

# Robustness of Prompting: Enhancing Robustness of Large Language Models Against Prompting Attacks

Lin Mu<sup>1</sup>, Guowei Chu<sup>1</sup>, Li Ni<sup>1</sup>, Lei Sang<sup>1</sup>, Zhize Wu<sup>2</sup>,  
Peiquan Jin<sup>3</sup>, Yiwen Zhang<sup>1\*</sup>

<sup>1</sup>Anhui University, <sup>2</sup>Hefei University,

<sup>3</sup>University of Science and Technology of China,

{mulin, nili, sanglei, zhangyiwen}@ahu.edu.cn chuguowei@std.ahu.edu.cn  
wuzz@hfuu.edu.cn jpq@ustc.edu.cn

## Abstract

Large Language Models (LLMs) have demonstrated remarkable performance across various tasks by effectively utilizing a prompting strategy. However, they are highly sensitive to input perturbations, such as typographical errors or slight character order errors, which can substantially degrade their performance. Despite advances in prompting techniques, developing a prompting strategy that explicitly mitigates the negative impact of such perturbations remains an open challenge. To bridge this gap, we propose **Robustness of Prompting (RoP)**, a novel prompting strategy specifically designed to enhance the robustness of LLMs. RoP consists of two stages: *Error Correction* and *Guidance*. In the *Error Correction* stage, RoP applies diverse perturbation methods to generate adversarial examples, which are then used to construct prompts that automatically correct input errors. In the *Guidance* stage, RoP generates an optimal guidance prompting based on the corrected input, steering the model toward more robust and accurate inferences. Through comprehensive experiments spanning arithmetic, commonsense, and logical reasoning tasks, we demonstrate that RoP significantly improves LLMs' robustness against adversarial perturbations. Notably, it maintains model accuracy with only minimal degradation compared to clean input scenarios, thereby establishing RoP as a practical and effective approach for enhancing LLM robustness in real-world applications.

## 1 Introduction

Large Language Models (LLMs)(Brown et al., 2020; Chowdhery et al., 2023) have demonstrated remarkable capabilities across a wide range of tasks (Wu et al., 2024; Mu et al., 2025). As the size of LLMs has increased, their in-context learning (Brown et al., 2020) abilities have emerged.

\*Corresponding author

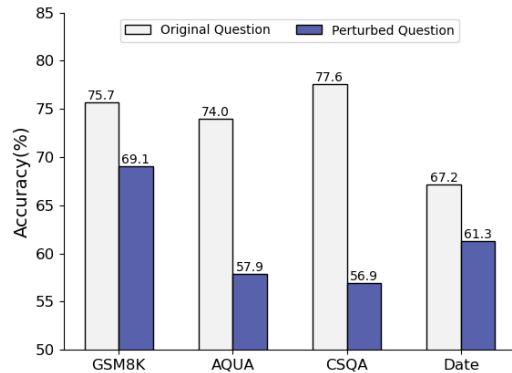


Figure 1: Comparison of the performance of GPT-3.5-Turbo before and after character-level perturbation in multiple reasoning tasks.

This allows them to observe the provided prompting, which includes task-related examples or instructions, to solve the given input question without requiring any updates to their parameters. The design of prompting is crucial, as it can significantly influence the performance of the LLMs in solving the given question.

Recent studies (Wang et al., 2025) have explored various prompting strategies to further improve LLM's performance. One notable method is Chain-of-Thought (CoT) prompting (Wei et al., 2022), which provides step-by-step reasoning examples to enhance the performance of LLMs. However, CoT prompting requires careful manual design, which can be resource-intensive. In contrast, the Automatic Prompt Engineer (APE) method (Zhou et al., 2023) proposes an automatic generation and selection of the prompting, eliminating manual intervention. Despite these advancements, a critical limitation persists: LLMs are highly sensitive to perturbations in input content (Gan et al., 2024). Even minor typographical mistakes or subtle character-level alterations—for instance, changing "buy" to "boy" or "third" to "thidr" in the input question—can sig-

nificantly degrade their performance. This fragility is illustrated in Figure 1, which shows a sharp decline in the inference accuracy of GPT-3.5-Turbo under minimal character-level perturbations across multiple reasoning tasks. While existing research highlights the sensitivity of LLMs to input content (Xu et al., 2023; Zhu et al., 2023), there remains a gap in understanding how to design a prompting strategy that enhances the robustness of these models against such perturbations.

To enhance the robustness of models, a widely used strategy is data augmentation (Wang et al., 2022). This involves perturbing input texts and incorporating them into the training dataset to fine-tune the model. While data augmentation can help improve robustness, it requires updating model parameters, a process that can be both computationally expensive and resource-intensive. To efficiently tackle this challenge, we propose a novel prompting strategy called **Robustness of Prompting (RoP)**, which aims at enhancing the robustness of LLMs without requiring updating their parameters. RoP introduces a two-stage pipeline comprising *Error Correction* and *Guidance*: In the *Error Correction* stage, we generate a set of adversarial examples by applying various perturbation methods to input questions. These are paired with their corresponding unperturbed versions to automatically generate prompting that teaches the LLMs to restore the original intended from perturbed inputs. In the *Guidance* stage, the corrected input question is used to automatically generate an optimal guidance prompting that steer the LLM’s inference process, ensuring robust inference despite earlier input noise.

We evaluate RoP across three diverse reasoning task categories: arithmetic, commonsense, and logical. Experimental results demonstrate that while baseline large language models (LLMs) experience significant performance degradation under perturbed input conditions, RoP effectively mitigates this drop. In particular, when inputs are perturbed through the insertion of irrelevant or distracting information, RoP improves accuracy by an average of 16.1% over baseline methods. This substantial improvement underscores the robustness of RoP in preserving inference quality under adversarial conditions.

## 2 Related Works

### 2.1 Automatic Prompting

Large Language Models (LLMs) (Brown et al., 2020; Chowdhery et al., 2023) have demonstrated substantial improvements in their ability to perform natural language reasoning tasks through a method known as prompting (Brown et al., 2020; Min et al., 2022). Prompting refers to the technique of providing guidance to the LLMs, typically in the form of instructions or input-output pairs, that directs the model on how to solve a given task. To further boost LLM performance, Chain-of-Thought (CoT) prompting was introduced (Wei et al., 2022). This method includes intermediate reasoning steps within the prompt to guide the LLM through complex tasks. Subsequently, the concept of Zero-Shot-CoT was proposed (Kojima et al., 2022), where LLMs were able to perform reasoning tasks with minimal prior examples, by simply appending the trigger sentence *Let’s think step by step* to each question.

While these methods significantly improved LLM reasoning, they require pre-designed prompt settings, which may lead to suboptimal performance for certain tasks. In response, a growing body of research has focused on automating the process of prompt generation. The Automatic Prompt Engineer (APE) (Zhou et al., 2023) provides a promising solution by automatically generating prompts. It first creates a set of candidate instructions and then selects the best-performing one. Similarly, Optimization by PROmpting (OPRO) (Yang et al., 2024) introduced an iterative optimization method that refines the prompt over multiple steps. In each iteration, the LLM generates new candidate solutions based on the current prompt, incorporating previously generated solutions and their corresponding evaluation values. These refined solutions are then used to further optimize the prompt.

### 2.2 Adversarial Attack

Despite the advancements in automatic prompting techniques, LLMs remain highly sensitive to the input text. Even the smallest perturbation in the input can lead to a substantial degradation in their performance. For example, research by (Gan et al., 2024) demonstrated that a single character-level perturbation in an input question can significantly affect inference accuracy. As reported in their study, the ac-

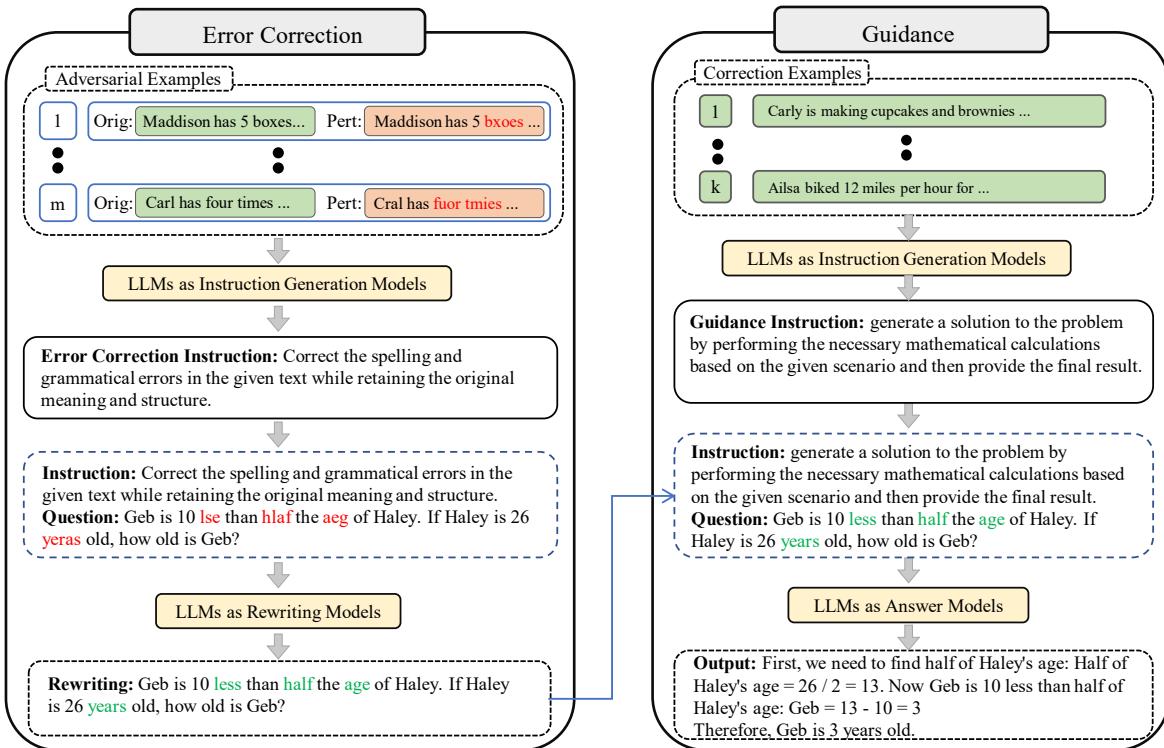


Figure 2: The framework of Robustness of Prompting (RoP).

curacy of the Mistral-7B-Instructs model \* dropped by 5% on the GSM8K (Cobbe et al., 2021) dataset due to such perturbations. Moreover, recent work by (Xu et al., 2023) explored adversarial attacks targeting the input, further confirming LLMs' vulnerability to even minor changes in the text. These perturbations can range from character-level errors to more complex changes in words, sentences, or semantics. For instance, (Zhu et al., 2023) proposed methods for adversarial attacks at multiple levels: character, word, sentence, and semantic, highlighting the extent of the issue across different types of input modifications.

While existing research has examined on various adversarial attack techniques, there remains a significant gap in efforts to enhance the robustness of LLMs against such perturbed inputs. The challenge of designing effective robustness prompting strategies—particularly in response to perturbed text—has not been thoroughly explored. Therefore, this paper aims to investigate methods for enhancing the robustness of LLMs against question-based perturbations.

### 3 Method

LLMs are highly sensitive to input perturbations, where even minor perturbations in the question can significantly impact their output. Given these vulnerabilities, we propose the Robustness of Prompting (RoP) strategy to enhance the robustness of LLM responses. The overall framework is illustrated in Figure 2.

#### 3.1 Background

The paradigm of in-context learning (Dong et al., 2022) allows LLMs to perform tasks by using only a few examples and an instruction within a prompting. Formally, given a question  $x$  and a set of candidate answers  $Y = \{y_1, \dots, y_n\}$ , an LLM, denoted as  $L_\theta$  and parameterized by  $\theta$ , predicts the answer that maximizes the conditional probability among the candidate answers based on the provided prompting  $P$ . The prompting  $P$  consists of an instruction and  $k$  examples, represented as  $P = \langle in, \{e(x_1, y_1), \dots, e(x_k, y_k)\} \rangle$ , where  $in$  denotes the description and requirements of the task and  $e(x_i, y_i)$  denotes an example expressed in natural language. Here,  $x_i$  denotes the demonstration question, and  $y_i$  is its corresponding answer. Accordingly, the in-context learning formulation

\*<https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2>

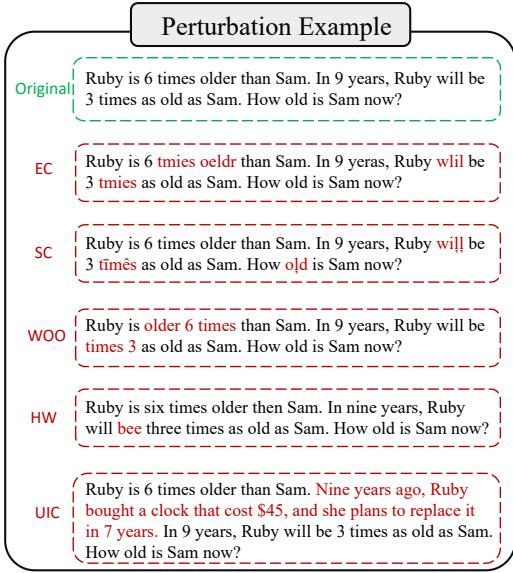


Figure 3: Five different perturbation methods.

can be expressed as follows:

$$\begin{aligned}\hat{y} &= \arg \max_{y_i \in Y} L_\theta(y | x, P) \\ &= \arg \max_{y_i \in Y} L_\theta(y | x, in, e(x_1, y_1), \dots, e(x_k, y_k))\end{aligned}\quad (1)$$

While in-context learning enables LLMs to perform question based prompting, this framework assumes ideal conditions. However, input noise and inconsistencies are inevitable, necessitating robust mechanisms to preserve model reliability under perturbation.

### 3.2 Perturbation Methods

To evaluate and enhance the robustness of LLMs, we introduce five distinct types of perturbations that simulate common errors in real-world inputs. These perturbations serve dual purposes: assessing model fragility and constructing adversarial examples that facilitate error correction instruction generating. The five designed perturbation types are as follows:

- **Error Character (EC):** This type of perturbation involves randomly shuffling the internal characters of words or changing characters to other (e.g., *times* → *tmies*, *will* → *wil*).
- **Similar Character (SC):** This type of perturbation involves replacing one or more characters with visually similar symbols or special characters (e.g., *will* → *will*, *times* → *t m s*).

- **Words Out of Order (WOO):** This type of perturbation involves rearranging neighboring word positions to disrupt syntactic structure (e.g., *6 times older* → *older 6 times*, *3 times* → *times 3*).

- **Homophone Words (HW):** This type of perturbation involves replacing original terms with phonetically equivalent alternatives (e.g., *be* → *bee*).

- **Unaffected Interference Conditions (UIC):** This type of perturbation involves appending irrelevant but plausible information that does not alter the answer (e.g., *Ruby is 6 times older than Sam.* → *Ruby is 6 times older than Sam. Nine years ago, Ruby bought a clock that cost \$45, and she plans to replace it in 7 years.*).

To automate the generation of perturbed questions  $\hat{x}$  from an original question  $x$  and its answer  $y$ , we employ the LLM itself as a perturbation model. Given a perturbation type  $pt$ , we define a perturbation prompting  $ap$  as follows:

---

*Your objective is to rewrite a given math question...*  
*Perturbation type: pt*  
*The given question: {x}*  
*Answer of the given question: {y}*  
*Please rewrite the question using the specified perturbation strategy ...*

---

With the perturbation prompting, we can generate a perturbation question  $\hat{x}$  based on  $x$ . Figure 3 shows the five different perturbation methods based on a given question.

### 3.3 Error Correction

Building on the adversarial examples generated via perturbation, the first stage of RoP—Error Correction—is designed to recover the intended semantics of flawed inputs through correctional prompting.

**Adversarial sample Generation:** Inspired by adversarial attack in other fields (Wang et al., 2022, 2023), we synthesize adversarial examples from perturbed inputs. Specifically, we sample  $k$  questions from the training dataset and apply aforementioned perturbation methods to obtain pairs  $\langle x_i, \hat{x}_i \rangle \in \mathcal{D}_{adv}$ , where  $x_i$  is the origin question, and  $\hat{x}_i$  its perturbed variant.

**Error Correction Instruction:** The next step is to convert these adversarial examples into a prompt-based correction task. To this end, we

sample a subset of  $m$  such examples, denoted as  $\hat{\mathcal{D}}_{adv} \subseteq \mathcal{D}_{adv}$ , and use the APE framework (Zhou et al., 2023) to automatically generate an instruction prompt  $in_{ec}$ . This prompt is crafted to instruct the LLM to recognize and correct perturbations in input questions while preserving the answer. The final error-corrected prompting is thus defined as:  $P = < in_{ec}, \hat{\mathcal{D}}_{adv} >$ , which is passed to the LLM along with a perturbed question  $\hat{x}$ . The model then outputs a corrected version  $\hat{x}_{ec}$  that aligns semantically with the original input  $x$ .

**Execute Correction:** The instruction  $in_{ec}$  and the adversarial subset  $\hat{\mathcal{D}}_{adv}$  are provided to an LLM, such as GPT-4o. The model utilizes the prompt  $P = < in_{ec}, \hat{\mathcal{D}}_{adv} >$  to rewrite an erroneous input  $x$ , producing a corrected version  $\hat{x}_{ec}$  that restores the original semantics despite the presence of input perturbations.

### 3.4 Guidance Instruction Generation

In the previous stage, we generate corrected questions based on the error correction instruction  $in_{ec}$ . To automatically generate optimal prompts for these corrected questions, we use APE (Zhou et al., 2023) to generate optimized instructions  $in_{opt}$ . These instructions guide the LLMs in producing robust responses using adversarial samples. Specifically, we randomly sample  $k$  question-answer pairs  $< \hat{x}_{ec}^i, a^i > \in \mathcal{D}_{cor}$ . The guidance instruction  $in_{opt}$  is generated using APE. Finally, given a question  $x_{ec}$ , the answer  $\hat{y}_{ec}$  is generated:

$$\begin{aligned}\hat{y}_{ec} &= \arg \max_{y_i \in Y} L_\theta(y | x_{ec}, in_{opt}, \mathcal{D}_{cor}) \\ &= \arg \max_{y_i \in Y} L_\theta(y | x_{ec}, in_{opt}, < \hat{x}_{ec}^1, a^1 > \\ &\quad \dots < \hat{x}_{ec}^k, a^k >)\end{aligned}\tag{2}$$

## 4 Experiment and Analysis

### 4.1 Arithmetic Reasoning

#### 4.1.1 Experiment Setup

In this section, we evaluate the effectiveness and performance of our method, RoP, across six arithmetic reasoning datasets. These datasets are: (1) GSM8K (Cobbe et al., 2021), (2) AQUA (Ling et al., 2017), (3) SingleEq (Single Equation) (Koncel-Kedziorski et al., 2015), (4) SVAMP (Simple Variations on Arithmetic Math Word Problems) (Patel et al., 2021), (5) Multi-Arith (Roy and Roth, 2015), (6) AddSub (Hosseini

et al., 2014). Detailed descriptions can be found in the Appendix A.

To evaluate the robustness of RoP against question perturbations, we apply five types of input perturbations described in Section 3.2 to the test dataset. Specifically, we use GPT-4o (Hurst et al., 2024) to generate the perturbed data, along with the error correction instructions and the guidance instructions required for the experiment. For the evaluation phase, we primarily employ GPT-3.5-turbo (Ouyang et al., 2022) to assess the performance of each method across various datasets. Additionally, to further evaluate the scalability of RoP, we conducted experiments using three more evaluation models: GPT-4o, o1-mini (Jaech et al., 2024), and o3-mini (OpenAI, 2025) <sup>†</sup>.

**Baseline:** We compare our method with the following methods: (1) Standard Prompting (Stand): The original unperturbed or perturbed question is directly fed into the model without any enhancements. (2) Chain of Thought (CoT) (Wei et al., 2022): Prompting includes exemplars of intermediate reasoning steps to improve model inference on multi-step problems. (3) Automatic Prompt Engineer (APE) (Zhou et al., 2023): An automated method that generates high-quality task instructions via prompt selection and optimization. (4) PromptAgent (Wang et al., 2024): A state-of-the-art agent-based prompt optimization framework that autonomously constructs expert-level prompts through iterative refinement.

### 4.1.2 Result

In this section, we compare the RoP method with the baseline approaches—Stand, CoT, APE, and PromptAgent—across six arithmetic reasoning datasets under various textual perturbations. First, we verify the inference accuracy of LLMs using Equation 1, applied to the original, unperturbed datasets. Then, we evaluate the inference accuracy of LLMs after applying each baseline method to the perturbed datasets. The result, summarized in Table 1. The findings highlight that RoP demonstrates superior robustness across most perturbation scenarios. Notably, for complex arithmetic reasoning tasks such as AQUA, GSM8K, and SVAMP, RoP maintains a relatively high level of accuracy, with only marginal reductions in performance following perturbation. This contrasts with baseline methods, which show more significant drops in accuracy.

<sup>†</sup><https://openai.com>

Method		SingleEq	AddSub	MultiArith	AQUA	GSM8K	SVAMP	Avg.
No Dert.	Stand	95.7	90.4	90.5	74.0	75.7	79.4	84.3
EC	Stand	89.2	86.8	87.0	57.9	69.0	75.0	77.5
	CoT	90.9	85.3	<b>96.3</b>	57.9	70.2	<b>80.3</b>	80.2
	APE	90.9	76.7	91.2	57.9	66.9	76.5	76.7
	PromptAgent	92.5	80.8	93.5	56.3	76.5	76.5	79.3
	<b>RoP (Our)</b>	<b>95.3</b>	<b>87.3</b>	95.0	<b>63.0</b>	<b>72.9</b>	79.7	<b>82.2</b>
SC	Stand	88.0	88.1	85.3	52.4	68.0	74.0	76.0
	CoT	88.4	86.3	<b>94.2</b>	65.8	74.2	76.2	80.8
	APE	92.7	<b>90.6</b>	86.2	52.8	71.4	76.2	78.3
	PromptAgent	92.5	46.3	85.0	58.7	67.3	77.4	71.2
	<b>RoP (Our)</b>	<b>95.1</b>	82.5	92.3	<b>70.1</b>	<b>75.4</b>	<b>78.3</b>	<b>82.3</b>
WOO	Stand	88.8	88.6	86.8	59.5	70.1	76.9	78.4
	CoT	88.2	86.3	<b>96.5</b>	56.7	<b>73.2</b>	78.2	79.9
	APE	<b>92.3</b>	<b>91.7</b>	88.0	60.6	71.3	72.7	79.4
	PromptAgent	85.6	83.5	88.8	57.1	71.4	76.8	77.2
	<b>RoP (Our)</b>	89.0	84.6	91.3	<b>65.8</b>	71.3	<b>79.5</b>	<b>80.2</b>
HW	Stand	84.1	82.0	78.7	56.3	66.4	65.9	72.2
	CoT	87.4	84.3	92.7	63.4	70.4	75.8	79.0
	APE	83.1	82.8	89.3	59.8	68.5	77.8	76.9
	PromptAgent	87.8	83.3	82.8	58.3	66.5	75.5	75.7
	<b>RoP (Our)</b>	<b>90.9</b>	<b>84.3</b>	<b>94.0</b>	<b>64.6</b>	<b>74.2</b>	<b>82.4</b>	<b>81.7</b>
UIC	Stand	63.6	72.2	54.5	57.1	51.9	54.2	58.9
	CoT	62.2	71.7	59.3	50.0	51.4	55.7	58.4
	APE	78.7	69.9	59.0	56.3	45.0	65.5	62.4
	PromptAgent	57.3	57.0	50.5	48.4	49.8	49.2	52.0
	<b>RoP (Our)</b>	<b>79.1</b>	<b>89.9</b>	<b>74.7</b>	<b>62.2</b>	<b>61.7</b>	<b>76.7</b>	<b>74.0</b>

Table 1: Accuracy(%) of arithmetic reasoning tasks.

In particular, the UIC perturbation posed a considerable challenge to the LLMs. The inference accuracy dropped from 84.3% on the clean dataset to 58.9% post-perturbation, a decrease of 25.4%. While traditional baseline approaches failed to mitigate this drop effectively, the RoP method limited the performance degradation to just 10.3%, thereby illustrating its capacity to enhance model robustness under semantic perturbations. However, under the WOO perturbation scenario, the advantage of RoP was less pronounced. In this case, the results are similar to those of CoT and APE. This outcome is likely due to the inherent ability of LLMs to handle minor syntactic reordering, given their training on diverse and syntactically varied corpora.

#### 4.1.3 Ablation Study

To systematically evaluate the impact of *Error Correction* and *Guidance* components of RoP, we con-

Method		AQUA	GSM8K	SingleEq
No Dert.	Stand	74.0	75.7	95.7
EC	Stand	57.9	69.1	89.2
	CO	61.0	73.8	94.7
	GO	57.9	66.9	90.9
	<b>RoP</b>	<b>63.0</b>	<b>73.9</b>	<b>95.3</b>
HW	Stand	56.3	66.4	84.1
	CO	59.1	73.2	87.4
	GO	59.8	68.5	83.1
	<b>RoP</b>	<b>64.6</b>	<b>74.2</b>	<b>90.9</b>

Table 2: Accuracy(%) of different setting.

ducted an ablation study. This study compares two settings: (1) Correction Only (CO), where only the *Error Correction* module is used without the *Guidance* module. (2) Guidance Only (GO), where

Method	GPT-4o	o1-mini	o3-mini
No Dert.	Stand	79.5	83.5
	Stand	79.1	80.7
	APE	80.3	79.5
EC	<b>RoP</b>	<b>81.5</b>	<b>82.7</b>
	Stand	78.4	81.9
	APE	<b>79.9</b>	72.4
WOO	<b>RoP</b>	79.1	<b>86.2</b>
	Stand	79.1	<b>87.8</b>
	APE	79.1	72.8

Table 3: Accuracy(%) of different LLM on AQUA dataset.

only the *Guidance* module is used without the *Error Correction* (essentially the APE method). For this experiment, GPT-3.5-Turbo was used as the evaluation model.

As shown in Table 2, we present the experiment results of two perturbation methods (EC and HW), applied to the AQUA and GSM8K. Detailed results can be found in Table 4. The findings reveal that using only the *Error Correction* module results in a smaller decrease in LLM’s performance, suggesting that *Error Correction* alone can effectively enhance the robustness of the LLMs. Similarly, the *Guidance* module also enhances the robustness of the LLMs when used alone. Notably, the combined use of both components (RoP) consistently achieved superior performance across both datasets and perturbation types. This demonstrates that the synergy between *Error Correction* and *Guidance* significantly amplifies the robustness of LLMs when dealing with input perturbations. This underscores the necessity of both components working together to optimize LLMs’ performance under perturbed conditions.

#### 4.1.4 Performance on Other Models

To further explore the scalability and cross-model generalizability of the RoP, we conducted a comprehensive evaluation using the AQUA dataset and a variety of LLMs. This investigation sought to determine whether RoP maintains its robustness-enhancing properties when applied to different model architectures and capacities. We tested RoP on more advanced and lightweight models, including GPT-4o, o1-mini, and o3-mini. Notably, GPT-4o was consistently employed as the optimizer model, ensuring consistency in optimization dynamics across experimental conditions.

The results of these experiments, which used various LLMs and two perturbation methods, are

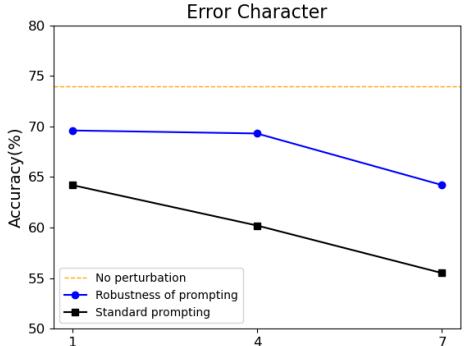


Figure 4: The effectiveness of different levels of EC perturbation.

presented in Table 3. More detailed experimental results can be found in Table 5 in Appendix C. The result shown that RoP demonstrated strong transferability, consistently mitigating performance degradation caused by perturbations across all evaluated LLMs

#### 4.1.5 Robustness Across Perturbation Levels

To assess the granularity of RoP’s robustness under increasing textual interference, we conducted a controlled experiment wherein the number of perturbed characters in each input question was systematically varied—specifically at 1, 4, and 7-character perturbation levels. This test evaluates the model’s ability to maintain inference accuracy as perturbation severity increases. As shown in Figure 4, RoP consistently improves accuracy across all levels of perturbation. Although performance degradation is observed as perturbation intensity increases (i.e., from 1 to 7 characters), the decline in accuracy for RoP is significantly less steep than that of unoptimized prompting strategies. This indicates that RoP effectively maintains strong performance, even when faced with various forms of textual interference. These findings underscore the gradual robustness of RoP, confirming that while no method remains entirely immune to aggressive perturbations, RoP degrades more gracefully, sustaining competitive accuracy under both mild and severe distortions. This robustness to variable perturbation intensity is a critical feature for real-world deployment in noisy or adversarial environments.

### 4.2 Other Reasoning

#### 4.2.1 Experiment Setup

To further evaluate the generalization capacity of the RoP, we extended our experiments to include

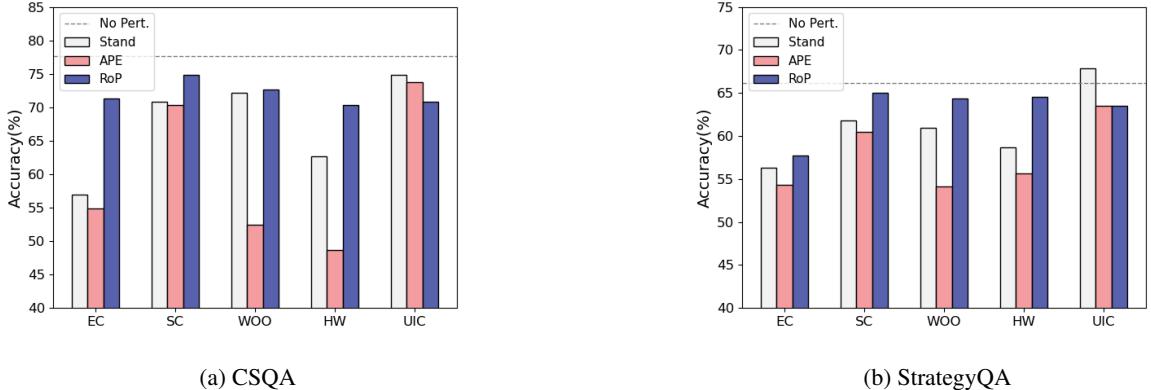


Figure 5: The performance of RoP with different perturbation methods in the commonsense reasoning task.

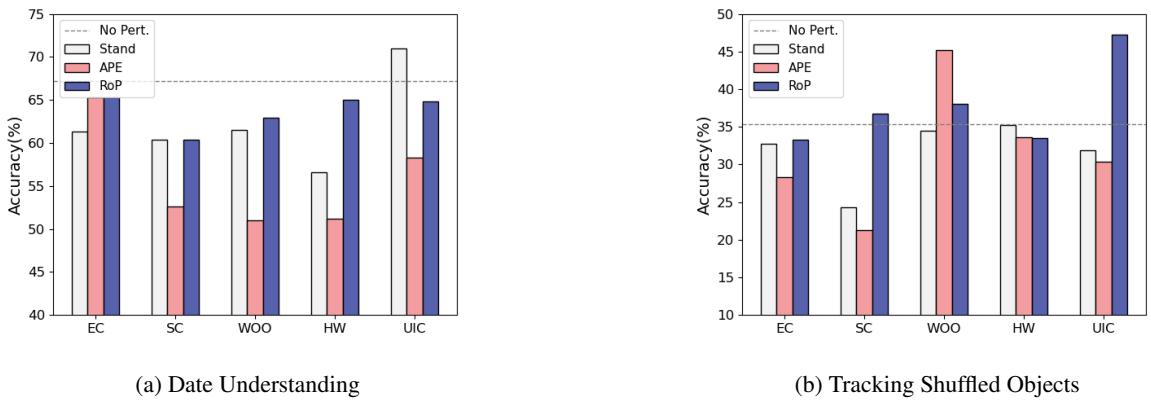


Figure 6: The performance of RoP with different perturbation methods in the logical reasoning task.

tasks in **Commonsense Reasoning** and **Logical Reasoning**. Specifically, we use two commonsense reasoning benchmarks: CommonsenseQA (CSQA) (Talmor et al., 2019) and StrategyQA (Geva et al., 2021), along with two logical reasoning tasks from BIG-bench (Srivastava et al., 2022): Date Understanding and Tracking Shuffled Objects. For these experiments, perturbations were applied to the test datasets by the protocol described in Section 3.2. Perturbed inputs, error correction instructions, and guidance prompts were generated using GPT-4o (Hurst et al., 2024), while performance evaluations were conducted using GPT-3.5-Turbo (Ouyang et al., 2022).

#### 4.2.2 Result

As illustrated in Figure 5 and 6, RoP consistently outperforms both Stand and the APE method across nearly all datasets and perturbation types, demonstrating its robustness and domain-transferability. On CSQA dataset under EC perturbation, RoP achieves 71.3% accuracy, outperforming APE

(54.9%) by +16.4% and Stand (56.9%) by +14.3%. On Tracking Shuffled Objects under the UIC perturbation, RoP reaches 47.3% accuracy, significantly surpassing APE (30.4%) by +16.9% and Stand (31.9%) by +15.5%.

These results confirm RoP’s robust performance in other tasks, affirming its utility as a general-purpose prompting for LLMs operating in adversarial contexts. An intriguing secondary finding emerged on StrategyQA and Date Understanding, where the inclusion of benign (irrelevant but non-adversarial) perturbations slightly improved model accuracy. This suggests that introducing peripheral or topic-adjacent noise may stimulate broader semantic activation in LLMs, thereby enhancing reasoning capacity under certain conditions.

### 5 Conclusion

In this paper, we propose Robustness of Prompting (RoP), which is designed to enhance the robustness of LLMs. RoP consists of two stages: First is the *Error Correction* stage, RoP generates adversarial

examples by applying diverse perturbation methods to construct prompting that automatically corrects the errors. Second is the *Guidance* stage, RoP generates an optimal guidance prompting based on the corrected input, steering the model’s inference process effectively. Our experiments demonstrate that RoP significantly improves LLMs’ robustness against adversarial perturbations, resulting in only minor performance degradation compared to unperturbed inputs.

## Limitation

While the Robustness of Prompting (RoP) framework significantly enhances the robustness of LLMs to structured input perturbations, several important limitations remain that warrant further investigation. First, the current robustness evaluation is grounded in a closed set of five predefined perturbation types (e.g., Error Character, Homophone Words). Although these categories capture a meaningful spectrum of textual noise, they do not fully encapsulate the complexity and variability of naturally occurring errors. For example, grammatical inconsistencies, syntactic ambiguities, or pragmatic shifts in discourse—common in user-generated content—are not explicitly addressed within the current perturbation taxonomy. Second, RoP relies heavily on synthetically generated perturbations, which, while controlled and reproducible, may fail to reflect the messiness and diversity of real-world linguistic inputs. Authentic perturbations often arise from colloquial speech, dialectal variation, slang, code-switching, or multilingual phrasing—phenomena that challenge the generalization capabilities of LLMs but are underrepresented in synthetic benchmarks.

## References

- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. [Language models are few-shot learners](#). *CoRR*, abs/2005.14165.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.
- Esther Gan, Yiran Zhao, Liying Cheng, Yancan Mao, Anirudh Goyal, Kenji Kawaguchi, Min-Yen Kan, and Michael Shieh. 2024. Reasoning robustness of llms to adversarial typographical errors. *arXiv preprint arXiv:2411.05345*.
- Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. [Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies](#). *Transactions of the Association for Computational Linguistics*, 9:346–361.
- Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. [Learning to solve arithmetic word problems with verb categorization](#). In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, Doha, Qatar. Association for Computational Linguistics.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.
- Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. [Parsing algebraic word problems into equations](#). *Transactions of the Association for Computational Linguistics*, 3:585–597.
- Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. [Program induction by rationale generation: Learning to solve and explain algebraic word problems](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*

- (Volume 1: Long Papers), pages 158–167, Vancouver, Canada. Association for Computational Linguistics.
- Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*.
- Lin Mu, Yide Cheng, Jun Shen, Yiwen Zhang, and Hong Zhong. 2025. Netprompt: Neural network prompting enhances event extraction in large language models. *IEEE Transactions on Big Data*, pages 1–15.
- OpenAI. 2025. Openai o3 and o4-mini system card.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.
- Arkil Patel, Satwik Bhattacharya, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094. Association for Computational Linguistics.
- Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.
- Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.
- Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.
- Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric Xing, and Zhiting Hu. 2024. Promptagent: Strategic planning with language models enables expert-level prompt optimization. In *The Twelfth International Conference on Learning Representations*.
- Xuezhi Wang, Haohan Wang, and Dyi Yang. 2022. Measure and improve robustness in NLP models: A survey. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4569–4586, Seattle, United States. Association for Computational Linguistics.
- Yaojing Wang, Shengqiong Wu, Yuecheng Zhang, Shuicheng Yan, Ziwei Liu, Jiebo Luo, and Hao Fei. 2025. Multimodal chain-of-thought reasoning: A comprehensive survey. *arXiv preprint arXiv:2503.12605*.
- Yulong Wang, Tong Sun, Shenghong Li, Xin Yuan, Wei Ni, Ekram Hossain, and H Vincent Poor. 2023. Adversarial attacks and defenses in machine learning-empowered communication systems and networks: A contemporary survey. *IEEE Communications Surveys & Tutorials*, 25(4):2245–2298.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed 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*, volume 35, pages 24824–24837. Curran Associates, Inc.
- Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. 2024. A survey on large language models for recommendation. *World Wide Web*, 27(5):60.
- Xilie Xu, Keyi Kong, Ning Liu, Lizhen Cui, Di Wang, Jingfeng Zhang, and Mohan Kankanhalli. 2023. An llm can fool itself: A prompt-based adversarial attack. *arXiv preprint arXiv:2310.13345*.
- Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*.
- Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*.
- Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Gong, et al. 2023. Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts. In *Proceedings of the 1st ACM Workshop on Large AI Systems and Models with Privacy and Safety Analysis*, pages 57–68.

## A Dataset

**Arithmetic Reasoning:** Our method is evaluated on six arithmetic reasoning datasets derived from arithmetic reasoning tasks:

- **GSM8K** (Cobbe et al., 2021) is a dataset consisting of 8,500 high-quality, linguistically diverse elementary math problems created by humans. Of these, 7,500 problems are designated for training, while the remaining 1,000 are reserved for testing. These math problems typically require between 2 and 8 steps to solve and involve fundamental arithmetic operations, including addition, subtraction, multiplication, and division. In this study, we evaluate our method using 1,319 test questions.
- **AQUA-RAT** (Ling et al., 2017) is a dataset comprising 100,000 multiple-choice questions focused on mathematics, encompassing a wide range of topics and varying levels of difficulty. It is regarded as the most challenging dataset among the six arithmetic reasoning datasets. For our evaluation, we use a carefully curated set of 250 test questions.
- **SingleEq** (Single Equation) (Koncel-Kedziorski et al., 2015) is a dataset containing 508 elementary school algebra word problems, each of which can be represented by a single equation involving multiple arithmetic operations.
- **SVAMP** (Simple Variations on Arithmetic Math Word Problems) (Patel et al., 2021) is a dataset that comprises arithmetic word problems appropriate for fourth-grade students and below. It contains a total of 1,000 test questions.
- **MultiArith** (Roy and Roth, 2015) is a dataset that includes a collection of polynomial arithmetic problems, consisting of a series of questions and corresponding answers. This dataset is commonly used for training and evaluating the performance of machine learning models for polynomial arithmetic.
- The **AddSub** (Hosseini et al., 2014) dataset consists of 395 problems specifically focused on addition and subtraction.

**Other Reasoning:** Our method is further evaluated on two commonsense reasoning benchmarks and two logical reasoning tasks:

- **CSQA** (CommonsenseQA) (Talmor et al., 2019) is a multiple-choice benchmark dataset that presents questions with complex semantics, typically requiring reasoning over prior world knowledge.
- **StrategyQA** (Geva et al., 2021) is a dataset that requires models to carry out implicit, multi-step reasoning to arrive at an answer.
- **Date Understanding** (Srivastava et al., 2022) is a dataset that evaluates a model’s ability to infer specific dates from given contextual information.
- **Tracking Shuffled Objects** (Srivastava et al., 2022) is a dataset that evaluates a model’s ability to predict the final state of objects after being given an initial configuration and then subjected to a series of random reorderings.

## B Ablation Study

We used GPT-3.5-Turbo as the evaluation model and applied five perturbation methods across five arithmetic reasoning datasets (AQUA, GSM8K, SingleEq, AddSub, and MultiArith). The detailed experimental results are presented in Table 4.

## C Performance on Other Model

We evaluated RoP on the AQUA dataset using more advanced base models (GPT-4o, o1-mini, and o3-mini) across four perturbation methods (EC, SC, WOO, and HW). The detailed experimental results are shown in Table 5.

## D Different Levels of Perturbations

We used GPT-3.5-Turbo as the evaluation model and conducted experiments on the AQUA dataset using two perturbation methods (SC, HW) at different perturbation levels (perturbing 1, 4, and 7 characters). Figures 7 respectively show the detailed results for the two perturbation methods HW, and SC.

Method	AQUA	GSM8K	SingleEq	AddSub	MultiArith
No Dert.	74.0	75.7	95.7	90.4	90.5
EC	Stand	57.9	69.1	89.2	86.6
	CO	61.0	73.8	94.7	84.8
	GO	57.9	66.9	90.9	76.7
	<b>RoP</b>	<b>63.0</b>	<b>73.9</b>	<b>95.3</b>	<b>87.3</b>
SC	Stand	52.4	68.0	88.0	88.1
	CO	56.7	74.9	94.9	77.7
	GO	52.8	71.4	92.7	<b>90.6</b>
	<b>RoP</b>	<b>70.1</b>	<b>75.4</b>	<b>95.1</b>	82.5
WOO	Stand	59.5	70.1	88.8	88.6
	CO	57.5	69.8	88.2	60.3
	GO	60.6	71.3	<b>92.3</b>	<b>91.7</b>
	<b>RoP</b>	<b>65.8</b>	<b>71.3</b>	89.0	84.6
HW	Stand	56.3	66.4	84.1	82.0
	CO	59.1	73.2	87.4	62.3
	GO	59.8	68.5	83.1	82.8
	<b>RoP</b>	<b>64.6</b>	<b>74.2</b>	<b>90.9</b>	<b>84.3</b>
UIC	Stand	57.1	51.9	63.6	72.2
	CO	56.0	<b>63.4</b>	71.1	89.6
	GO	56.3	45.0	78.7	69.9
	<b>RoP</b>	<b>62.2</b>	62.9	<b>79.1</b>	<b>89.9</b>

Table 4: Accuracy(%) of different setting.

Method	GPT-4o	o1-mini	o3-mini
No Dert.	79.5	83.5	85.4
EC	Stand	79.1	80.7
	APE	80.3	79.5
	<b>RoP</b>	<b>81.5</b>	<b>82.7</b>
SC	Stand	71.7	<b>87.8</b>
	APE	74.4	70.9
	<b>RoP</b>	<b>75.2</b>	<b>87.4</b>
WOO	Stand	78.4	81.9
	APE	<b>79.9</b>	72.4
	<b>RoP</b>	79.1	<b>86.2</b>
HW	Stand	77.6	82.3
	APE	79.9	42.5
	<b>RoP</b>	<b>81.1</b>	<b>85.8</b>

Table 5: Accuracy(%) of different LLM on AQUA dataset.

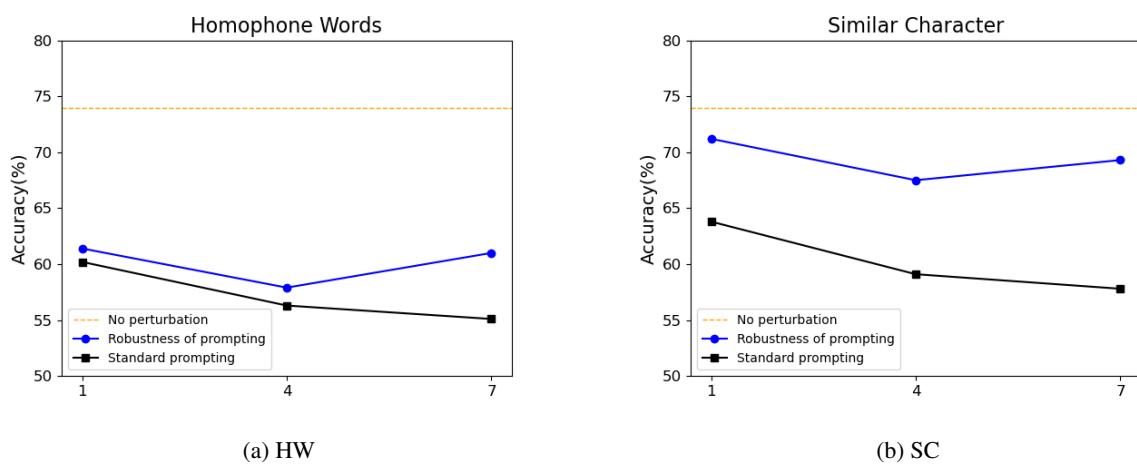


Figure 7: The effectiveness of different levels of perturbation.