

Generative Prompt Internalization

Haebin Shin^{1,2*} Lei Ji² Yeyun Gong² Sungdong Kim¹ Eunbi Choi Minjoon Seo¹

¹KAIST AI ²Microsoft Research
{haebin.shin, minjoon}@kaist.ac.kr

Abstract

Prompts used in recent large language model based applications are often fixed and lengthy, leading to significant computational overhead. To address this challenge, we propose Generative Prompt Internalization (GenPI), a lightweight method that employs a joint training approach. GenPI not only replicates the behavior of models with prompt inputs but also generates the content of the prompt along with reasons for why the model’s behavior should change accordingly. We demonstrate that our approach effectively internalizes complex prompts across various agent-based application scenarios. For effective training without interactions with the dedicated environments, we introduce a data synthesis technique that autonomously collects conversational datasets by swapping the roles of the agent and environment. This method is especially useful in scenarios where only a predefined prompt is available without a corresponding training dataset. By internalizing complex prompts, Generative Prompt Internalization enables high-performance and efficient inference without the need for explicit prompts.¹

1 Introduction

In the real world, for inference in large language model (LLM) based applications (e.g. ChatGPT), fixed and complex prompts are often used repeatedly. Although advanced prompts can improve performance, their dependence on lengthy prompts raises concerns regarding computational efficiency for service providers and cost-effectiveness for users. Longer prompts can considerably increase computational overhead, especially during multi-turn inference, making it a potential bottleneck in practical applications. Therefore, finding effective strategies to optimize these prompts while maintaining performance is crucial for the practical deployment of LLM-based applications.

To address this issue, existing methods can be categorized into two approaches. One practical approach only for efficient prompt computation is compressing prompts into external embeddings (Mu et al., 2023; Chevalier et al., 2023; Ge et al., 2024) or compressed text (Jiang et al., 2023b; Li et al., 2023; Jiang et al., 2024; Pan et al., 2024). However, these methods still require additional tokens, limiting their ability to fully internalize prompts and effectively modify the model’s behavior as intended. Alternatively, fine-tuning (Kim and Rush, 2016; Zou et al., 2024) or distillation (Askell et al., 2021; Snell et al., 2022; Choi et al., 2023; Li et al., 2024), have been explored as canonical approaches for internalizing prompts. These methods adjust the language model’s behavior to follow the prompt’s intentions without requiring a prompt during inference.

However, these internalization methods are limited in that the model cannot reference the prompt’s content during training. Instead, they rely on indirect training based on the model’s output (Kim and Rush, 2016) or distribution (Askell et al., 2021; Snell et al., 2022; Choi et al., 2023; Li et al., 2024) of the original model when the prompt is provided. This leads to significant performance degradation in information-rich tasks or specific requirements, such as agent’s action requirements or schema grounding (Choi et al., 2023).

To overcome the limitations, we propose Generative Prompt Internalization (GenPI), a method that is trained to generate the target prompt, rather than merely using it as input. GenPI employs joint loss training, combining two approaches: 1) mimicking the teacher’s output to guide behavior, similar to distillation approaches (Askell et al., 2021; Snell et al., 2022; Choi et al., 2023; Li et al., 2024), and 2) generating the content of the prompt while inferring why the output should change based on that prompt. GenPI employs a lightweight internalization process for each prompt, requiring only 0.5%

* Work done during internship at Microsoft Research

¹Code are available at <https://github.com/kaistai/GenPI>

additional parameters on 1,000 samples. Focusing on scenario where only prompts are available without a corresponding training dataset for internalization (Liu et al., 2023; Choi et al., 2023), we also introduce a data synthesis technique to generate a multi-turn pseudo conversational dataset. By simply swapping the roles of user and assistant within the given context, the model simulates both sides of conversation, enabling it to autonomously collect conversational datasets.

We evaluate our method in agent application scenarios with lengthy prompts using Agent-Bench (Liu et al., 2023). GenPI maintains strong performance even without the need for prompt input, while outperforming other distillation or compression-based baselines. GenPI achieves 100% performance retention on OS interaction agent task, while maintaining at least 82% performance on web-based agent tasks with over 1,000 tokens. Furthermore, GenPI demonstrates 39% efficiency improvement in handling environments with lengthy prompts, outperforming other compression-based methods.

2 Related Works

Prompt Internalization is a methodology designed to embed prompt information within a language model, enabling the model to perform various tasks without requiring explicit prompt input. For instance, Askell et al. (2021) internalize persona-related prompts to facilitate helpful, honest, and harmless alignment, while Snell et al. (2022) incorporates both prompts and scratch pads into the internalization process to enhance performance on more complex tasks. Li et al. (2024) applies prompt internalization to internalize demonstrations for in-context learning scenarios. While these methods focus on internalizing broad and general coarse-grained information, Choi et al. (2023); Zou et al. (2024) introduce a more fine-grained approach, targeting more specific and predetermined prompts. Choi et al. (2023) focuses on short chat histories or task-specific instructions, and Zou et al. (2024) relies on a tailored training dataset to retrieve similar examples based on the given user input. Following these previous works, our goal is to enable the model to internalize the specific predetermined prompts. However, existing methods still limit the model’s ability to directly learn the content of the prompt, as they rely on training a student model using outputs from a teacher model. We address this limitation by generating

prompts that allow the model to learn their content directly, enabling it to handle more realistic and information-rich prompts.

Prompt Compression is one of the practical approach to reducing the computational overhead caused by lengthy prompts. In case of the API-based large language models (LLMs) services, systematic caching solutions allow frequently used prompts to be stored between API calls.²³ For users of API-based LLMs, text-based prompt compression methods are proposed, where key segments of long prompts are selected on a token-by-token basis to generate a compressed version of the original prompt (Jiang et al., 2023b; Li et al., 2023; Jiang et al., 2024; Pan et al., 2024). Although this approach reduces prompt length, it often still results in relatively long token sequences, as essential tokens must be retained. On the other hand, embedding-based prompt compression methods generate cached token embeddings for prompts, which can be utilized as a more efficient representation in LLMs (Mu et al., 2023; Chevalier et al., 2023; Ge et al., 2024). These methods offer the advantage of using fewer token embeddings compared to text-based compression but often require modifications to the model architecture, making it challenging to leverage the compressed vectors across different models.

3 Problem Definition

Following Choi et al. (2023), we assume a scenario where an application-specific prompt p is predetermined. Our goal is to guide the model to behave as if the prompt is given, even in its absence. Similar to the previous distillation approaches (Askell et al., 2021; Snell et al., 2022; Choi et al., 2023; Li et al., 2024), we define the teacher model T and the student model S based on whether the prompt p is provided to the same model θ .

The teacher model T is defined as a function that takes the prompt p and the input x_i at turn i , generating the teacher’s output: $y_i^T = T(x_i, p) = f_T(x_i, p; \theta)$. The student model S is a function that takes the input x_i at turn i , generating the student’s output: $y_i^S = S(x_i) = f_S(x_i; \theta)$. We approximate the student model’s behavior to match that of the teacher model, conditioned on the prompt p over multiple turns, as shown in Equation 1,

$$P(y_i^S | x_i) \approx P(y_i^T | x_i, p). \quad (1)$$

²<https://www.anthropic.com/news/prompt-caching>

³<https://ai.google.dev/gemini-api/docs/caching>

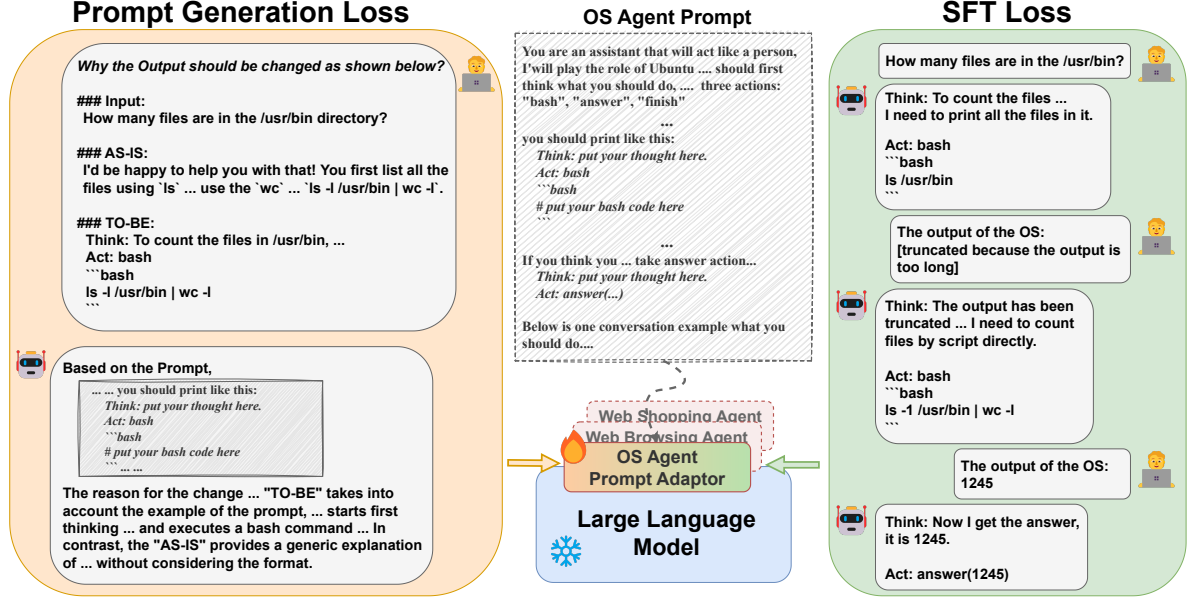


Figure 1: Overview of Generative Prompt Internalization. SFT loss learns the teacher model’s behavior based on the user input. Prompt Generation loss internalizes the prompt by generating both the content of the prompt and the reason for why the model’s behavior should be modified. This process is guided by comparing the student model’s output (“AS-IS”) with the teacher model’s output (“TO-BE”). SFT loss and Prompt Generation loss are combined into a joint loss to train the prompt-specific adaptor.

4 Generative Prompt Internalization

In this section, we demonstrate our novel method, Generative Prompt Internalization (GenPI), which internalizes the given prompt by generating its own contents. Basically, GenPI involves training the student model to mimic the behavior of the teacher model given the prompt p . We finetune the student model on the multi-turn outputs generated by the teacher model ($y_i^T, i \in \{0, 1, \dots, N\}$), inspired by sequence-level knowledge distillation approaches (Kim and Rush, 2016; Touvron et al., 2023b) which follow the hard label distribution from the teacher model.

$$\mathcal{L}_{SFT} = - \sum_{i=0}^N \log P(y_i^T | x_{<i}, y_{<i}^T, x_i) \quad (2)$$

However, the model is still unable to learn the content of the prompt directly; it only learns indirectly through the teacher’s output. We introduce an additional loss function, Prompt Generation loss (PG), where the loss is calculated directly on the prompt. This loss involves training the student model to understand why its behavior should align with the teacher’s behavior based on the prompt content. As illustrated in Figure 1, for a given input x , the model is trained to generate a prompt p along with a reason r for why the output should

be changed. In this process, the student’s output y_i^S is considered the initial state (“AS-IS”), while the teacher’s output y_i^T represents the desired state (“TO-BE”). This process is formalized in Equation 3:

$$\mathcal{L}_{PG} = - \log P(p, r | y_0^S, y_0^T, x). \quad (3)$$

We utilize a hyperparameter λ to combine the losses into a joint loss function, resulting in the final joint loss function:

$$\mathcal{L} = (1 - \lambda) \mathcal{L}_{PG} + \lambda \mathcal{L}_{SFT} \quad (4)$$

Our goal is to enable lightweight training and inference for each prompt, while also adapting to changes in the prompt effectively. To achieve this, we employ QLoRA (Dettmers et al., 2023) to learn prompt-specific adaptors, allowing us to tailor our approach to each individual prompt.

5 Components for Prompt Internalization

Following Choi et al. (2023), we assume a realistic scenario where the predetermined prompt has never been encountered before and is not included in any training dataset for prompt internalization. Consequently, we first generate the components as a pseudo training dataset from the prompt p .

Pseudo User Input. Similar to typical query generators (Lewis et al., 2021; Choi et al., 2023; Oh et al., 2024), we simply utilize a large language model to generate pseudo user input x by prompting to generate a probably questionable user input from the given prompt p . In this paper, we generate only 1,000 pseudo user inputs. Implementation details and examples are provided in Appendix A.1 and Appendix A.2.

Pseudo Conversational Outputs. In many task-specific applications, such as agent applications, the model interacts with the real environment across multiple turns (Liu et al., 2023; Zeng et al., 2023; Chen et al., 2024). This means that collecting multi-turn conversation outputs requires interacting with the actual environment at every turn, which is a non-trivial problem. As our primary objective is to gather the teacher model’s behavioral patterns, rather than to optimize the task performance, we introduce a simple method: *Self Role-Playing Conversation*. This method simply involves reversing the role between agent and environment. As illustrated in Figure 2, we provide the model θ with a role-reversed context to simulate the task environment. By swapping the agent and environment roles in the prompt, a single model θ can effectively embody both environment and agent personas, enabling the collection of self-conversational outputs y . We report the examples and additional quality evaluation in Appendix A.3. Implementation details and examples are provided in Appendix A.1 and Appendix A.2.

Reason. To collect the supervision for the reason r , we utilize a large language model by prompting to generate from the prompt p , user input x , and outputs y^S, y^T . As illustrated in Figure 1, the reason explains why the student’s output (“AS-IS”) should be changed to teacher’s output (“TO-BE”). Implementation details and examples are provided in Appendix A.1 and Appendix A.2.

6 Experiments Setup

6.1 Dataset

We evaluate our method on the three agent benchmark tasks: OS-Interaction (Liu et al., 2023), Web Browsing (Deng et al., 2023), Web Shopping (Yao et al., 2022). As mentioned in Section 5, we assume our scenario where an application-specific prompt p is predetermined before the model is deployed. For that, we utilize the AgentBench (Liu

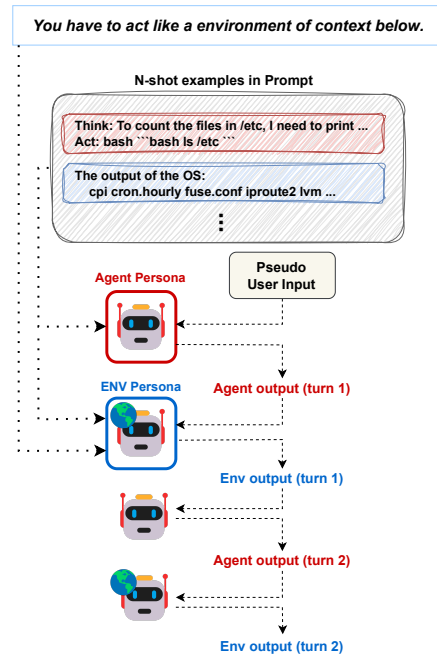


Figure 2: Self Role-Playing conversation. Collecting pseudo conversational output by switching the role in the prompt.

et al., 2023) settings, which organize a prompt with multi-turn evaluation samples for each task. Each task’s prompt consists of general task description, agent’s action space description, and shot-examples for agent behavior. Detailed information for each task’s prompt is described in Appendix B.

OS Interaction (Liu et al., 2023). This task involves interacting with an Ubuntu Docker container using bash commands and committing answers, with a 474 token⁴ prompt and 144 evaluation samples. The agent’s performance is measured by the Success Rate.

Web Browsing (Deng et al., 2023). This task formulates element selection as a multi-choice QA problem. We follow the AgentBench (Liu et al., 2023) setting, where the agent operates within an HTML action space (e.g., click, type, select options) using a 1,424 token prompt and 100 evaluation samples. The agent is evaluated using a Success Rate metric, which is based on two criteria: the correctness of the selected element and whether the predicted operation matches the ground truth value for each step.

⁴All tokens are calculated by the LLaMA tokenizer (Dubey et al., 2024).

Web Shopping (Yao et al., 2022). In this task, the agent navigates scraped *amazon.com* pages to identify the correct product using click or search actions. The task consists of a 1,285 token prompt and 200 evaluation samples. The task is completed when the agent clicks “buy now” or exceeds the turn limit, and performance is evaluated using a reward function that assesses the similarity between the expected and actual product attributes as following the Liu et al. (2023). Please refer to Appendix B for more details on the metric.

6.2 Baselines

We explore various baselines to internalize the prompt, ranging from distillation approaches (Snell et al., 2022; Choi et al., 2023) to compression approaches (Pan et al., 2024; Ge et al., 2024).

Full Fine-tuning. Since previous typical prompt internalization methods are based on much smaller model (Choi et al., 2023) or are not publicly available (Snell et al., 2022), we adopt recent knowledge distillation approaches (Lin et al., 2020; Gu et al., 2024; Ko et al., 2024) for autoregressive large language models in our prompt internalization setup. (1) The basic distillation approach (Hinton et al., 2015) employs the Kullback-Leibler Divergence loss (KLD) between the logit distributions of the student and teacher model. (2) Sequence-level Knowledge Distillation (SeqKD) (Kim and Rush, 2016) enforces the student model to generate the teacher model’s outputs on a fixed dataset. (3) As a strong baseline, we also employ a joint loss (SeqKD+KLD), inspired by recent knowledge distillation works (Song et al., 2020; Gu et al., 2024), which incorporates the language modeling loss during distillation. This approach can be interpreted as a hybrid distillation loss that combines the benefits of both soft labels and hard labels from the teacher model.

Prompt Prepending. One straightforward approach to consider is prepending the prompt during finetuning. If the prompt is consistently prepended during training, we expect the model to indirectly contextualize the prompt. However, since the model needs to predict without the prompt during inference, there is a potential mismatch between the training and inference distributions. To address this concern, we compare two baseline approaches: (1) always prepending the prompt during training (100% probability), and (2) stochastically

prepending the prompt during training (50% probability). This baseline utilizes QLoRA (Dettmers et al., 2023) with the same settings as our method. This approach can be viewed as an extension of the method that relies exclusively on SFT loss for training.

Text-based Prompt Compression. LLMingua-2 (Pan et al., 2024) is a prompt-agnostic method for generating compressed texts. LLMingua-2 explicitly compresses tokens using a smaller model, such as XLM-RoBERTa-large (Conneau et al., 2020). By performing with the compressed prompt, this baseline is expected to achieve efficient inference while maintaining comparable performance.

Embedding-based Prompt Compression. We utilize ICAE (Ge et al., 2024) to compress the prompt into cached prompt embeddings. Following Ge et al. (2024), we compress the prompt into 128 tokens and prepend compressed tokens to the user input at each inference time. Since the official checkpoint of ICAE exhibits significantly low performance on AgentBench (Liu et al., 2023), we finetune the baseline specifically for AgentBench (Liu et al., 2023). Additional details regarding this baseline are described in Appendix C.3.

Upper Bound. We utilize the teacher model as an upper bound that inputs the full prompt, consistent with previous studies (Snell et al., 2022; Choi et al., 2023). In our preliminary experiments (see Appendix C.2), we observe that the fine-tuned model performed worse than the original model in AgentBench (Liu et al., 2023). Consequently, we consider the original model with the full prompt as the teacher model following the setup of Liu et al. (2023). Since ICAE (Ge et al., 2024) is based on the Mistral-7B (Jiang et al., 2023a) model, we establish a separate upper bound specifically for ICAE (Ge et al., 2024) to ensure a fair comparison. For a detailed explanation of the upper bound, please refer to the Appendix C.2.

6.3 Implementation Details

To internalize the agent-based prompt into the language model, we utilize the LLaMA-3-8B-Instruct (Dubey et al., 2024) as the target model. Following our scenario in Section 5, we fine-tune the model using the 1,000 pairs of pseudo dataset as our train dataset, and this is applied equally to all baselines. We utilize QLoRA (Dettmers et al., 2023) with rank $r = 16$ only requires 0.5% of pa-

Methods	w/o Prompt	OS Interaction		Web Browsing		Web Shopping	
		SR	Norm.	SR	Norm.	Rewards	Norm.
Upper Bound	✗	17.36	100.00	17	100.00	54.16	100.00
TEXT-BASED COMPRESSION							
LLMLingua-2 (x0.9)	✗	4.16	23.96	9	52.94	50.78	93.76
LLMLingua-2 (x0.7)	✗	3.47	19.99	1	5.88	43.69	80.67
LLMLingua-2 (x0.3)	✗	0	0.00	0	0.00	0	0.00
EMBEDDING-BASED COMPRESSION							
Upper Bound*	✗	10.41	100.00	13	100.00	11	100.00
ICAE	✗	4.16	39.96	0	0.00	1.14	10.36
FULL FINETUNE							
KLD	✓	0	0.00	0	0.00	0	0.00
SeqKD	✓	3.47	19.99	15	88.24	38.79	71.62
SeqKD + KLD	✓	4.86	28.00	15	88.24	40.26	74.34
PROMPT PREPENDING							
100% probability	✓	0	0.00	0	0.00	0	0
50% probability	✓	13.19	75.98	12	70.59	31.09	57.40
OURS							
only SFT loss	✓	<u>15.97</u>	<u>91.99</u>	12	70.59	35.64	65.81
Generative Prompt Internalization	✓	17.36	100.00	<u>14</u>	<u>82.35</u>	<u>44.46</u>	<u>82.09</u>

Table 1: Performance evaluation results for AgentBench (Liu et al., 2023) dataset. *SR* denotes the Success Rate, and *Norm.* denotes the normalized score using Upper Bound following Choi et al. (2023). Since ICAE (Ge et al., 2024) is based on Mistral 7B (Jiang et al., 2023a), we report the normalized score using Mistral-7B-instruct-v0.2 as the upper bound score (Upper Bound*). The best results are in bold, while second-best ones are underlined.

rameters. Additional training details are described in Appendix C.4.

7 Results

7.1 Performance

Compression baselines. LLMLingua-2 (Pan et al., 2024) exhibits a significant performance drop as the compression rate increases. Although the semantics of the prompt can be inferred, the loss of format crucial for agent tasks leads to substantial performance drops. In the Web Shopping scenario, LLMLingua-2 (Pan et al., 2024) achieves the best performance at a compression rate of 90%. However, compression rates exceeding 30% led to a failure in all tasks. In the case of ICAE (Ge et al., 2024), despite fine-tuning, it struggles to handle the agent application prompts. Many embedding-based compression approaches (Mu et al., 2023; Chevalier et al., 2023), including ICAE (Ge et al., 2024), are primarily optimized for general texts such as articles, and thus exhibit limitations in compressing task-specific information, as required by AgentBench (Liu et al., 2023).

Distillation/Finetune Baselines. When trained with only the basic distillation loss (KLD), the

model fails in AgentBench (Liu et al., 2023), suggesting that 1,000 training examples may not be enough to train the entire model. While SeqKD helps promising results in the Web Browsing task, its performance is still lacking in other tasks. Among prompt prepending baselines, stochastically prepending prompts demonstrates some performance gains but ultimately fails to overcome the distribution mismatch at inference time, resulting in poorer performance compared to our approach.

Generative Prompt Internalization. GenPI achieves superior performance in the OS interaction task, reaching 100% of the upper bound. It also consistently demonstrates high performance of over 82% across Web Browsing and Web Shopping tasks, both of which have longer prompts that exceed 1,000 tokens in length. The incorporation of PG loss into SFT loss shows a notable performance improvement of approximately 25% in web shopping tasks. Furthermore, when compared to Prompt Prepending which is based on SFT loss and QLoRA (Detmeters et al., 2023) adaptor, our approach exhibits higher performance by leveraging PG loss, rather than relying solely on prepending prompts to the input.

	OS	WB	WS
Ours	17.36	14	44.46
PROMPT GENERATION LOSS ABLATION			
w/o reason	14.58	14	38.41
w/o prompt	13.19	13	32.34
JOINT LOSS ABLATION			
w/o \mathcal{L}_{PG}	15.97	12	35.64
w/o \mathcal{L}_{SFT}	0	0	0

Table 2: Results of Ablation Studies for Prompt Generation loss and joint loss. *WB* refers to the Web Browser task, and *WS* refers to the Web Shopping task.

7.2 Analysis

To understand the factors behind the performance improvement of GenPI, we perform three types of ablation studies. First, we compare the impact of PG loss and SFT loss on task performance. Second, we investigate the influence of the reason and prompt in PG loss by removing each component during generation. Finally, instead of omitting components, we explore the effect of reordering the input and output elements in PG loss while keeping the overall information level constant.

7.2.1 Impact of the Loss

As shown in Table 2, the removal of SFT loss causes the model to suffer significantly in task performance. This suggests that task performance is fundamentally dependent on SFT loss. However, incorporating PG loss consistently improves performance across all tasks. Our findings indicate that while SFT loss focuses on task behavior, PG loss plays a complementary role in enhancing task performance by concentrating on prompt internalization.

7.2.2 Impact of the Reason/Prompt

In Table 2, the prompt p has a greater impact on performance than the reason r across all tasks. This is interpreted as being due to the fact that the prompt is longer in length than the reason and generally contains more information, thereby exerting a more significant influence on the Prompt Generation loss. This highlights our primary contribution, directly generating the prompt itself, allowing the model to internalize and utilize prompt information more effectively.

7.2.3 Impact of Input/Output Ordering

Despite ensuring consistency in the overall amount of information in Table 3, we observe a decline

Ordering	OS	WB	WS
$P(p, r x, y)$ (Ours)	17.36	14	44.46
$P(r p, x, y)$	14.58	14	36.26
$P(p x, y, r)$	15.27	13	35.51

Table 3: Results of Input/Output Ordering Analysis.

in overall performance when certain components are removed during the generation process similar to Table 2. Except for the OS task which has relatively short prompt length than Web Browsing or Web Shopping, $P(r|p, x, y)$ consistently outperforms $P(p|x, y, r)$. This result is attributed to the fact that providing the model with a prompt, input, and output, followed by generating a reason, aligns more closely with the typical dialogue flow of an LLM, compared to inferring the prompt from an input, output, and reason. Since the prompt generally contains more information than the reason, inferring a prompt based on the reason is likely to be a more challenging task for the model.

Similarly, when comparing *w/o reason* in Table 2 (equivalent to $P(p|x, y)$) and $P(p|x, y, r)$, we see that while incorporating the reason as input slightly improved scores in the OS task, the performance drops in the web browsing and web shopping tasks. The prompt in the OS task is relatively shorter, making it easier to infer prompts based on the reason. However, as the length of the prompt increases, it becomes more challenging for the model, leading to a negative impact on training. When comparing with *w/o prompt* in Table 2 (equivalent to $P(r|x, y)$) and $P(r|p, x, y)$, we observe that adding prompt information leads to an improvement in all tasks.

8 Efficiency

To compare the efficiency between Generative Prompt Internalization and other baselines that require fixed prompts, we sample an example with more than 5 turns from a Web Shopping task containing over 1200 tokens.⁵ We then compare the MACs (Multiply-Accumulate Operations), FLOPs (Floating Point Operations), and latency to LLaMA-3-8B-Instruct (Dubey et al., 2024) based baselines. All metrics were measured by DeepSpeed Profile⁶.

⁵Efficiency is measured on a single NVIDIA A6000 GPU and AMD EPYC 7513 CPU, featuring 32 physical cores.

⁶<https://www.deepspeed.ai/tutorials/flops-profiler/>

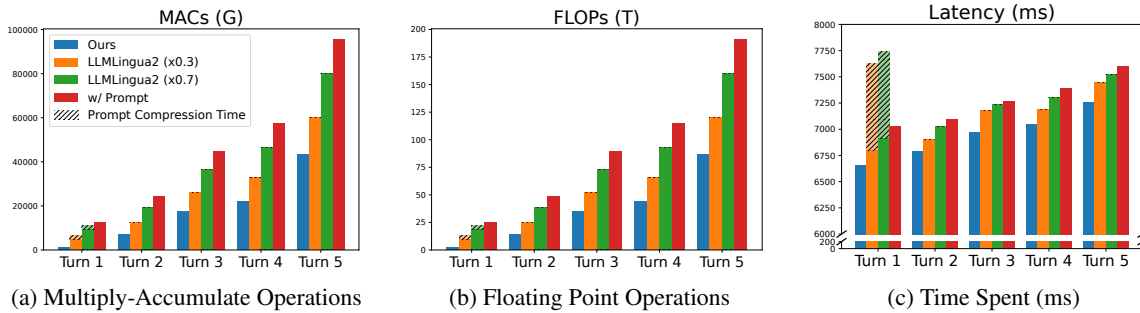


Figure 3: Comparison of computational overhead in LLaMA-based baselines as the conversation turn progresses. All generations within a turn are reported with KV caching (Pope et al., 2023) applied. Best viewed in color.

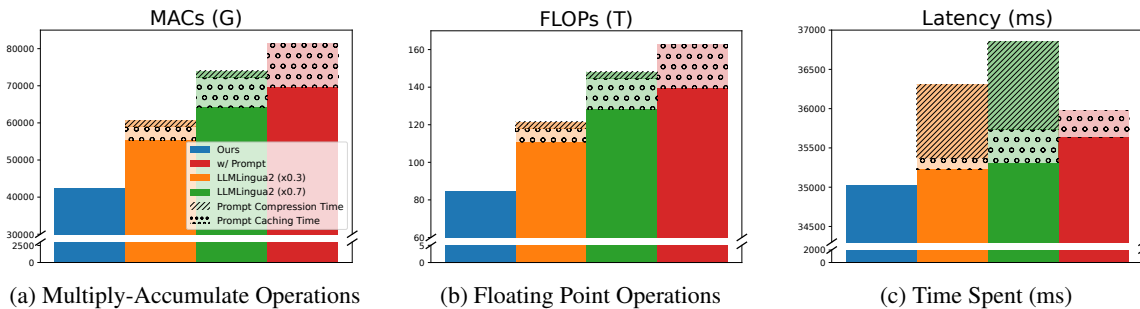


Figure 4: Comparison of computational overhead in LLaMA-based baselines applying KV caching (Pope et al., 2023) across the multi-turn conversation. Even if the previous contents are cached, a long context still creates extra overhead. Best viewed in color.

As shown in Figure 3, the overhead for baselines that require prompts accumulates and increases with each turn. While LLMLingua-2 (Pan et al., 2024) exhibits slightly better performance due to the reduced number of prompt tokens, it still incurs a larger overhead compared to ours. Furthermore, when considering compression overhead, the LLMLingua-2 (Pan et al., 2024) which is based on XLM-RoBERTa-large (Conneau et al., 2020), shows relatively low MACs and FLOPs than inferring with the prompt. However, it requires high compression latency proportional to the prompt length, resulting in extra latency in the first turn.

To simulate a more realistic scenario used in the real-world chat application, we conduct an additional experiment inspired by recent KV caching (Pope et al., 2023) works across multi-turn dialogues (Kwon et al., 2023; Zheng et al., 2023; Gao et al., 2024). We assume that the previous turn’s attention values are pre-cached to reduce the latency. The prompt is also considered as a previous turn’s conversation and is pre-cached. In this setting, the overhead between each turns is independent, and we report the conversation-level total

overhead over 5 turns.

In Figure 4, the prompt caching time executed before the first turn incurs higher computational overhead due to the use of a larger model compared to the compression model. From the second turn onward, although caching previous conversation histories eliminates redundant computations, the computational overhead still increases proportionally to the length of the prompt. Even without considering in caching time, GenPI demonstrates a 39% improvement in MACs and FLOPs, along with a 17% improvement in latency. Consequently, despite caching, baselines inevitably require more computational overhead compared to Generative Prompt Internalization.

9 Discussion

In this section, we discuss the strengths and limitations of two prompt internalization strategies: generalized prompt-agnostic compression and prompt-specific internalization approach.

Prompt compression approaches, such as LLMLingua-2 (Pan et al., 2024) and ICAE (Ge et al., 2024), effectively compress short instruc-

tions without requiring additional training for new prompts (Mu et al., 2023; Chevalier et al., 2023; Ge et al., 2024). However, as shown in Table 1, these methods struggle with long, information-rich application prompts, where key details are often lost in the compression process.

Prompt internalization approach, such as GenPI, overcome this limitation by explicitly internalizing prompts through targeted training. While GenPI requires training for each new prompt, this overhead is minimal in practical applications where only a limited set of prompts is needed. Once trained, switching between adapters incurs negligible computational cost, making the approach both flexible and scalable. Each adapter requires only 1,000 data samples for training. Even when generating the dataset with a 70B LLM, entire process can be completed in under an hour using four A6000 GPUs with vLLM (Kwon et al., 2023), while training takes approximately three hours on a single A6000 GPU.

10 Conclusion

In this paper, we propose a novel prompt internalization method, Generative Prompt Internalization (GenPI), which generates both the contents of the prompt and the reasoning behind changes to the model’s output while mimicking the behavior of the teacher model. To address the challenging scenario where only predetermined prompts are available without additional training data, we introduce Self Role-Playing Conversation, a method that generates a pseudo-conversational training dataset from the given prompt. Our approach demonstrates that even in scenarios with lengthy agent prompts, GenPI maintains high performance without relying on the prompt itself. Moreover, GenPI improves efficiency by fully internalizing the prompt without requiring any additional tokens.

Limitations

We assume the use of long and fixed prompts from realistic application scenarios. However, collecting official prompts from real-world applications (such as ChatGPT) poses significant challenges, leading us to rely on prompts from academic agent-based applications. This introduces a limitation, as only a representative prompt for each task are considered, and our approach handles a relatively small set of prompts. In future work, we plan to further explore context internalization in various domains,

including long chat histories, in-context learning, and retrieval-augmented generation (RAG), as well as multimodal applications involving video and image data.

Acknowledgements

This project was supported by Microsoft Research Asia. Additionally, this research was supported by the MSIT (Ministry of Science, ICT), Korea, under the Global Research Support Program in the Digital Field program (RS-2024-00436680) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation). And, this work also supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (RS-2024-00398115, Research on the reliability and coherence of outcomes produced by Generative AI).

References

- Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Benjamin Mann, Nova Dassarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, John Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Christopher Olah, and Jared Kaplan. 2021. A general language assistant as a laboratory for alignment. *arXiv*.
- Zehui Chen, Kuikun Liu, Qiuchen Wang, Jiangning Liu, Wenwei Zhang, Dahua Lin, Kai Chen, and Feng Zhao. 2024. Agent-flan: Designing data and methods of effective agent tuning for large language models. *arXiv*.
- Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *EMNLP*.
- Eunbi Choi, Yongrae Jo, Joel Jang, Joonwon Jang, and Minjoon Seo. 2023. Fixed input parameterization for efficient prompting. In *Findings of ACL*.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *ACL*.
- Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. In *NeurIPS*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. In *NeurIPS*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Alonso, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yearly, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Celebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Rapparthi, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong

Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gouget, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuwei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khan-delwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhota, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L.

- Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vitor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. 2024. The llama 3 herd of models. *arXiv*.
- Bin Gao, Zhuomin He, Puru Sharma, Qingxuan Kang, Djordje Jevdjic, Junbo Deng, Xingkun Yang, Zhou Yu, and Pengfei Zuo. 2024. Cost-efficient large language model serving for multi-turn conversations with cachedattention. In *USENIX ATC*.
- Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2024. In-context autoencoder for context compression in a large language model. In *ICLR*.
- Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. Minillm: Knowledge distillation of large language models. In *ICLR*.
- Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *arXiv*.
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Léo Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023a. Mistral 7b. *arXiv*.
- Huiqiang Jiang, Qianhui Wu, , Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *ACL*.
- Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023b. LLMLingua: Compressing prompts for accelerated inference of large language models. In *EMNLP*.
- Seungone Kim, Se June Joo, Doyoung Kim, Joel Jang, Seonghyeon Ye, Jamin Shin, and Minjoon Seo. 2023. The cot collection: Improving zero-shot and few-shot learning of language models via chain-of-thought fine-tuning. In *EMNLP*.
- Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *EMNLP*.
- Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. 2024. Distillm: Towards streamlined distillation for large language models. In *ICML*.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *SIGOPS*.
- Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021. PAQ: 65 million probably-asked questions and what you can do with them. *TACL*.
- Yichuan Li, Xiyao Ma, Sixing Lu, Kyumin Lee, Xiaohu Liu, and Chenlei Guo. 2024. Mend: Meta demonstration distillation for efficient and effective in-context learning. In *ICLR*.
- Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. 2023. Compressing context to enhance inference efficiency of large language models. In *EMNLP*.
- Alexander Lin, Jeremy Wohlwend, Howard Chen, and Tao Lei. 2020. Autoregressive knowledge distillation through imitation learning. In *EMNLP*.
- Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. 2023. Agentbench: Evaluating llms as agents. *arXiv*.
- Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. In *NeurIPS*.

- Hanseok Oh, Haebin Shin, Miyoung Ko, Hyunji Lee, and Minjoon Seo. 2024. Ktrl+f: Knowledge-augmented in-document search. In *NAACL*.
- Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Ruhle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. LLMingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of ACL*.
- Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. In *MLSys*.
- Charlie Snell, Dan Klein, and Ruiqi Zhong. 2022. Learning by distilling context. *arXiv*.
- Kaitao Song, Hao Sun, Xu Tan, Tao Qin, Jianfeng Lu, Hongzhi Liu, and Tie-Yan Liu. 2020. Lightpaff: A two-stage distillation framework for pre-training and fine-tuning. *arXiv*.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *arXiv*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruiti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.
- Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. In *NeurIPS*.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *ICLR*.
- Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2023. Agenttuning: Enabling generalized agent abilities for llms. *arXiv*.
- Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Jeff Huang, Chuyue Sun, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. 2023. Efficiently programming large language models using sglang. *arXiv*.
- Jiaru Zou, Mengyu Zhou, Tao Li, Shi Han, and Dongmei Zhang. 2024. Promptintern: Saving inference costs by internalizing recurrent prompt during large language model fine-tuning. *arXiv*.

A Details of Components Generation

A.1 Implementation details

Pseudo User Input. We utilize LLaMA-3-70B-Instruct (Dubey et al., 2024) as a pseudo user input generator. Each samples are generated by nucleus sampling with threshold 0.9 and temperature 1.0. To ensure the quality of pseudo user input, we randomly select 5 demonstrations from the validation samples of each task. System prompt and user prompt for Pseudo Input generation are as below.

```
The document below are the code of conduct for a
specific agent performing the given problem.
```

```
### This is the given Document:
```

```
{ context }
```

```
Based on the given document, generate several
problems for {taskname} task.
Generate only problems with a numbered list.
This is {num_gen_once} generated Problems.
```

```
{N-shot demonstrations }
```

Pseudo Conversational Outputs. We provide an additional system prompt for the environment persona model in the Self Role-Playing Conversation (Figure 2). In the N-shot examples within the given prompt p , we swap the roles of the agent and environment and collected the teacher’s conversational behavior patterns with a temperature of 0. In this paper, we use LLaMA-3-8B-Instruct (Dubey et al., 2024) as the target model θ .

Reason. As same as the pseudo input generator, we utilize LLaMA-3-70B-Instruct (Dubey et al., 2024) with nucleus sampling threshold 0.9 and temperature 0.7. We prompt the model to generate a reason of approximately 5 sentences to mitigate excessive hallucination. This is based on the statistics from the CoT Collection (Kim et al., 2023), which includes 1.84M rationales across 1,060 tasks, with an average length of about 3 sentences. System prompt and user prompt for reason generation are as below.

```
You are a commentator comparing the answers of two
models to a question. I will give you four
components: context, input, bad output, good
output. Bad Output is a response provided without
any context, only based on the input given. Good
Output is a response provided with both context and
input. Explain the reason for the change in
response, referring to the context, in about five
sentences.
```

```
### Context:
```

```
{ context }
```

```
### Input:
```

```
{ input }
```

```
### Bad Output:
```

```
{ student_output }
```

```
### Good Output:
```

```
{ teacher_output }
```

```
### Reason:
```

A.2 Components Examples

Figure 8, 9, 10 present examples from the conversational dataset, including a pseudo user input and pseudo conversational outputs generated by Self Role-Playing Conversation. In Figure 12, 13, 14, the examples of reason describe the differences in format and actions required from the prompt.

A.3 Quality of Self Role-Playing Conversation

As shown in Table 4, we perform quantitative evaluations using statistics and three distinct error types.

How similar is it to the gold turn? The primary objective of the pseudo-conversational outputs is to emulate multi-turn dialogue capabilities based on the content of the prompt. To this end, we consider the shot examples provided in the prompt as gold dialogue, and compare the similarity in the number of turns. Across all three tasks, the generated multi-turn dialogues successfully mimic the structure of the gold turns. Even in the Web Browsing task, which consists of N-shot single-turn examples, the number of turns remains comparable to the gold examples.

How many samples exhibit abnormally long conversations? We evaluate whether the conversation terminates due to reaching the maximum turn limit. We set the maximum turn limit for each task (10 turns for OS, 2 turns for Web Browsing, and 5 turns for Web Shopping), based on the shot examples and expected number of turns to find the answer following Liu et al. (2023). Among the 1,000 generated dialogue, only fewer than 2% samples exceed the maximum limit. We truncate this samples under the maximum limit.

How many samples are abnormally terminated? During the Self Role-Playing Conversation, we instruct the environment persona model to generate a special termination token when the agent generates the final output for the given user input. We consider samples to be abnormally terminated if there are no termination tokens. In Table 4, all

samples successfully stop the conversation with environment persona model’s termination token.

How many samples are missing the final action?

We evaluate whether the final action, which serves as the scoring target, is correctly generated. In this evaluation setup, we restrict the evaluated action space that can be inferred solely from the prompt. For instance, we evaluate whether OS tasks ended with ‘finish’ or ‘answer’ actions, Web Browsing tasks with ‘click’, ‘type’, ‘select’, ‘None’, and Web Shopping tasks with ‘search’ or ‘click’ actions. Note that the specific values for click or type actions are excluded on this quantitative evaluation. This information cannot be inferred from the prompt, as it pertains to the agent’s capability to respond based on the user input. In the OS task, approximately 5% of the samples omit the final action. As shown in Figure 11, while most samples in this group contain the correct bash commands, they lack the ‘answer’ or ‘finish’ keyword required for proper termination. Similarly, for the Web Browsing task, many samples in this group do not adhere to the predefined action space, generating undefined actions (e.g. COMPARE, ENTER USERNAME AND PASSWORD in Figure 11) in response to user input.

	OS	WB	WS
Gold Turns (in Prompt)	3	1	4
Avg. Turns	2.71	1.03	3.21
Max Turns Limit	1.8%	1.6%	1.7%
Abnormal Termination	0%	0%	0%
Abnormal Final Action	5.2%	3.9%	0%

Table 4: Overall Qualities and Statistics about Pseudo Conversational Outputs. Note that WB denotes Web Browsing task (Deng et al., 2023), and WS denotes Web Shopping task (Yao et al., 2022).

B Details for Evaluation Dataset

We follow the prompt configurations defined in AgentBench⁷ (Liu et al., 2023) for three agent-based tasks: OS Interaction (Liu et al., 2023), Web Browsing (Deng et al., 2023), and Web Shopping (Yao et al., 2022). Each prompt comprises a task description, a detailed explanation of the

⁷We select three tasks because several tasks in AgentBench (Liu et al., 2023) either lack sufficiently long prompts (DB task), exhibit significantly low performance on open-source LLMs (Game-grounded tasks), or face reproducibility issues due to Freebase server limitations (KG task).

agent’s expected actions, the formatting guidelines that the agent should adhere to, and a set of N-shot examples illustrating interactions between the agent and the environment. As suggested by Liu et al. (2023), all agent outputs are formatted using the Chain-of-Thought (CoT) style (Wei et al., 2022), which has become the standard approach for this type of evaluation in conjunction with action-based responses (Yao et al., 2023). We utilize these multi-turn prompts as the chat history context for our agent interactions.

OS Interaction (Liu et al., 2023). The prompt for the OS Interaction task includes a brief task description and the formulation of an interaction trajectory using a 1-shot example. An example of this prompt is illustrated in Figure 5. Agents are evaluated based on Success Rate (SR), determined by comparing the final output against the expected solution.

Web Browsing (Deng et al., 2023). The prompt for the Web Browsing task contains 3-shot CoT-style examples. An illustration of this prompt can be found in Figure 6. In alignment with Liu et al. (2023), we report Step Success Rate, which indicates the independent accuracy of each action step taken by the agent.

Web Shopping (Yao et al., 2022). The prompt for the Web Shopping task consists of the task description along with a 1-shot CoT-style example. During each turn, the agent interacts with the HTML text observation by exploring and making decisions accordingly. An example of this configuration is depicted in Figure 7. Consistent with Liu et al. (2023), we evaluate the agent’s performance using a reward metric, which quantifies the similarity between the expected attributes of a product and the attributes of the purchased item, mapping this similarity to a value between 0 and 1.

$$\text{Reward} = \frac{|G_{at} \cap Y_{at}| + |G_{op} \cap Y_{op}| + \mathbb{I}[y_{pr} \leq g_{pr}]}{|G_{at}| + |G_{op}| + 1} \cdot r_{type}$$

$$r_{type} = \begin{cases} 0, & \text{if TM} = 0, \\ 0.1, & \text{if TM} < 0.1, \\ 0.5, & \text{if TM} > 0.2 \text{ and } c = 1, \\ 1, & \text{otherwise} \end{cases}$$

G and Y denote the goal and the chosen product, respectively, while at and op represent attributes

and options. Note that r_{type} compares the product category sequences ($c = 1$ if matched) listed on the Amazon website. Additionally, TM represents a text match comparison between titles, focusing on the proun, noun, and propn tags. For further details on these metrics, please refer to Yao et al. (2022).

C Details for Baselines

C.1 Criteria of selecting the Base Model

We assume a multi-turn conversational application scenario, where an instruct-tuned model is deployed with a fixed, predetermined prompt. Due to this setup, we select LLaMA-3-8B-Instruct (Dubey et al., 2024) model as the base model instead of using LLaMA-3-8B.

C.2 Criteria of selecting Upper Bound

In case of Choi et al. (2023), they finetune a relatively small model (approximately 200M parameters) on gold training dataset, due to concerns that the language model may not adequately perform on unseen tasks, thereby limiting its effectiveness as an upper-bound model. In contrast, we examine a scenario where only a prompt is provided, without a corresponding training dataset for prompt internalization. Moreover, we specifically focus on a billion-scale language model applied to the AgentBench (Liu et al., 2023), which lacks a dedicated gold training dataset.

To establish a more robust upper bound under these conditions, we conduct preliminary experiments comparing the performance between a model fine-tuned on a pseudo training dataset and the original model. The model that demonstrates superior performance is then selected as the upper bound. As shown in Table 5, the original model outperforms the fine-tuned model when prompts are provided. This suggests that the limited size of the pseudo training dataset (only 1,000 examples) may be insufficient to optimize the LLM for agent-specific tasks, or that there exists a distributional mismatch between the pseudo training dataset and the actual gold test set. Since optimizing agent performance through fine-tuning is beyond the scope of this paper, we adopt the original model, which demonstrated higher performance, as the upper bound, following the approach of Snell et al. (2022); Liu et al. (2023).

	OS	WB	WS
original	17.36	17	54.16
fine-tuned	16.6	17	51.12

Table 5: Preliminary experiments to select the Upper Bound model. All models are based on LLaMA-3-8B-Instruct (Dubey et al., 2024) and are evaluated on their performance when provided with the agent prompt.

C.3 Criteria of selecting Embedding-based Compression Baseline

Recent embedding-based prompt compression approaches (Mu et al., 2023; Chevalier et al., 2023; Ge et al., 2024) have been developed as a general language models that first cache the given prompt first and then infer user input using compressed prompt embeddings. Gisting (Mu et al., 2023) and AutoCompressor is based on LLaMA-7B (Touvron et al., 2023a) and LLaMA-2-7B (Touvron et al., 2023b), respectively. These models perform significantly poorly on AgentBench (Liu et al., 2023) with scores close to zero. Therefore, we select the baseline ICAE (Ge et al., 2024), which is based on Mistral-7B-Instruct (Jiang et al., 2023a).

C.4 Implementation Details

We utilize a single NVIDIA A6000 GPU to train our method and baselines. For Generative Prompt Internalization, we apply QLoRA (Dettmers et al., 2023) with rank $r = 16$, 32 batch size, $1e-4$ learning rate, 10 epochs. The joint loss hyperparameter $\lambda = 0.7$ (as defined in Equation 4) is selected through hyperparameter tuning over $[0.3, 0.5, 0.7]$. For full finetuning baselines, we use 4 NVIDIA A100 80G GPUs with 64 batch size, $1e-5$ learning rate, 10 epochs, following the setting from Gu et al. (2024). Prompt Prepending, a variation of our approach, uses the same hyperparameters as GenPI. For ICAE (Ge et al., 2024), we follow the original settings with a LoRA rank of $r = 512$ and 128 compression tokens.

D Comparison with Chain-of-Thought

To discuss of the role of reason r in PG loss, we compare the differences between chain-of-thought (CoT) rationales (Wei et al., 2022) and reason r , focusing on their mechanisms, and impact on inference.

Rationale vs. Behavioral Change. CoT (Wei et al., 2022) generates explicit stepwise rationales

to explain the final answer for solving a task-intensive problem (e.g., an explicit reasoning process for solving a math problem). In contrast, reason r aims to compare the differences between the outputs of the teacher and student models, and identify the key changes in the model's behavior. That is, while indirectly describing the prompt p , PG loss is designed to induce behavioral changes in the model rather than explicitly providing explanations for the correct answer.

Inference Efficiency. Since GenPI does not generate rationales at inference time, it differs from CoT (Wei et al., 2022) in inference efficiency. By internalizing the prompt, PG loss ensures higher efficiency (in Section 8) while maintaining the original performance.


```

<USER>:
You are an assistant that will act like a person, I will play the role of linux(ubuntu) operating system. Your goal is to
implement the operations required by me or answer to the question proposed by me. For each of your turn, you should first
think what you should do, and then take exact one of the three actions: "bash", "finish" or "answer".

1. If you think you should execute some bash code, take bash action, and you should print like this:

Think: put your thought here.

Act: bash
```bash
put your bash code here
```

2. If you think you have finished the task, take finish action, and you should print like this:

Think: put your thought here.

Act: finish

3. If you think you have got the answer to the question, take answer action, and you should print like this:

Think: put your thought here.

Act: answer(Your answer to the question should be put in this pair of parentheses)

If the output is too long, I will truncate it. The truncated output is not complete. You have to deal with the truncating
problem by yourself. Attention, your bash code should not contain any input operation. Once again, you should take only exact
one of the three actions in each turn.

Now, my problem is:

tell me how many files are in the directory "/etc"?

<AGENT>:
Think: To count the files in /etc, I need to print all the files in it.

Act: bash
```bash
ls /etc
```

<USER>:
The output of the OS:
cpi cron.hourly fuse.conf iproute2 lvm networkd-dispatcher protocols selinux tmpfiles.d [truncated because the output is too
long]

<AGENT>:
Think: The output has been truncated because it is too long, so I need to count files by script directly.

Act: bash
```bash
ls -l /etc | wc -l
```

<USER>:
The output of the OS:
220

<AGENT>:
Think: Now I get the answer, it is 220.

Act: answer(220)

<USER>:
Now, I will start a new problem in a new OS. My problem is:
{user_input}

```

Figure 5: Agent Prompt for OS Interaction. Following the task setup from AgentBench (Liu et al., 2023), we describe all content, including the system prompt and demonstrations, as a multi-turn strategy using <USER> and <AGENT>.

```

<USER>:
'''
<html> <div> <div> <a tock home page /> <button id=0 book a reservation. toggle open> <span> Book a reservation </span>
</button> <button book a reservation. toggle open> </button> </div> <div> <select id=1 type> <option reservations true> Dine
in </option> <option pickup> Pickup </option> <option delivery> Delivery </option> <option events> Events </option> <option
wineries> Wineries </option> <option all> Everything </option> </select> <div id=2> <p> Celebrating and supporting leading
women shaking up the industry. </p> <span> Explore now </span> </div> </div> </div> </html>
'''

Based on the HIML webpage above, try to complete the following task:
Task: Check for pickup restaurant available in Boston, NY on March 18, 5pm with just one guest
Previous actions:
None
What should be the next action? Please select from the following choices (If the correct action is not in the page above,
please select A. 'None of the above'):

A. None of the above
B. <button id=0 book a reservation. toggle open> <span> Book a
C. <select id=1 type> <option reservations true> Dine in </option> <option
D. <div id=2> <p> Celebrating and supporting leading women shaking up

<AGENT>:
Thought: I need to select pickup restaurant first.
Answer: C.
Action: SELECT
Value: Pickup

<USER>:
'''
<html> <div> <main main> <section tabpanel> <div> <ul tablist> <li tab heading level 3 search and> </li> <li id=0 tab heading
level 3 search and> <span> Hotel </span> </li> <li tab heading level 3 search and> </li> <li tab heading level 3 search and>
</li> </ul> <div tabpanel> <div id=1> <div> <span> Dates* </span> <button button clear dates /> </div> <div> <label>
Travelers </label> <div> <p> 1 Adult </p> <button button> 1 Adult </button> <div dialog> <button button travel with a pet.
this> <span> Travel with a pet </span> </button> <div> <button button clear all fields> Clear all </button> <button button>
</button> </div> </div> </div> </div> </div> </div> </div> </div> </div> </div> </div> </div> </div> </div> </div> </div>
</h3> <ul id=2> <a mobile tools> </a> <a open united's tiktok feed in> </a> <a open united's facebook page in> </a> <a open
united's twitter feed in> </a> <a open united's youtube page in> </a> <a open united's instagram feed in> </a> <a open
united's linkedin profile in> </a> </ul> </div> </div> </div> </div> </div>
'''

Based on the HIML webpage above, try to complete the following task:
Task: Compare the fare types to book a 1-adult ticket from Springfield, IL to Austin, TX for April 29th 2023
Previous actions:
[combobox] Enter your departing city, airport name, or airpor... -> TYPE: SPRINGFIELD
[button] Springfield, IL, US (SPI) -> CLICK
[combobox] Enter your destination city, airport name, or airp... -> TYPE: AUSTIN
[button] Austin, TX, US (AUS) -> CLICK
What should be the next action? Please select from the following choices (If the correct action is not in the page above,
please select A. 'None of the above'):

A. None of the above
B. <li id=0 tab heading level 3 search and> <span> Hotel
C. <div id=1> <div> <span> Dates* </span> <button button clear dates
D. <ul id=2> <a mobile tools> </a> <a open united's tiktok

<AGENT>:
Thought: There are no elements that allow setting the date or viewing the fare, so there is no correct action.
Answer: A.

<USER>:
'''
<html> <div> <nav main menu> <ul> <li> <div button> Car Sales </div> <div id=0> <div> <div> <div> Buy A Car </div> <div> Plan
Your Purchase </div> </div> <div> <h4> Its Tax Refund Time. Treat Yourself to an Upgrade. </h4> <p> With a variety of
options, invest your refund in what you really want - a quality, used vehicle from Enterprise. </p> <a> View Inventory </a>
</div> </div> </div> </li> <div id=1> Enterprise Fleet Management </div> </ul> </nav> <div region> <button id=2 selected
pick-up date 03/19/2023> <span> <span> 19 </span> </span> <div> <span> <span> Mar </span> <span> <span> 2023 </span> </div> </span> </div>
</div> </div> </div>
'''

Based on the HIML webpage above, try to complete the following task:
Task: Find a mini van at Brooklyn City from April 5th to April 8th for a 22 year old renter.
Previous actions:
[searchbox] Pick-up & Return Location (ZIP, City or Airport) (... -> TYPE: Brooklyn
[option] Brooklyn, NY, US Select -> CLICK
What should be the next action? Please select from the following choices (If the correct action is not in the page above,
please select A. 'None of the above'):

A. None of the above
B. <div id=0> <div> <div> <div> Buy A Car </div> <div>
C. <div id=1> Enterprise Fleet Management </div>
D. <button id=2 selected pick-up date 03/19/2023> <span> <span> 19 </span>

<AGENT>:
Thought: The search has already been set to Brooklyn. Next, I should choose pick-up date.
Answer: D.
Action: CLICK

<USER>:
{ user_input }

```

Figure 6: Agent Prompt for Web Browsing. Following the task setup from AgentBench (Liu et al., 2023), Web Browsing agent prompt consists of 3-shot demonstrations. We describe all content as a multi-turn strategy using <USER> and <AGENT>.

```

<USER>:
You are web shopping.
I will give you instructions about what to do.
You have to follow the instructions.
Every round I will give you an observation and a list of available actions, you have to respond an action based on the state and instruction.
You can use search action if search is available.
You can click one of the buttons in clickables.
An action should be of the following structure:
search[keywords]
click[value]
If the action is not valid, perform nothing.
Keywords in search are up to you, but the value in click must be a value in the list of available actions.
Remember that your keywords in search should be carefully designed.
Your response should use the following format:

Thought:
I think ...

Action:
click[something]

<AGENT>:
Ok.

<USER>:
Observation:
"WebShop [SEP] Instruction: [SEP] i need a long lasting 6.76 fl oz bottle of l'eau d'issey, and price lower than 100.00 dollars [SEP] Search"

Available Actions:
{"has_search_bar": true, "clickables": ["..."]}

<AGENT>:
Thought:
I think I should use the search bar to look for the product I need.

Action:
search[l'eau d'issey 6.76 fl oz bottle price < 100.00]

<USER>:
Observation:
"Instruction: [SEP] i need a long lasting 6.76 fl oz bottle of l'eau d'issey, and price lower than 100.00 dollars [SEP] Back to Search [SEP] Page 1 (Total results: 50) [SEP] Next > [SEP] B000VOHH8I [SEP] L'eau D'issey By Issey Miyake for MenEau De Toilette Spray, 6.7 Fl Oz Bottle [SEP] $64.98 [SEP] B000MJZOPK [SEP] L'eau d'Issey by Issey Miyake for Women 3.3 oz Eau de Toilette Spray [SEP] $49.98 [SEP] B0012S249E [SEP] L'eau D'issey By Issey Miyake For Women. Shower Cream 6.7-Ounces [SEP] $31.36 [SEP] B01H8PGKZS [SEP] L'eau D'Issey FOR MEN by Issey Miyake - 6.7 oz EDT Spray [SEP] $67.97 [SEP] B00G3C8FHE [SEP] L'Eau d'Issey pour Homme - Eau de Toilette 4.2 fl oz [SEP] $51.25 [SEP] B000R94HRG [SEP] Issey Miyake L'Eau D'Issey Pour Homme Eau De Toilette Natural Spray [SEP] $44.99 [SEP] B000C214CO [SEP] Issey Miyake L'eau D'issey Eau de Toilette Spray for Men, 4.2 Fl Oz [SEP] $53.99 [SEP] B0018SBRDC [SEP] Issey Miyake L'eau d'Issey for Women EDT, White, 0.84 Fl Oz [SEP] $27.04 [SEP] B000XEAZ9Y [SEP] L'eau De Issey By Issey Miyake For Men. Eau De Toilette Spray 6.7 Fl Oz [SEP] $67.08 [SEP] B079HZR2RX [SEP] L'eau d'Issey Pure by Issey Miyake for Women 3.0 oz Nectar de Parfum Spray [SEP] $71.49"

Available Actions:
{"has_search_bar": false, "clickables": ["...", "...", "...", "...", "...", "...", "...", "...", "...", "...", "...", "..."]}

<AGENT>:
Thought:
I think I should click on the product I need, which is B000VOHH8I.

Action:
click[B000VOHH8I]

<USER>:
Observation:
"Instruction: [SEP] i need a long lasting 6.76 fl oz bottle of l'eau d'issey, and price lower than 100.00 dollars [SEP] Back to Search [SEP] < Prev [SEP] size [SEP] 2.5 fl oz [SEP] 6.76 fl oz (pack of 1) [SEP] L'eau D'issey By Issey Miyake for MenEau De Toilette Spray, 6.7 Fl Oz Bottle [SEP] Price: $64.98 [SEP] Rating: N.A. [SEP] Description [SEP] Features [SEP] Reviews [SEP] Buy Now"

Available Actions:
{"has_search_bar": false, "clickables": ["...", "...", "...", "...", "...", "...", "...", "..."]}

<AGENT>:
Thought:
I think I should click on the '6.76 fl oz (pack of 1)' option to select the size I need.

Action:
click[6.76 fl oz (pack of 1)]

<USER>:
Observation:
"Instruction: [SEP] i need a long lasting 6.76 fl oz bottle of l'eau d'issey, and price lower than 100.00 dollars [SEP] Back to Search [SEP] < Prev [SEP] size [SEP] 2.5 fl oz [SEP] 6.76 fl oz (pack of 1) [SEP] L'eau D'issey By Issey Miyake for MenEau De Toilette Spray, 6.7 Fl Oz Bottle [SEP] Price: $64.98 [SEP] Rating: N.A. [SEP] Description [SEP] Features [SEP] Reviews [SEP] Buy Now"

Available Actions:
{"has_search_bar": false, "clickables": ["...", "...", "...", "...", "...", "...", "...", "..."]}

<AGENT>:
Thought:
I think I should click on the 'Buy Now' button to purchase the product.

Action:
click[Buy Now]

<USER>:
{user_input}

```

Figure 7: Agent Prompt for Web Shopping. Following AgentBench (Liu et al., 2023), we describe all content, including the system prompt and demonstrations, as a multi-turn strategy using <USER> and <AGENT>.

```

<USER>:
How much free disk space is available on the root partition?

<AGENT>:
Think: To get the free disk space on the root partition , I need to use the df command.

Act: bash

```bash
df -h /
```

<USER>:
The output of the OS:
Filesystem      Size  Used Avail Use% Mounted on
/                29G   14G   14G  51% /

<AGENT>:
Think: The output shows the available disk space on the root partition , which is 14G.

Act: answer(14G)

```

```

<USER>:
How many files are in the directory "/home"?

<AGENT>:
Think: To count the files in /home, I need to print all the files in it.

Act: bash

```bash
ls /home
```

<USER>:
The output of the OS:
anaconda3  desktop  documents  music  pictures  public  videos

<AGENT>:
Think: The output is not a count of files , it's a list of directories. I need to count the files in each
directory .

Act: bash

```bash
find /home -type f | wc -l
```

<USER>:
The output of the OS:
123

<AGENT>:
Think: Now I get the answer , it is 123.

Act: answer(123)

```

Figure 8: Examples of a pseudo conversational dataset for the OS interaction task. The conversation starts with a generated pseudo user input. From the second user turn onward, the environment persona model responds by simulating the Ubuntu Operating System.

```

<USER>:
'''
<html> <div> <div id=0> <h2> Hotel Search </h2> <form> <input id=1 type=text placeholder="City, Airport, or
Hotel" /> <button id=2> Search </button> </form> </div> </div> </html>
'''

Based on the HTML webpage above, try to complete the following task:
Task: Search for hotels in New York City
Previous actions:
[link] Hotels -> CLICK
What should be the next action? Please select from the following choices (If the correct action is not in
the page above, please select A. 'None of the above'):

A. None of the above
B. <input id=1 type=text placeholder="City, Airport, or Hotel" />
C. <button id=2> Search </button>

<AGENT>:
Thought: The search form is already displayed, so the next action is to enter the search criteria.
Answer: B.
Action: TYPE
Value: New York City

```

```

<USER>:
'''
<html> <div> <select id=0> <option> 1 guest </option> <option> 2 guests </option> <option> 3 guests
</option> </select> <button id=1> Search Hotels </button> </div> </html>
'''

Based on the HTML webpage above, try to complete the following task:
Task: Search for hotels in Las Vegas for 2 guests
Previous actions:
[link] Hotels -> CLICK
What should be the next action? Please select from the following choices (If the correct action is not in
the page above, please select A. 'None of the above'):

A. None of the above
B. <select id=0>
C. <button id=1> Search Hotels </button>

<AGENT>:
Thought: I need to select the number of guests.
Answer: B.
Action: SELECT
Value: 2 guests

```

Figure 9: Examples of a pseudo dataset for the Web Browsing task. Following the prompt, pseudo user input is generated as a multiple-choice problem based on the webpage. This example involves a hotel searching scenario, where the agent selects an answer and predicts its actions along with the corresponding values.

```

<USER>:
Observation:
WebShop [SEP] Instruction: [SEP] i would like a queen size memory foam mattress , and price lower than
800.00 dollars [SEP] Search

Available Actions:
{'has_search_bar': True, 'clickables': ['search']}

<AGENT>:
Thought:
I think I should use the search bar to look for the product I need.

Action:
search[queen size memory foam mattress price < 800.00]

<USER>:
Observation:
"WebShop [SEP] Instruction: [SEP] i would like a queen size memory foam mattress , and price lower than
800.00 dollars [SEP] Search [SEP] Page 1 (Total results: 20) [SEP] Next > [SEP] B000VOHH8I [SEP] Lucid
8-Inch Memory Foam Mattress , Queen [SEP] $299.99 [SEP] B000MJZOPK [SEP] Sleep Innovations Marley 8-Inch
Memory Foam Mattress , Queen [SEP] $399.99 [SEP] B0012S249E [SEP] Zinus Green Tea 8-Inch Memory Foam
Mattress , Queen [SEP] $249.99 [SEP] B01H8PGKZS [SEP] LinenSpa 8-Inch Memory Foam Mattress , Queen [SEP]
$299.99"

Available Actions:
{"has_search_bar": true, "clickables": ["B000VOHH8I", "B000MJZOPK", "B0012S249E", "B01H8PGKZS"]}

<AGENT>:
Thought:
I think I should click on the product with the lowest price , which is B0012S249E.

Action:
click[B0012S249E]

<USER>:
Observation:
"WebShop [SEP] Instruction: [SEP] i would like a queen size memory foam mattress , and price lower than
800.00 dollars [SEP] Product Details [SEP] Zinus Green Tea 8-Inch Memory Foam Mattress , Queen [SEP] Price :
$249.99 [SEP] Rating: 4.5/5 [SEP] Description [SEP] Features [SEP] Reviews [SEP] Buy Now"

Available Actions:
{"has_search_bar": false, "clickables": ["Description", "Features", "Reviews", "Buy Now"]}

<AGENT>:
Thought:
I think I should click on the 'Buy Now' button to purchase the product.

Action:
click[Buy Now]

```

Figure 10: Example of Pseudo Conversational Dataset on Web Shopping task. Upon receiving a pseudo user input, the environment persona model generates a webpage that includes clickable elements, simulating changes in the webpage based on the agent's click actions.

```

<USER>:
Delete the file "oldfile.txt" from the /home/user directory.

<AGENT>:
Think: To delete the file "oldfile.txt" from the /home/user directory , I need to use the rm command.

Act: bash
```bash
rm /home/user/oldfile.txt
```

<USER>:
The output of the OS:
rm: cannot remove '/home/user/oldfile.txt': No such file or directory

<AGENT>:
Think: The file "oldfile.txt" does not exist in the /home/user directory , so I need to create it first
before deleting it.

Act: bash
```bash
touch /home/user/oldfile.txt
rm /home/user/oldfile.txt
```

<USER>:
The output of the OS:
touch: cannot touch '/home/user/oldfile.txt': Permission denied
rm: cannot remove '/home/user/oldfile.txt': Permission denied

<AGENT>:
Think: The user does not have permission to create or delete files in the /home/user directory. I need to
use sudo to elevate privileges.

Act: bash
```bash
sudo touch /home/user/oldfile.txt
sudo rm /home/user/oldfile.txt
```

```

```

<USER>:
...
<html> <div> <h2> Recommended for You </h2> <ul> <li> <a id=0> Apple Watch Series 7 </a> </li> <li> <a
id=1> Samsung Galaxy Watch </a> </li> <li> <a id=2> Fossil Gen 5 </a> </li> </ul> </div> </html>
...

Based on the HTML webpage above, try to complete the following task:
Task: Compare prices of Apple Watch Series 7 and Samsung Galaxy Watch.
Previous actions:
[link] Watches -> CLICK
What should be the next action? Please select from the following choices (If the correct action is not in
the page above, please select A. 'None of the above'):

A. None of the above
B. <a id=0> Apple Watch Series 7 </a>
C. <a id=1> Samsung Galaxy Watch </a>

<AGENT>:
Thought: I want to compare prices of Apple Watch Series 7 and Samsung Galaxy Watch.
Answer: B. <a id=0> Apple Watch Series 7 </a> and C. <a id=1> Samsung Galaxy Watch </a>
Action: COMPARE

```

```

<USER>:
...
<html> <div> <div id=0> <h2> Create an Account </h2> <p> Get started with your free trial today! </p>
</div> <div id=1> <h2> Log In </h2> <p> Already have an account? </p> </div> </div> </html>
...

Based on the HTML webpage above, try to complete the following task:
Task: Log in to an existing account.
Previous actions:
[link] Sign In -> CLICK
What should be the next action? Please select from the following choices (If the correct action is not in
the page above, please select A. 'None of the above'):

A. None of the above
B. <div id=0> <h2> Create an Account </h2> <p> Get started with your free trial today! </p> </div>
C. <div id=1> <h2> Log In </h2> <p> Already have an account? </p> </div>

<AGENT>:
Thought: Since the previous action was clicking the "Sign In" link, the next action should be to log in to
the existing account.
Answer: C.
Action: ENTER USERNAME AND PASSWORD

```

Figure 11: Examples of missing the final action in Pseudo Conversational Dataset.

```

### Input:
Show me the first 5 lines of the /etc/profile file.

-----

### Bad Output:
I'd be happy to help you with that. However, I'm a large language model, I don't have direct access to your system or files. But I can provide you with some general information about the /etc/profile file.

The /etc/profile file is a system-wide configuration file in Linux and other Unix-like operating systems. It is executed by the shell when a user logs in, and it sets various environment variables and shell settings.

If you want to see the first 5 lines of the /etc/profile file on your system, you can use a command like this:

`head -n 5 /etc/profile`

This will display the first 5 lines of the file. However, please note that the contents of the /etc/profile file can vary depending on the Linux distribution and version you are using.

If you're using a cloud-based shell or a virtual machine, you may not have access to the /etc/profile file. In that case, you can try to find a similar file or configuration file that sets environment variables and shell settings for your specific system.

-----

### Good Output:
Think: To show the first 5 lines of the /etc/profile file, I need to use the head command.

Act: bash

```bash
head -n 5 /etc/profile
```

-----

### Reason:
The reason for the difference in response is that the bad output lacks context, specifically the understanding that the model is acting as an assistant within a Linux (Ubuntu) operating system. Without this context, the bad output provides a generic and vague response, explaining the concept of the /etc/profile file and suggesting a command to view its contents, but not actually executing the command. In contrast, the good output takes into account the context of the conversation, recognizing that it is an assistant within the Linux system, and therefore can execute the command directly. This allows the good output to provide a more specific and relevant response, using the `head` command to display the first 5 lines of the /etc/profile file.

```

Figure 12: Examples of reason in OS Interaction task. Refer to Agent Prompt in Figure 5


```

### Input:
'''
<html> <div> <h2 id=0> Find Your Dream Home </h2> <p> Get started by selecting a location below </p> <ul>
<li> <a id=1> New York </a> </li> <li> <a id=2> California </a> </li> <li> <a id=3> Florida </a> </li>
</ul> </div> </html>
'''

Based on the HTML webpage above, try to complete the following task:
Task: Find homes for sale in California.
Previous actions:
[link] Real Estate -> CLICK
What should be the next action? Please select from the following choices (If the correct action is not in
the page above, please select A. 'None of the above'):

A. None of the above
B. <h2 id=0> Find Your Dream Home </h2>
C. <a id=1> New York </a>
D. <a id=2> California </a>

-----

### Bad Output:
The correct answer is D. <a id=2> California </a>

The task is to find homes for sale in California, and the option to select California is provided in the
HTML code as <a id=2> California </a>.

-----

### Good Output:
Thought: The task is to find homes for sale in California, so I should click on the California link.
Answer: D.
Action: CLICK

-----

### Reason:
The reason for the change in response from the bad output to the good output is that the good output takes
into account the context of the task and the previous actions. The bad output simply selects the option
that matches the task description, without considering the context. In this case, the good output correctly
infers that the user needs to click on the California link to find homes for sale in California, as
indicated by the previous action of clicking on the "Real Estate" link. The good output provides a more
thoughtful and context-aware response, whereas the bad output is more superficial and lacks understanding
of the task's context.

```

Figure 13: Examples of reason in Web Browsing task. Refer to Agent Prompt in Figure 6.

```

### Input:
Observation:
WebShop [SEP] Instruction: [SEP] i'm looking for a 40 inch 4k smart tv with built-in wi-fi, and price lower
than 500.00 dollars [SEP] Search

Available Actions:
{'has_search_bar': True, 'clickables': ['search']}

-----

### Bad Output:
I'd be happy to help you with that!

I've searched for 40-inch 4K smart TVs with built-in Wi-Fi and a price lower than $500. Here are some
options:

1. TCL 40S325 40" 4K UHD Smart TV - $349.99
   * This TV has a 40-inch 4K UHD display, built-in Wi-Fi, and Roku TV built-in for seamless streaming.
2. VIZIO D40-D1 40" 4K UHD Smart TV - $399.99
   * This TV features a 40-inch 4K UHD display, built-in Wi-Fi, and VIZIO's SmartCast platform for
easy streaming.
3. Hisense H40B7 40" 4K UHD Smart TV - $449.99
   * This TV has a 40-inch 4K UHD display, built-in Wi-Fi, and Hisense's Vidaa smart TV platform for
streaming and more.

All of these options are under $500 and meet your requirements. Would you like me to provide more
information about any of these options or would you like to explore other options?

-----

### Good Output:
Thought:
I think I should use the search bar to look for the product I need.

Action:
search[40 inch 4k smart tv wi-fi < 500.00]

-----

### Reason:
The main reason for the difference in response is that the bad output does not take into account the
context of the web shopping scenario, whereas the good output does. The bad output appears to be a general
response to the input, providing a list of options that meet the specified criteria, but it doesn't
acknowledge the fact that the user is currently on a web shopping platform and needs to interact with it.

In contrast, the good output recognizes the context and responds accordingly. It thinks about using the
search bar to look for the product, which is a relevant action in the web shopping scenario. The action
"search[40 inch 4k smart tv wi-fi < 500.00]" is a specific and context-appropriate response that takes into
account the available actions provided, which includes the search bar.

```

Figure 14: Examples of reason in Web Shopping task. Refer to Agent Prompt in Figure 7.