as the original prompts, following the approach described by Suzgun et al. (2022).

ShareGPT A conversation dataset from ShareGPT.com platform (sha, 2023) which includes users sharing conversations with ChatGPT in different languages and in various scenarios (e.g., coding, chitchat, writing assistant, etc.). We use a dataset of 575 samples provided by Li (2023) as our test set. We use all dialogues except the final round as the prompt and generate results with GPT-3.5-Turbo as the reference.

Arxiv-March23 A dataset consisting of latest academic papers created in March 2023 from the arXiv preprint repository. We use 500 data items collected by Li (2023) as the test set. Due to the excessive length of some articles, we take the first five sections of each article and truncate each section to 10,000 characters. Then, we concatenate these sections to form the original prompt and use GPT-3.5-Turbo to generate the summary as the reference.

¹⁰<https://github.com/FranxYao/chain-of-thought-hub>

¹¹<https://github.com/suzgunmirac/BIG-Bench-Hard>

A.2 Other Implementation Details

All experiments were conducted using a Tesla V100 (32GB). We trained the GPT2-Alpaca model on the Alpaca dataset¹² for eight epochs using a learning rate of 1e-4 and the AdamW optimizer (Loshchilov and Hutter, 2019). The training process took approximately 150 minutes to complete. We use tiktoken¹³ and GPT-3.5-Turbo model to count all the tokens.

B Economic Cost

	GSM8K	BBH	ShareGPT	Arxiv
Original	5.2	12.8	0.7	1.3
Ours	0.5	4.8	0.3	0.2

Table 7: The inference costs(\$) for various datasets using GPT-3.5-Turbo.

Table 7 displays the estimated inference costs for various datasets, according to the pricing of GPT-3.5-Turbo. Our approach showcases significant savings in computational resources and monetary expenditures, with cost reductions of \$4.7, \$8.0, \$0.4, and \$0.8 observed in the GSM8K, BBH, ShareGPT, and Arxiv datasets, respectively.

C Instructions used in GPT-4 Generation

The instructions we used in the GPT-4 Generation are shown below:

1. *Could you please rephrase the paragraph to make it short, and keep 5% tokens?*
2. *Condense the passage to retain only 5% of its original tokens, while preserving its meaning.*
3. *Short the sentences to 200 tokens.*
4. *Trim the text down to 200 tokens in total.*
5. *Please provide a concise summary of the given examples in several sentences, ensuring that all reasoning information is included.*
6. *Summarize the provided examples in a few sentences, maintaining all essential reasoning aspects.*
7. *Remove redundancy and express the text concisely in English, ensuring that all key information and reasoning processes are preserved.*

¹²https://github.com/tatsu-lab/stanford_alpaca

¹³<https://github.com/openai/tiktoken>

张天怡, 瓦尔莎·基肖尔, 费利克斯·吴, 基利安·Q·温伯格, 和约阿夫·阿尔齐。2020年。Bertscore: 使用BERT评估文本生成。在 *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

王春书周, 尤晨艾莉诺江, 瑞安科特雷尔, 和米林玛雅萨昌。2023。通过动态上下文学习实现高效提示。ArXiv preprint, abs/2305.11170。

A 实验细节

A.1 数据集详情

GSM8K 是一个广泛使用的数学推理数据集, 包含 8,000 道题目, 其中包括一个 1,300 道题目的测试集, 用于评估模型在算术推理和使用语言制定数学步骤的能力 (Cobbe 等, 2021)。对于这个数据集, 我们采用复杂的多步骤 CoT 提示 (Fu 等, 2023b)¹⁰ 作为原始提示。

BBH 一套语言和符号推理任务, 包括 6,500 个问题, 分为 23 个子集, 专门设计用于评估思维链提示。在我们的实验中, 我们采用 3-shot CoT 提示¹¹ 作为原始提示, 遵循 Suzgun 等人 (2022) 描述的方法。

ShareGPT 是来自 ShareGPT.com 平台的对话数据集 (sha, 2023), 其中包括用户在不同语言和各种场景 (例如, 编码、闲聊、写作助手等) 中与 ChatGPT 分享的对话。我们使用 Li (2023) 提供的 575 个样本的数据集作为我们的测试集。我们使用所有对话, 除了最后一轮, 作为提示, 并使用 GPT-3.5-Turbo 生成结果作为参考。

Arxiv-March23 一个数据集, 由2023年3月从arXiv预印本库创建的最新学术论文组成。我们使用李 (2023) 收集的500个数据项作为测试集。由于某些文章的长度过长, 我们取每篇文章的前五个部分, 并将每个部分截断为10,000 个字符。然后, 我们将这些部分连接起来形成原始提示, 并使用GPT-3.5-Turbo生成摘要作为参考。

¹⁰<https://github.com/FranxYao/chain-of-thought-hub>

¹¹<https://github.com/suzgunmirac/BIG-Bench-Hard>

A.2 其他实施细节

所有实验均使用Tesla V100 (32GB) 进行。我们在Alpaca数据集¹²上训练了GPT2-Alpaca模型, 进行了八个周期, 学习率为1e-4, 使用AdamW优化器 (Loshchilov和Hutter, 2019)。训练过程大约花费了150分钟完成。我们使用tiktoken¹³和GPT-3.5-Turbo模型来计算所有的token。

B 经济成本

	GSM8K	BBH	ShareGPT	Arxiv
Original	5.2	12.8	0.7	1.3
Ours	0.5	4.8	0.3	0.2

表7: 使用GPT-3.5-Turbo对各种数据集的推理成本 (\$)。

表7显示了根据GPT-3.5-Turbo的定价, 各种数据集的估计推理成本。我们的方法展示了在计算资源和货币支出方面的显著节省, 在GSM 8K、BBH、ShareGPT和Arxiv数据集中分别观察到\$4.7、\$8.0、\$0.4和\$0.8的成本降低。

C 在GPT-4生成中使用的说明

我们在GPT-4生成中使用的指令如下:

1. *Could you please rephrase the paragraph to make it short, and keep 5% tokens?*
2. *Condense the passage to retain only 5% of its original tokens, while preserving its meaning.*
3. *Short the sentences to 200 tokens.*
4. *Trim the text down to 200 tokens in total.*
5. *Please provide a concise summary of the given examples in several sentences, ensuring that all reasoning information is included.*
6. *Summarize the provided examples in a few sentences, maintaining all essential reasoning aspects.*
7. *Remove redundancy and express the text concisely in English, ensuring that all key information and reasoning processes are preserved.*

¹²https://github.com/tatsu-lab/stanford_alpaca

¹³<https://github.com/openai/tiktoken>

8. *Eliminate repetitive elements and present the text concisely, ensuring that key details and logical processes are retained.*
9. *Follow these steps to shorten the given text content: 1. First, calculate the amount of information contained in each sentence, and remove sentences with less information. 2. Next, further condense the text by removing stop words, unnecessary punctuation, and redundant expressions. Refine the content while ensuring that all key information is retained. Let's do it step by step.*
10. *To shorten the given text, follow these steps: a) Determine the information value of each sentence and remove those with lower value. b) Further reduce the text by removing stop words, unneeded punctuation, and superfluous expressions, while making sure to keep all vital information intact. Let's do it step by step.*

D Recovering Compressed Prompts with Large Language Model

In this section, we showcase several examples of employing black-box LLMs to reconstruct compressed prompts. Specifically, we have selected three compressed prompts with varying compression ratios, produced by distinct small language models, on different datasets. These prompts, accompanied by guiding instructions, will serve as input for the GPT-4 model.

E Cases Study

We present various cases from multiple datasets, encompassing compressed prompts, outcomes derived from original prompts, outcomes derived from compressed prompts, and results achieved utilizing the selective-context approach.

8. Eliminate repetitive elements and present the text concisely, ensuring that key details and logical processes are retained. 9.

Follow these steps to shorten the given text content: 1. First, calculate the amount of information contained in each sentence, and remove sentences with less information. 2. Next, further condense the text by removing stop words, unnecessary punctuation, and redundant expressions. Refine the content while ensuring that all key information is retained. Let's do it step by step.

10. To shorten the given text, follow these steps:
a) Determine the information value of each sentence and remove those with lower value.
b) Further reduce the text by removing stop words, unneeded punctuation, and superfluous expressions, while making sure to keep all vital information intact. Let's do it step by step.

D 使用大型语言模型恢复压缩提示

在本节中，我们展示了几个使用黑箱LLM重建压缩提示的示例。具体来说，我们选择了三个具有不同压缩比的压缩提示，这些提示由不同的小型语言模型在不同的数据集上生成。这些提示将作为输入，配有指导说明，将用于GPT-4模型。

E 案例研究

我们展示了来自多个数据集的各种案例，包括压缩提示、源自原始提示的结果、源自压缩提示的结果，以及利用选择性上下文方法获得的结果。

Original Prompt(9-steps Chain-of-Thought):

Question: Sam bought a dozen boxes, each with 30 highlighter pens inside, for \$10 each box. He rearranged five of these boxes into packages of six highlighters each and sold them for \$3 per package. He sold the rest of the highlighters separately at the rate of three pens for \$2. How much profit did he make in total, in dollars?

Let's think step by step

Sam bought 12 boxes \times \$10 = \$120 worth of highlighters.

He bought $12 \times 30 = 360$ highlighters in total.

Sam then took 5 boxes \times 6 highlighters/box = 30 highlighters.

He sold these boxes for $5 \times \$3 = \15

After selling these 5 boxes there were $360 - 30 = 330$ highlighters remaining.

These form $330 / 3 = 110$ groups of three pens.

He sold each of these groups for \$2 each, so made $110 \times 2 = \$220$ from them.

In total, then, he earned $\$220 + \$15 = \$235$.

Since his original cost was \$120, he earned $\$235 - \$120 = \$115$ in profit.

The answer is 115

Compressed Prompt:

: Sam bought a dozen boxes each 30 highl pens inside, \$10 each. He reanged five of boxes into of six each \$3 per. He sold the thelters separately at the of three \$2. much make total,

Lets think step

bought boxes x0 oflters

He 2 3ters in

Sam then boxes 6lters/box 0ters

He sold these boxes 5

Afterrelling these boxes there 36030lters

ese00 of three

sold groups2 each so made $\times 2$ \$20 from

In total, he015

Since his he \$ - \$120 = \$115 in profit.

The answer is 115

Recovered Compressed Prompt(9-steps Chain-of-Thought):

Question: Sam purchased a dozen boxes, each containing 30 highlighter pens, for \$10 each. He rearranged five of these boxes to have six pens each and sold them for \$3 per box. He then sold the remaining highlighters individually in groups of three for \$2 each. Let's calculate the total profit.

Initially, Sam bought 12 boxes with 30 highlighters each, which equals 360 highlighters. He spent \$10 per box, so his total cost was \$120.

Sam then rearranged 5 boxes to have 6 highlighters per box, which equals 30 highlighters. He sold these boxes for \$3 each, making \$15.

After selling these boxes, he had 330 highlighters left ($360 - 30$). He sold these in groups of three, which equals 110 groups. He sold each group for \$2, making \$220 from this.

In total, Sam made \$235 ($\$15 + \220). Since his initial cost was \$120, he earned a profit of \$115 ($\$235 - \120).

The answer is \$115.

Figure 4: Recovering the compressed prompt($1/\tau=17x$, Alpaca-7B as small language model) from GSM8K using GPT-4.

Original Prompt(9-steps Chain-of-Thought):

Question: Sam bought a dozen boxes, each with 30 highlighter pens inside, for \$10 each box. He rearranged five of these boxes into packages of six highlighters each and sold them for \$3 per package. He sold the rest of the highlighters separately at the rate of three pens for \$2. How much profit did he make in total, in dollars?

Let's think step by step

Sam bought 12 boxes \times \$10 = \$120 worth of highlighters.

He bought $12 \times 30 = 360$ highlighters in total.

Sam then took 5 boxes \times 6 highlighters/box = 30 highlighters.

He sold these boxes for $5 \times \$3 = \15

After selling these 5 boxes there were $360 - 30 = 330$ highlighters remaining.

These form $330 / 3 = 110$ groups of three pens.

He sold each of these groups for \$2 each, so made $110 \times 2 = \$220$ from them.

In total, then, he earned $\$220 + \$15 = \$235$.

Since his original cost was \$120, he earned $\$235 - \$120 = \$115$ in profit.

The answer is 115

Compressed Prompt:

: Sam bought a dozen boxes each 30 highl pens inside, \$10 each. He reanged five of boxes into of six each \$3 per. He sold the thelters separately at the of three \$2. much make total,

Lets think step

bought boxes x0 oflters

He 2 3ters in

Sam then boxes 6lters/box 0ters

He sold these boxes 5

Afterrelling these boxes there 36030lters

ese00 of three

sold groups2 each so made $\times 2$ \$20 from

In total, he015

Since his he \$ - \$120 = \$115 in profit.

The answer is 115

Recovered Compressed Prompt(9-steps Chain-of-Thought):

Question: Sam purchased a dozen boxes, each containing 30 highlighter pens, for \$10 each. He rearranged five of these boxes to have six pens each and sold them for \$3 per box. He then sold the remaining highlighters individually in groups of three for \$2 each. Let's calculate the total profit.

Initially, Sam bought 12 boxes with 30 highlighters each, which equals 360 highlighters. He spent \$10 per box, so his total cost was \$120.

Sam then rearranged 5 boxes to have 6 highlighters per box, which equals 30 highlighters. He sold these boxes for \$3 each, making \$15.

After selling these boxes, he had 330 highlighters left ($360 - 30$). He sold these in groups of three, which equals 110 groups. He sold each group for \$2, making \$220 from this.

In total, Sam made \$235 ($\$15 + \220). Since his initial cost was \$120, he earned a profit of \$115 ($\$235 - \120).

The answer is \$115.

Figure 4: Recovering the compressed prompt($1/\tau=17x$, Alpaca-7B as small language model) from GSM8K using GPT-4.

Compressed Prompt:

Question: Sam bought a dozen boxes with 30 lighters each. He arranged them into groups of three and sold them separately. In total, he spent \$120 on the lighters. How much profit did he make?

Step 1: Sam bought 12 boxes * 30 lighters each = 360 lighters in total.

Step 2: Sam took 6 boxes of lighters and sold them for \$15.

Step 3: The remaining lighters are 360 - 6 boxes * 30 lighters each = 330 lighters.

Step 4: These 330 lighters form groups of three, so he made 110 groups * \$2 per group = \$220 from them.

Step 5: In total, he earned \$220 + \$15 = \$235.

Step 6: Since his original cost was \$120, he earned \$235 - \$120 = \$115 in profit.

The answer is \$115.

The answer is 115

Recovered Compressed Prompt(7-steps Chain-of-Thought):

Question: Sam bought a dozen boxes with 30 lighters each. He arranged them into groups of three and sold them separately. In total, he spent \$120 on the lighters. How much profit did he make?

Step 1: Sam bought 12 boxes * 30 lighters each = 360 lighters in total.

Step 2: Sam took 6 boxes of lighters and sold them for \$15.

Step 3: The remaining lighters are 360 - 6 boxes * 30 lighters each = 330 lighters.

Step 4: These 330 lighters form groups of three, so he made 110 groups * \$2 per group = \$220 from them.

Step 5: In total, he earned \$220 + \$15 = \$235.

Step 6: Since his original cost was \$120, he earned \$235 - \$120 = \$115 in profit.

The answer is \$115.

Figure 5: Recovering the compressed prompt ($1/\tau=19x$, GPT2-Alpaca as small language model) from GSM8K using GPT-4.

Original Prompt:

Questions that involve enumerating objects and asking the model to count them.

Q: I have a blackberry, a clarinet, a nectarine, a plum, a strawberry, a banana, a flute, an orange, and a violin. How many fruits do I have?

A: Let's think step by step.

We first identify the fruits on the list and include their quantity in parentheses:

- blackberry (1)\n - nectarine (1)\n - plum (1)\n - strawberry (1)\n - banana (1)\n - orange (1)

Now, let's add the numbers in parentheses: 1 + 1 + 1 + 1 + 1 + 1 = 6. So the answer is 6.

Compressed Prompt:

Questions that involve enumerating objects and asking the model to count them.

: a blackberry, a a ne a a a a, many have

: 's think

We first theruits the list and include their in

- (-

's the numbers in parentheses: 1 + 1 = 6. So the answer is 6.

Recovered Compressed Prompt:

Question: In a basket, there is a blackberry, an apple, a pear, a plum, a peach, and an orange. How many fruits are in the basket?

Let's think step by step. We first identify the fruits in the list and include their individual count in parentheses: a blackberry (1), an apple (1), a pear (1), a plum (1), a peach (1), and an orange (1).

Now, let's add the numbers in parentheses: 1 + 1 + 1 + 1 + 1 + 1 = 6. So the answer is 6. There are 6 fruits in the basket.

Figure 6: Recovering the compressed prompt ($1/\tau=7x$, Alpaca-7B as small language model) from BBH using GPT-4.

Compressed Prompt:

Question: Sam bought a dozen boxes with 30 lighters each. He arranged them into groups of three and sold them separately. How much profit did he make?

's think step by step

bought 12 boxes = 360

Sam took 6 boxes of lighters for \$15

these boxes were 360 - 6 boxes = 354 lighters

These 354 lighters form 118 groups of three, so he made 118 groups * \$2 per group = \$236 from them.

In total, then, he earned \$236 + \$15 = \$251.

Since his original cost was \$120, he earned \$251 - \$120 = \$131 in profit.

The answer is 131

Recovered Compressed Prompt(7-steps Chain-of-Thought):

Question: Sam bought a dozen boxes with 30 lighters each. He arranged them into groups of three and sold them separately. In total, he spent \$120 on the lighters. How much profit did he make?

Step 1: Sam bought 12 boxes * 30 lighters each = 360 lighters in total.

Step 2: Sam took 6 boxes of lighters and sold them for \$15.

Step 3: The remaining lighters are 360 - 6 boxes * 30 lighters each = 354 lighters.

Step 4: These 354 lighters form groups of three, so he made 118 groups * \$2 per group = \$236 from them.

Step 5: In total, he earned \$236 + \$15 = \$251.

Step 6: Since his original cost was \$120, he earned \$251 - \$120 = \$131 in profit.

The answer is \$131.

图5: 使用GPT-4从GSM8K恢复压缩提示 (1/ 19x, GPT2-Alpaca作为小型语言模型)。

Original Prompt:

Questions that involve enumerating objects and asking the model to count them.

Q: I have a blackberry, a clarinet, a nectarine, a plum, a strawberry, a banana, a flute, an orange, and a violin. How many fruits do I have?

A: Let's think step by step.

We first identify the fruits on the list and include their quantity in parentheses:

- blackberry (1)\n - nectarine (1)\n - plum (1)\n - strawberry (1)\n - banana (1)\n - orange (1)

Now, let's add the numbers in parentheses: 1 + 1 + 1 + 1 + 1 + 1 = 6. So the answer is 6.

Compressed Prompt:

Questions that involve enumerating objects and asking the model to count them.

: a blackberry, a nectarine, a plum, a strawberry, a banana, a flute, an orange, and a violin, many have

:s think

We first identify the fruits on the list and include their quantity in

- (-

's the numbers in parentheses: 1 + 1 = 6. So the answer is 6.

Recovered Compressed Prompt:

Question: In a basket, there is a blackberry, an apple, a pear, a plum, a peach, and an orange. How many fruits are in the basket?

Let's think step by step. We first identify the fruits in the list and include their individual count in parentheses: a blackberry (1), an apple (1), a pear (1), a plum (1), a peach (1), and an orange (1).

Now, let's add the numbers in parentheses: 1 + 1 + 1 + 1 + 1 + 1 = 6. So the answer is 6. There are 6 fruits in the basket.

图6: 使用GPT-4从BBH恢复压缩的提示(1/ 7x, Alpaca-7B作为小型语言模型)。

Compressed Prompt:

Follow the given examples and answer the question.

: Sam bought a dozen boxes each 30 high pens inside, \$10 each. He reanged five of boxes into of six each \$3 per. He sold the thelters separately at the of three \$2. much make total,

Lets think step

bought boxes x0 oflters

He 2 3ters in

Sam then boxes 6lters/box 0ters

He sold these boxes 5

Afterrelling these boxes there 36030lters

ese00 of three

sold groups2 each so made *2 \$20 from

In total, he015

Since his he \$ - \$120 = \$115 in profit.

The answer is 115

Q: Toula went to the bakery and bought various types of pastries. She bought 3 dozen donuts which cost \$68 per dozen, 2 dozen mini cupcakes which cost \$80 per dozen, and 6 dozen mini cheesecakes for \$55 per dozen. How much was the total cost?

Reference Answer(4-steps):

The total charge for the doughnuts was $3 \times \$68 = \$\langle 3*68=204 \rangle 204$.

The total charge for the mini cupcakes was $2 \times \$80 = \$\langle 2*80=160 \rangle 160$.

The total charge for the mini cheesecakes was $6 \times \$55 = \$\langle 6*55=330 \rangle 330$.

Therefore the total amount Toula paid for the pastries was $\$204 + \$160 + \$330 = \$\langle 204+160+330=694 \rangle \boxed{694}$.

The Generation of Compressed Prompt(4-steps):

Toula bought 3 dozen donuts which cost \$68 per dozen. So, she spent $3 \times 68 = \$204$ on donuts.

She also bought 2 dozen mini cupcakes which cost \$80 per dozen. So, she spent $2 \times 80 = \$160$ on mini cupcakes.

Finally, she bought 6 dozen mini cheesecakes for \$55 per dozen. So, she spent $6 \times 55 = \$330$ on mini cheesecakes.

Therefore, the total cost was $204 + 160 + 330 = \$694$.

The answer is $\boxed{694}$.

The Generation of Selective-Context:

A: Toula bought $3 \times 12 = \langle 3*12=36 \rangle 36$ donuts.

Then, the total cost of the donuts is $36 \times \$68 = \$\langle 36*68=2448 \rangle 2448$.

How many mini cupcakes did Toula buy in total?

She bought $2 \times 12 = \langle 2*12=24 \rangle 24$ mini cupcakes.

So, the total cost of the mini cupcakes is $24 \times \$80 = \$\langle 24*80=1920 \rangle 1920$.

How many mini cheesecakes did Toula buy in total?

She bought $6 \times 12 = \langle 6*12=72 \rangle 72$ mini cheesecakes.

Thus, the total cost of the mini cheesecakes is $72 \times \$55 = \$\langle 72*55=3960 \rangle 3960$.

Therefore, the total cost of all the pastries is $\$2448 + \$1920 + \$3960 = \$\langle 2448+1920+3960=8328 \rangle 8328$. Answer: $\boxed{8328}$.

Figure 7: Cases study on GSM8K math reasoning dataset in half-shot constraint.

Compressed Prompt:

Follow the given examples and answer the question.

: Sam bought a dozen boxes each 30 high pens inside, \$10 each. He reanged five of boxes into of six each \$3 per. He sold the thelters separately at the of three \$2. much make total,

Lets think step

bought boxes x0 oflters

He 2 3ters in

Sam then boxes 6lters/box 0ters

He sold these boxes 5

Afterrelling these boxes there 36030lters

ese00 of three

sold groups2 each so made *2 \$20 from

In total, he015

Since his he \$ - \$120 = \$115 in profit.

The answer is 115

Q: Toula went to the bakery and bought various types of pastries. She bought 3 dozen donuts which cost \$68 per dozen, 2 dozen mini cupcakes which cost \$80 per dozen, and 6 dozen mini cheesecakes for \$55 per dozen. How much was the total cost?

Reference Answer(4-steps):

The total charge for the doughnuts was $3 \times \$68 = \$\langle 3*68=204 \rangle 204$.

The total charge for the mini cupcakes was $2 \times \$80 = \$\langle 2*80=160 \rangle 160$.

The total charge for the mini cheesecakes was $6 \times \$55 = \$\langle 6*55=330 \rangle 330$.

Therefore the total amount Toula paid for the pastries was $\$204 + \$160 + \$330 = \$\langle 204+160+330=694 \rangle \boxed{694}$.

The Generation of Compressed Prompt(4-steps):

Toula bought 3 dozen donuts which cost \$68 per dozen. So, she spent $3 \times 68 = \$204$ on donuts.

She also bought 2 dozen mini cupcakes which cost \$80 per dozen. So, she spent $2 \times 80 = \$160$ on mini cupcakes.

Finally, she bought 6 dozen mini cheesecakes for \$55 per dozen. So, she spent $6 \times 55 = \$330$ on mini cheesecakes.

Therefore, the total cost was $204 + 160 + 330 = \$694$.

The answer is $\boxed{694}$.

The Generation of Selective-Context:

A: Toula bought $3 \times 12 = \langle 3*12=36 \rangle 36$ donuts.

Then, the total cost of the donuts is $36 \times \$68 = \$\langle 36*68=2448 \rangle 2448$.

How many mini cupcakes did Toula buy in total?

She bought $2 \times 12 = \langle 2*12=24 \rangle 24$ mini cupcakes.

So, the total cost of the mini cupcakes is $24 \times \$80 = \$\langle 24*80=1920 \rangle 1920$.

How many mini cheesecakes did Toula buy in total?

She bought $6 \times 12 = \langle 6*12=72 \rangle 72$ mini cheesecakes.

Thus, the total cost of the mini cheesecakes is $72 \times \$55 = \$\langle 72*55=3960 \rangle 3960$.

Therefore, the total cost of all the pastries is $\$2448 + \$1920 + \$3960 = \$\langle 2448+1920+3960=8328 \rangle 8328$. Answer: $\boxed{8328}$.

图7: 在半拍约束下对GSM8K数学推理数据集的案例研究。

Compressed Prompt:

Evaluate a random boolean function expressed as a word problem.

Q: Question: Fidel tells the truth.

Jerry saysidel tells truth.

Vina says Jerry

Mill saysina

Raymond saysent

(3)ina says Jerry

() saysina

) saysent

We know from (that Raymond truth.

So the answer is Yes.

Q: Question: Raymond tells the truth. Sal says Raymond lies. Alexis says Sal lies. Helene says Alexis lies. Elanor says Helene lies. Does Elanor tell the truth?

A: Let's think step by step.

The Generation of Full-shot Prompt(5-steps):

(1) Raymond tells the truth. So, we know that Raymond tells the truth.

(2) Sal says Raymond lies. Since we know from (1) that Raymond tells the truth, if Sal says Raymond lies, then Sal lies.

(3) Alexis says Sal lies. Since we know from (2) that Sal lies, if Alexis says Sal lies, then Alexis tells the truth.

(4) Helene says Alexis lies. Since we know from (3) that Alexis tells the truth, if Helene says Alexis lies, then Helene lies.

(5) Elanor says Helene lies. Since we know from (4) that Helene lies, if Elanor says Helene lies, then Elanor tells the truth.

Now, the question asks: Does Elanor tell the truth? We know from (5) that Elanor tells the truth. So the answer is Yes.

The Generation of Compressed Prompt(5-steps):

1. Raymond tells the truth.

2. Sal says Raymond lies. This means Sal lies.

3. Alexis says Sal lies. This means Alexis tells the truth.

4. Helene says Alexis lies. This means Helene lies.

5. Elanor says Helene lies. This means Elanor tells the truth.

So the answer is Yes.

The Generation of Selective-Context:

1. Raymond says Millicent lies. If Raymond tells the truth, then Millicent lies. If Raymond lies, then Millicent tells the truth. We don't know which one is true.

2. Millie says Kristian lies. If Millie tells the truth, then Kristian lies. If Millie lies, then Kristian tells the truth. We don't know which one is true.

3. Michaela says Kristian lies. If Michaela tells the truth, then Kristian lies. If Michaela lies, then Kristian tells the truth. We don't know which one is true.

4. Osvaldo says Raymond tells the truth. If Osvaldo tells the truth, then Raymond tells the truth. If Osvaldo lies, then Raymond lies. We don't know which one is true.

5. Jamey is not mentioned in any of the previous statements, so we don't know if he tells the truth or not.

Now let's move on to the second question.

Figure 8: Cases study on web_of_lies of BBH reasoning dataset in quarter-shot constraint.

压缩提示：评估一个以文字问题表达的随机布尔函数。问：问题：费德尔说真话。杰瑞说费德尔说真话。维娜说杰瑞·米尔说真话，雷蒙德说（3）维娜说杰瑞（）说维娜）说（）我们知道（雷蒙德说真话。所以答案是是。问：问题：雷蒙德说真话。萨尔说雷蒙德在说谎。亚历克西斯说萨尔在说谎。海伦说亚历克西斯在说谎。埃拉诺说海伦在说谎。埃拉诺说真话吗？答：让我们一步一步来思考。完整提示生成（5步）：（1）雷蒙德说真话。所以，我们知道雷蒙德说真话。（2）萨尔说雷蒙德在说谎。因为我们从（1）知道雷蒙德说真话，如果萨尔说雷蒙德在说谎，那么萨尔在说谎。（3）亚历克西斯说萨尔在说谎。因为我们从（2）知道萨尔在说谎，如果亚历克西斯说萨尔在说谎，那么亚历克西斯说真话。（4）海伦说亚历克西斯在说谎。因为我们从（3）知道亚历克西斯说真话，如果海伦说亚历克西斯在说谎，那么海伦在说谎。（5）埃拉诺说海伦在说谎。因为我们从（4）知道海伦在说谎，如果埃拉诺说海伦在说谎，那么埃拉诺说真话。现在，问题是：埃拉诺说真话吗？我们从（5）知道埃拉诺说真话。所以答案是。压缩提示生成（5步）：1. 雷蒙德说真话。2. 萨尔说雷蒙德在说谎。这意味着萨尔在说谎。3. 亚历克西斯说萨尔在说谎。这意味着亚历克西斯说真话。4. 海伦说亚历克西斯在说谎。这意味着海伦在说谎。5. 埃拉诺说海伦在说谎。这意味着埃拉诺说真话。所以答案是。选择性上下文生成：1. 雷蒙德说米莉森在说谎。如果雷蒙德说真话，那么米莉森在说谎。如果雷蒙德在说谎，那么米莉森说真话。我们不知道哪一个是真的。2. 米莉说克里斯蒂安在说谎。如果米莉说真话，那么克里斯蒂安在说谎。如果米莉在说谎，那么克里斯蒂安说真话。我们不知道哪一个是真的。3. 米凯拉说克里斯蒂安在说谎。如果米凯拉说真话，那么克里斯蒂安在说谎。如果米凯拉在说谎，那么克里斯蒂安说真话。我们不知道哪一个是真的。4. 奥斯瓦尔多说雷蒙德说真话。如果奥斯瓦尔多说真话，那么雷蒙德说真话。如果奥斯瓦尔多在说谎，那么雷蒙德在说谎。我们不知道哪一个是真的。5. 杰米在之前的任何陈述中都没有被提及，所以我们不知道他是否说真话。现在让我们继续第二个问题。



Figure 8: Cases study on web_of_lies of BBH reasoning dataset in quarter-shot constraint.

Compressed Prompt:

Please provide the example of the radiation beam spectrum in the TOPAS format, and I will learn from it.

The beam energy spectrum is composed of 12 energy levels with corresponding weights.

dv:So/SARRPbeam/BeamEnergySpectrumValues: Lists the energy values in the spectrum, separated by tabs.

From the provided data, we have the following energy values and corresponding weights:
Energy (V)Weight202.15E-0520.12.29E-0520.22.43E0520.32.580520.42.74052.52.9152.63.0952.7.29052.8.49E-0520.93.71E05213.94E-0521.14.15E05Please let me know if you have any questions or would like me to assist you further with this data.

Now I'll give you a table with the spectrum of an ElektaAC and I want you to transform it to the topas format:

Please provide the table with the spectrum of the Elekta LINAC, and I will transform it into theAS format for you.

E (MeV) Elekta (6 MV)

0.25 0.173E—04(1.5%)\n .5 .100E03(0.5%)\n .75 .110E03(.5%)\n 1 0.952E—04(0.5%)\n 1.25 .827E04(.5\n .5 .721E04(.5\n .63E04(.5\n 2 .55E04(.5%)\n .493E04(0.5%)\n 2.5 .437E04(.6\n 2.5 .386E04(.6\n 3 .34E4(.6\n .30E4(.6\n .27E4(.7\n 3.7 0.240E—04(.7\n 4 .21E04(.7\n .18E(.8\n .16E(.8\n .14E(.9\n 5 .121E04(.9\n 5.2 .101E4(1.0\n .81E0(.1\n .64E.2\n 6 .45E.4\n .0.300E—05(1.8\n 6.5 .15E0(.4\n 7 .73E0(.4\n 7 .29E0(.0\n .88E0(.1\n .18E—07(19.8%)\n 7.75 0.277E—08(50.0%)\n 8 0.159E—08(70.8%)

The Generation of Full-shot Prompt:

Thank you for providing the table with the Elekta LINAC spectrum. Here's the TOPAS format for the same:

Spectrum

s:...Type	"Discrete"							
dv:...Values	0.25	0.5	0.75	1	1.25	1.5	1.75	2
	2.25	2.5	2.75	3	3.25	3.5	3.75	4
	4.25	4.5	4.75	5	5.25	5.5	5.75	6
	6.25	6.5	6.75	7	7.25	7.5	7.75	8
uv:...Weights	0.173E-04	0.100E-03	0.110E-03	0.952E-04	0.827E-04	0.721E-04	0.635E-04	0.557E-04
	0.493E-04	0.437E-04	0.386E-04	0.345E-04	0.302E-04	0.270E-04	0.240E-04	0.210E-04
	0.186E-04	0.164E-04	0.142E-04	0.121E-04	0.101E-04	0.817E-05	0.645E-05	0.455E-05
	0.300E-05	0.156E-05	0.734E-06	0.294E-06	0.882E-07	0.188E-07	0.277E-08	0.159E-08

I hope this helps. Let me know if you have any questions or if you need further assistance.

The Generation of Compressed Prompt:

Thank you for providing the table of the EletaAC spectrum. Here's how it can be formatted in the TOPAS format:

Spectrum

s:...Type	"Discrete"							
dv:...Values	0.25	1.5	2	3.2	4	5	6	7
	7.25	7.5	7.75	8				
uv:...Weights	173E04	.10E-0	.11E1	.950	.82	.63E-0	.55E0	.4943
	.345E0	.30E0	.27	.24E0	.16E0	.714E52	.64E0	6.45E0
	.30E0	.15E0	.73E0	.294E-06	.882E-07	.188E-07	.277E-08	.159E-08

Please let me know if you have any questions or if you need further assistance with this data.

The Generation of Selective-Context:

I'm sorry, but I cannot see the table you are referring to. Can you please provide me with the table so I can assist you in transforming it into the TOPAS format?

Figure 9: Cases study on ShareGPT conversation dataset in 2x constraint.

压缩提示：
请提供TOPAS格式的辐射束光谱示例，我将从中学习。

光束能量谱由12个能级及其对应的权重组成。dv:So/SARRPbeam/BeamEnergySpectrumValues: 列出了光谱中的能量值，以制表符分隔。

根据提供的数据，我们有以下能量值和相应的权重：能量 (V) 权重 202.15E-0520.12.29E-0520.22.43E0520.32.580520.42.74052.52.9152.63.0 952.7.29052.8.49E-0520.93.71E05213.94E-0521.14.15E05 如果您有任何问题或希望我进一步协助您处理这些数据，请告诉我。

现在我将给你一张ElektaAC的光谱表，我希望你将其转换为topas格式：

请提供Elekta LINAC的光谱表，我将为您将其转换为AS格式。

E (MeV) Elekta (6 MV)
0.25 0.173E-04(1.5%)\n .5 .100E03(0.5%)\n .75 .110E03(.5%)\n 1 0.952E-04(0.5%)\n 1.25 .827E04(.5\n .5 .721E04(. \n . .63E04(. \n 2 .55E04(.%) \n . .493E04(0.5%)\n 2.5 .437E04(.6\n 2.5 .386E04(.6\n 3 .34E4(.6\n . .30E4(.6\n . .27E4(.7\n 3.7 0.240E-04(.7\n 4 .21E04(.7\n . .18E(.8\n . .16E(.8\n .14E(.9\n 5 .121E04(.9\n 5.2 .101E4(1.0\n . .81E0(.1\n . .64E.2\n 6 .45E.4\n .0.300E-05(1.8\n 6.5 .15E0(.4\n .7 .73E0(.4\n 7 .29E0(.0\n . .88E0(. \n .18E-07(19.8%)\n 7.75 0.277E-08(50.0%)\n 8 0.159E-08(70.8%)

全景提示的生成：
感谢您提供包含Elekta LINAC光谱的表格。以下是相同内容的TOPAS格式：

光谱

s:...Type	"Discrete"							
dv:...Values	0.25	0.5	0.75	1	1.25	1.5	1.75	2
	2.25	2.5	2.75	3	3.25	3.5	3.75	4
	4.25	4.5	4.75	5	5.25	5.5	5.75	6
	6.25	6.5	6.75	7	7.25	7.5	7.75	8
uv:...Weights	0.173E-04	0.100E-03	0.110E-03	0.952E-04	0.827E-04	0.721E-04	0.635E-04	0.557E-04
	0.493E-04	0.437E-04	0.386E-04	0.345E-04	0.302E-04	0.270E-04	0.240E-04	0.210E-04
	0.186E-04	0.164E-04	0.142E-04	0.121E-04	0.101E-04	0.817E-05	0.645E-05	0.455E-05
	0.300E-05	0.156E-05	0.734E-06	0.294E-06	0.882E-07	0.188E-07	0.277E-08	0.159E-08

我希望这能帮到你。如果你有任何问题或需要进一步的帮助，请告诉我。

压缩提示的生成：
感谢您提供ElektaAC光谱的表格。以下是如何以TOPAS格式进行格式化：

光谱

s:...Type	"Discrete"							
dv:...Values	0.25	1.5	2	3.2	4	5	6	7
	7.25	7.5	7.75	8				
uv:...Weights	173E04	.10E-0	.11E1	.950	.82	.63E-0	.55E0	.4943
	.345E0	.30E0	.27	.24E0	.16E0	.714E52	.64E0	6.45E0
	.30E0	.15E0	.73E0	.294E-06	.882E-07	.188E-07	.277E-08	.159E-08

请告诉我 如果您有任何问题或需要进一步的帮助，请随时联系。这些数据的相符性。
选择性上下文的生成：
对不起，我看不到您提到的表格。您能否提供给我表格，以便我可以帮助您将其转换为TOPAS格式？

Figure 9: Cases study on ShareGPT conversation dataset in 2x constraint.

Compressed Prompt:

Redshift drift in a universe with structure II: Light rays propagated through aian Nbody simulation

koksbang@cp3.sdk

CP3-Orins,mark,, Mmark

Theshiftift is computed alongs propagating through aulated universe on the Newtonian Nbody simulation code GGET2 with a perturbedmann-Lreer metric in theian gauge.

It is that the meanshiftift is to theft of the meanshift to the precision of the numericalations and that this is to a high of cancellation between two dominant components theshiftift.

This result is contrary to earlier findings based on inhomogeneous cosmological models exhibiting cosmic backreaction.

For simplicity, the results neglect contributions from optical drift.

Based on a study of the redshift drift in a Lemaitre-Tolman-Bondi model, the optical drift effects are estimated to be at most of order 10% of the redshift drift signal.

The Generation of Full-shot Prompt:

The article discusses the computation of redshift drift along light rays in a simulated universe based on the Newtonian N-body simulation code GADGET-2 combined with a perturbed Friedmann-Lemaître-Robertson-Walker metric in the Newtonian gauge. The study finds that the mean redshift drift is equal to the drift of the mean redshift to the precision of the numerical computations, and this is due to a high degree of cancellation between two dominant components of the redshift drift. The study neglects contributions from optical drift, but it is estimated that these effects are at most of order 10% of the redshift drift signal. The study also finds that the redshift drift contribution from peculiar acceleration of the emitter is negligible in the simulation setup, but it is expected that this contribution will be larger for real observations.

The Generation of Compressed Prompt:

The document discusses the computation of redshift drift in a simulated universe using the Nbody simulation code GGET2 with a perturbed Lemaître-Lemaître metric. The results show that the mean shift is equivalent to the shift of the mean to a high degree of cancellation between two dominant components. This contradicts earlier findings based on inhomogeneous cosmological models exhibiting cosmic backreaction. The document also estimates that optical drift effects are at most 10% of the redshift drift signal.

The Generation of Selective-Context:

The document discusses the use of redshift drift measurements, which were first proposed in 1962. The study involves analyzing non-radial light rays observed by three different observers with non-radial lines of sight. The present-day density profile of the LTB model is shown in a figure, and quadri-linear interpolation is used to interpolate between snapshots. The observation time is set to 30 years, and the mean and fluctuations of the individual components of the redshift drift are shown in another figure. The study does not address the effects of peculiar acceleration.

Figure 10: Cases study on Arxiv-March23 summarization dataset in 200 tokens constraint.

压缩提示：

红移漂移在具有结构 II 的宇宙中：光线通过 aian Nbody 模拟传播

koksbang@cp3.sdk CP3-
Orins,mark,, Mmark

在牛顿N体模拟代码GGET2中，沿着在扰动的曼德尔布罗特度量下的宇宙传播计算 $\{v^*\}$ 的变化。

它是指均值偏移是均值偏移的精度与数值之间的关系，并且这与两个主导成分之间的高消除有关。

这个结果与早期基于表现出宇宙反应的非均匀宇宙模型的发现相悖。

为了简单起见，结果忽略了光学漂移的贡献。

基于对Lemaître-Tolman-Bondi模型中红移漂移的研究，光学漂移效应估计最多为红移漂移信号的10%。

全景提示的生成：

本文讨论了在基于牛顿N体模拟代码GADGET-2的模拟宇宙中，沿光线计算红移漂移的过程，该代码结合了在牛顿规范下扰动的弗里德曼-勒梅特-罗伯逊-沃克度规。研究发现，平均红移漂移等于平均红移的漂移，精度与数值计算相符，这归因于红移漂移的两个主要成分之间的高度抵消。研究忽略了光学漂移的贡献，但估计这些效应至多占红移漂移信号的10%。研究还发现，发射体的特殊加速度对红移漂移的贡献在模拟设置中可以忽略不计，但预计在实际观测中这一贡献会更大。

压缩提示的生成：

该文档讨论了在使用扰动的Lemaître-Lemaître度量的Nbody模拟代码GGET2的模拟宇宙中计算红移漂移。结果表明，平均漂移等同于在两个主要成分之间高度抵消的情况下，均值的漂移。这与基于表现出宇宙反应的非均匀宇宙模型的早期发现相矛盾。该文档还估计，光学漂移效应至多占红移漂移信号的10%。

选择性上下文的生成：

本文讨论了红移漂移测量的使用，该方法首次提出于1962年。该研究涉及分析由三位不同观察者以非径向视线观察的非径向光线。LTB模型的现今密度轮廓在一幅图中展示，采用四线性插值法在快照之间进行插值。观察时间设定为30年，红移漂移各个分量的均值和波动在另一幅图中展示。该研究未涉及特殊加速度的影响。

Figure 10: Cases study on Arxiv-March23 summarization dataset in 200 tokens constraint.