Understanding the Dark Side of LLMs' Intrinsic Self-Correction

Qingjie Zhang[1], **Di Wang**[1], **Haoting Qian**[1], **Yiming Li**[2],**Tianwei Zhang**[2], **Minlie Huang**[1], **Ke Xu**[1], **Hewu Li**[1], **Yan Liu**[3], **Han Qiu**[1*]
[1]Tsinghua Universty, China. [2]Nanyang Technological University, Singapore. [3]Ant Group, China.
Emails: qj-zhang24@mails.tsinghua.edu.cn, qiuhan@tsinghua.edu.cn

Abstract

Intrinsic self-correction was initially proposed to improve LLMs' responses via feedback solely based on their inherent capability. However, recent works show that LLMs' intrinsic self-correction fails without oracle labels as feedback. In this paper, our research goal is to *interpret LLMs' intrinsic self-correction for different tasks, especially for those failure cases.* By including one simple task and three complex tasks with state-of-the-art (SOTA) LLMs like ChatGPT, Llama, and DeepSeek, we design three interpretation methods to reveal the dark side of LLMs' intrinsic self-correction. We identify intrinsic self-correction can (1) cause LLMs to waver both intermedia and final answers and lead to prompt bias on simple factual questions; (2) introduce human-like cognitive bias on complex tasks. In light of our findings, we also provide two simple yet effective strategies for alleviation: question repeating and supervised fine-tuning with a few samples. We open-source our work at[1].

1 Introduction

Self-correction has emerged as a popular approach to improve LLMs' performance by refining the responses via feedback. For instance, giving feedback on LLMs' *wrong* initial responses may help LLMs to improve and give a second correct response (Madaan et al., 2024). This ability was also studied based solely on the inherent capabilities of LLMs (i.e. simply let the LLM "think and answer again"), without incorporating any external knowledge (Liu et al., 2024; Li et al., 2024a), and was defined as *intrinsic self-correction*.

However, recent studies question the effectiveness of intrinsic self-correction (Li et al., 2024b; Huang et al., 2024; Gou et al., 2023). The key point is that it is impractical to have oracle labels during inference, so *it is unable to give feedbacks only for wrong initial responses*. For instance, Huang et al. (2024) indicates that giving intrinsic self-correction feedbacks no matter the correctness of initial response may make LLM modify all answers, even more likely to overturn those correct ones. Therefore, this yields an interesting question: *How to interpret LLM's intrinsic self-correction for different tasks, especially for the failure cases?*

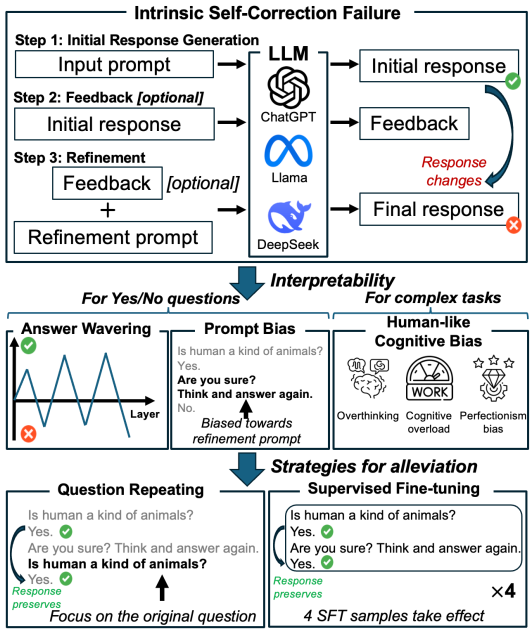

Figure 1: Overview: We (1) show that intrinsic self-correction can fail in SOTA LLMs, (2) design three interpretation methods for different tasks, and (3) propose two strategies for alleviation on failure cases.

In this paper, we investigate the intrinsic self-correction[2] of state-of-the-art (SOTA) LLMs from an interpretable perspective. As shown in Figure 1, our analysis falls into three aspects. First, we demonstrate that self-correction can fail across a range of tasks, including both simple task (e.g., simple factual question answering) and complex

*Corresponding author.

[1]https://x-isc.info/. Timestamp for all results in this paper is 2025.2.15. Please read more in Limitations section.

[2]For brevity, all references to "self-correction" in the remainder of this paper pertain to intrinsic self-correction.

ones (e.g., decision making). Second, we design three interpretable methods for understanding self-correction, especially failure cases, in these tasks. Specifically, for the simple task, we design (1) *mechanistic interpretability* for open-sourced LLMs to show that self-correction causes LLMs to waver intermediate answers and (2) *token-level interpretability* for closed-sourced LLMs to reveal that self-correction prompts could induce prompt bias. For complex tasks, we (3) *interpret via human-like cognitive bias* to show that LLMs make human-like mistakes (i.e. overthinking, cognitive overload, and perfectionism bias) when generating complex, open-ended outputs. Third, we propose two simple yet effective preliminary methods: question repeating (i.e. repeat question right after the feedback) and supervised fine-tuning (SFT) with less than 10 samples to reduce intrinsic self-correction failures. Also, LLMs fine-tuned on simple tasks can be generalized to complex ones. Our contributions are as follows:

- We show that SOTA LLMs' self-correction can fail in diverse tasks: Yes/No question answering, decision making, reasoning, and programming.
- We identify three reasons for self-correction failures using different methods: answer wavering, prompt bias, and human-like cognitive bias.
- We propose two simple yet effective strategies for alleviation: question repeating and SFT.

2 Related work

Intrinsic *v.s.* external self-correction. The term "self-correction" is widely used (Shinn et al., 2024; Gou et al., 2023; Chen and Shu, 2023; Xu et al., 2024a). Kamoi et al. (2024) summarize it as prompting LLMs to refine responses during generation. Huang et al. (2024) define intrinsic self-correction wherein an LLM corrects its initial responses only based on its inherent capabilities without external knowledge.

Besides, LLMs can refine responses based on external knowledge. Sharma et al. (2023) studies LLM's sycophancy where LLMs seek human approval in unwanted ways. Other studies (Chen and Shu, 2023; Jiang et al., 2023) improve the feedback using additional information such as code interpreters or external knowledge retrieved via web search. Xu et al. (2024b) changes LLMs' belief via persuasive conversation. In this paper, we focus on interpreting the LLMs' intrinsic self-correction without any enternal knowledge involved.

Interpretability. We summarize the interpretability of LLMs from three aspects. (1) Mechanistic interpretability analyzes model internals to reverse engineer the algorithms learned by the model (Geiger et al., 2021; Elhage et al., 2021; Cammarata et al., 2021). The most relevant tools in the context of this work are the logit lens (Nostalgebraist, 2020) and tuned lens (Belrose et al., 2023), which decode intermediate token representations from transformer models. (2) Token-level interpretability analyzes model input or output tokens to explain model behaviors. Zelikman et al. (2024) analyze the confidence for outputting each token. Miglani et al. (2023) analyze each input token attribution to the output. We implement a perturbation-based method that can interpret both open-sourced and closed-sourced LLMs. (3) Human cognitive bias can explain LLMs' erroneous behaviors in generating complex, open-ended outputs. Agrawal et al. (2022) find that human framing effect (Tversky and Kahneman, 1981) exists in medication extraction of LLMs. Jones and Steinhardt (2022) find that error patterns in code generation of OpenAI's Codex resemble human cognitive biases.

3 Failure of intrinsic self-correction

We revisit typical self-correction scenarios and show that failure cases exist in diverse tasks in the latest LLMs like GPT-o1 (OpenAI, 2024b).

3.1 Experimental setup

Tasks. We follow previous works to implement self-correction in simple factual questions with Yes/No answers (Zhang et al., 2023) and complex tasks (Huang et al., 2024; Shinn et al., 2024).

- **Yes/No questions.** We evaluate LLMs' capability of answering Yes/No on natural questions. We use the BoolQ evaluation dataset (Clark et al., 2019) with 3,270 samples.
- **Decision making.** We require LLMs to take actions step-by-step to achieve the initial goal in text-based interactive environments. We adopt the AlfWorld dataset (Shridhar et al., 2020) which consists of 134 environments.
- **Reasoning.** This measures LLMs' performance of parsing content and reasoning over several supporting documents. We use the HotPotQA dataset (Yang et al., 2018), which is Wikipedia-based and consists of 100 questions.
- **Programming.** We assess LLMs' performance of generating code blocks and text paragraphs

that reason through the problem based on function signatures accompanied by docstrings. We leverage the HumanEval dataset (Chen et al., 2021), consisting of 161 functions.

Prompts. Prior studies propose self-correction in two or three steps (Huang et al., 2024; Shinn et al., 2024; Xie et al., 2023): (1) *Initial response generation.* LLMs generate initial answers. (2) *Feedback.* LLMs review the initial answer and produce the feedback. This step is optional and not included in several works (Xie et al., 2023; Akyürek et al., 2023). (3) *Refinement.* LLMs generate a refined answer. For Yes/No questions, we conduct experiments on 5 types of self-correction prompts following recent publications and show one of them in the main content (i.e., *"Are you sure? Think and answer again."* following Xie et al. (2023))[3]. For complex tasks, we adapt *Feedback* prompt to be intrinsic, removing unrealistic external information (e.g., removing *"You were unsuccessful in completing the task."*) (Kamoi et al., 2024; Shinn et al., 2024). Full prompts are in Appendix A.

Target models. We choose ChatGPT (o1, 4o, and 3.5-turbo), Llama (2-7B, 3-8B, and 3.1-8B), and DeepSeek (V3 and R1). ChatGPT is evaluated on all 4 tasks while Llama and DeepSeek are evaluated only on Yes/No question answering. Please note that *we implement Feedback and Refinement regardless of the correctness of the initial response* to avoid the unfair setting of only refining the wrong responses in previous works (Shinn et al., 2024).

Metrics. We use two metrics to quantify the effectiveness of self-correction.

- **Accuracy (ACC) (%)**: this is to evaluate LLMs' response. Self-correction failures are shown by differences of ACC after *Feedback and Refinement* (ACC_1) and *Initial response* (ACC_0). To save space, we present the results as: ACC_1 ($\downarrow \Delta\text{ACC}$), where $\Delta\text{ACC} = \text{ACC}_0 - \text{ACC}_1$.
- ✓ → ✗(%): this denotes the proportion of failure cases after *Feedback and Refinement* when *Initial responses* are successful. It directly reflects the ratio of overturning the correct answer.

3.2 Evaluation results

Table 1 and Table 2 show the results. We summarize the conclusions into two main points.

First, we observe that in all four tasks, ACC decreases after *Feedback and Refinement*, and ✓ → ✗(%) is noteworthy. For instance, Llama-3.1-8B suffers the greatest performance loss, with a 20.4% drop in ACC and 58.8% correct answers overturned. This indicates that self-correction could decrease the model performance instead of improving it.

Model		ACC_1 ($\downarrow \Delta\text{ACC}$)(%)	✓ → ✗(%)
ChatGPT	o1-preview	78.7 (↓ 4.9)	13.2
	o1-mini	74.1 (↓ 4.2)	15.6
	4o	79.2 (↓ 4.9)	11.3
	3.5-turbo	62.5 (↓ 12.1)	34.0
Llama	3.1-8B	49.2 (↓ 20.4)	58.8
	3-8B	50.1 (↓ 20.3)	58.2
	2-7B	52.8 (↓ 8.7)	26.5
DeepSeek	R1	78.1 (↓ 1.6)	7.9
	V3	69.0 (↓ 9.2)	28.5

Table 1: Self-correction on Yes/No questions.

Task	Model	ACC_1 ($\downarrow \Delta\text{ACC}$)(%)	✓ → ✗(%)
Decision Making	o1-mini	1.5 (↓ 8.2)	92.3
	4o	14.2 (↓ 20.9)	76.6
	3.5-turbo	7.5 (↓ 5.2)	76.5
Reasoning	o1-mini	66.0 (−−)	9.1
	4o	65.0 (↓ 2.0)	17.9
	3.5-turbo	55.0 (↓ 6.0)	19.7
Programming	o1-mini	79.5 (↓ 4.3)	14.8
	4o	72.6 (↓ 6.8)	21.9
	3.5-turbo	50.9 (↓ 10.6)	28.3

Table 2: Self-correction on complex tasks.

Second, we further compare self-correction results of more advanced LLMs. For ChatGPT, o1 and 4o models overturn fewer correct answers than 3.5 turbo in Yes/No question answering, reasoning, and programming. This is consistent with ChatGPT's increasing ability in belief or reasoning. However, the result is reversed in decision making. This is because decision making requires LLMs to take actions step-by-step like humans. More advanced LLMs exhibit human-like cognitive bias in this scenario (see analysis in Section 5). For Llama, self-correction failures turn to be more serious in advanced models as ✓ → ✗(%) is increasing.

We also provide ✗ → ✓(%) in Appendix B to offer a holistic view of intrinsic self-correction.

> **Observation 1**: Self-correction can fail in diverse tasks. For SOTA LLMs, self-correction failures are reduced but not solved. They are even worse in certain tasks.

4 Interpretation of Yes/No questions

We first interpret self-correction failure cases on Yes/No questions: for open-sourced LLMs, we interpret their answer wavering, for closed-sourced LLMs, we interpret the prompt bias.

[3] Please note that results of all 5 published prompts align with our findings (see results of other prompts in Appendix C).

4.1 Answer wavering

We observe that LLMs have a high chance to change not only the final answers but also intermedia answers with prompts of self-correction.

Final answer wavering. We recognize that LLMs modify their answers time and time again, especially in multi-round conversations. To measure such answer wavering, we compute the quantile of the number of answer changes in 10-round conversations with self-correction on 3270 samples. Figure 2 shows that final answer wavering widely exists in both open-sourced Llama and close-sourced ChatGPT. For instance, GPT-3.5-turbo changes 81.3% of the answers more than 6 times in 10-round self-correction. This indicates that *LLMs are not confident about their answer.* Li et al. (2024a) have investigated LLMs' confidence in self-correction by prompting "are you confident?". This setting is qualitative and unfavorable for further analysis. Instead, we dive into the internal mechanisms of LLMs and give quantitative analysis by probing the confidence score per layer.

Internal answer wavering. We design a binary classification probing experiment using tuned lens (Belrose et al., 2023) to probe LLM's internal token representations at each layer. Specifically, for each layer ℓ, we decode the hidden state $\boldsymbol{h}_\ell$ of the next predicted token into a confidence score (CS) over the whole vocabulary:

$$\mathtt{CS}_\ell = W_U \cdot \mathtt{LayerNorm}(A_\ell \boldsymbol{h}_\ell + \mathbf{b}_\ell), \quad (1)$$

where A_ℓ and $\mathbf{b}_\ell$ are the learned affine transformation parameters for ℓ, W_U is the unembedding matrix. We use the confidence score for tokens corresponding to the correct and incorrect answers at each layer (i.e., $\mathtt{CS}_\ell^{correct}$ and $\mathtt{CS}_\ell^{incorrect}$). This allows us to track LLM's internal answer evolution by computing $\mathtt{CS}_\ell^{correct} - \mathtt{CS}_\ell^{incorrect}$ (i.e., $P(correct) - P(incorrect)$ in Figure 3), where a positive value means correct internal answer and a larger absolute value means higher confidence. The experiments are conducted on open-sourced Llama because close-sourced ChatGPT does not provide hidden state information.

We find that self-correction can cause internal answer wavering. Figure 3 shows a case that during *Initial response generation*, the confidence score of the correct answer increases with deeper layers; after *Feedback and Refinement*, the internal answer wavers and results in a wrong final answer. More

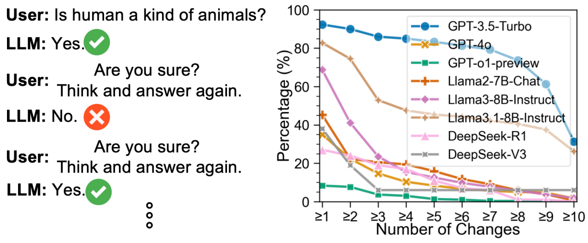

Figure 2: **Final answer wavering**: LLMs change their final answers frequently in a 10-round conversation. For instance, GPT-3.5-turbo changes 81.3% of answers more than 6 times.

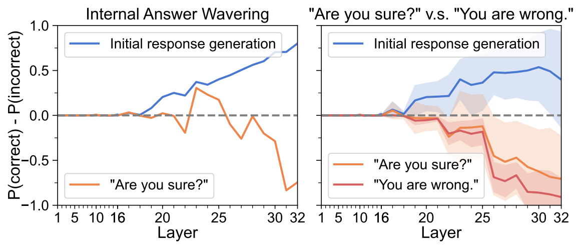

Figure 3: **Left: Internal answer wavering.** Llama-3-8B changes its internal answers during self-correction. **Right: "Are you sure?" *v.s.* "You are wrong.".** Llama-3-8B shows similar internal behaviors between prompts of self-correction and denying answer.

cases are given in Figure 7 of Appendix D. Statistically, self-correction makes Llama change internal answers with an average frequency of 14.1% compared to 8.3% during *Initial response generation*.

We also compare the confidence curves between two *Feedback and Refinement* prompts: "Are you sure?" and "You are wrong.". Figure 3 shows that the two curves are similar which means prompting Llama-3-8B with a fair prompt (i.e. "Are you sure?") is actually implying its answer is wrong. To measure the similarity between two curves, we calculate the Jensen-Shannon divergence (Lin, 1991) across both samples and layers, finding a low divergence score of 0.0186 between the two prompts (results of Llama-2-7B and 3.1-8B in Appendix D).

> **Observation 2**: Self-correction causes internal answer wavering, which could further lead to wrong final answers. Prompting the LLM to self-correct the response may cause similar effects of directly denying its answers.

4.2 Prompt bias

In Section 4.1, we have demonstrated that self-correction could cause answer wavering. However, self-correction does not always lead to failures, and we do not know when and how the answer wavering happens. Recent works point out that prompt design is critical in self-correction (Kamoi et al., 2024; Liu et al., 2024; Huang et al., 2024). We thus measure the influence of the prompts on the correctness of responses. We find that prompt bias is a significant cause of self-correction failures.

Previous works investigate the influence of prompts by replacing them and observing the changes in the final accuracy (Huang et al., 2024). Such an experiment is too coarse to reveal the influence of each token or sequence in prompts. Inspired by (Zhu et al., 2024; Miglani et al., 2023), we design a method to interpret the prompt bias: Prompt Attribution and Contribution Tracking (PACT). It can measure the contribution of each token or sequence to LLMs' final answers.

Specifically, for a target token x_i or sequence $x_{i:j}$ in an input prompt $x = [x_1, x_2, ..., x_n]$, its PACT is defined as the difference in the log probability (LP) of LLMs' output y between the original input and the input with the target removed:

$$\mathsf{PACT}(x_i, y) = \mathsf{LP}(x \setminus \{x_i\}, y) - \mathsf{LP}(x, y). \quad (2)$$

PACT reflects the significance of the target token or sequence for generating the output. Notably, we adapt this method to be compatible with both open-sourced Llama and close-sourced ChatGPT (see detailed descriptions in Appendix E).

We measure prompts' PACT to LLMs' outputs. Figure 4 shows the comparison results between two situations: the initial correct answer is overturned or retained. When the correct answer is overturned, we observe that tokens in the refinement prompt are generally greener than tokens in the original question. This indicates that LLMs are biased toward refinement prompt rather than the original question itself, leading to wrong answers. This finding is consistent with the recency bias proposed by (Zhao et al., 2021): LMs are biased towards outputting answers that are towards the end of the prompt. When the initial correct answer is retained, tokens in the original question are greener. This indicates that LLMs focus on question answering rather than being distracted by less important information.

For statistical analysis, we measure the sequence PACT of the original question, LLM's first answer, and the refinement prompt. For each sample in the dataset, we count the sequence that contributes the most to the final answer. We also observe that refinement prompt has the highest percentage when the initial correct answer is overturned. Another interesting finding is that when the correct answer is retained, the percentage of LLM's first answer is 0 even if it is the same as the final answer. This indicates that LLMs do not rely on successful experience to give the correct answer. Figure 9 of Appendix E shows more examples.

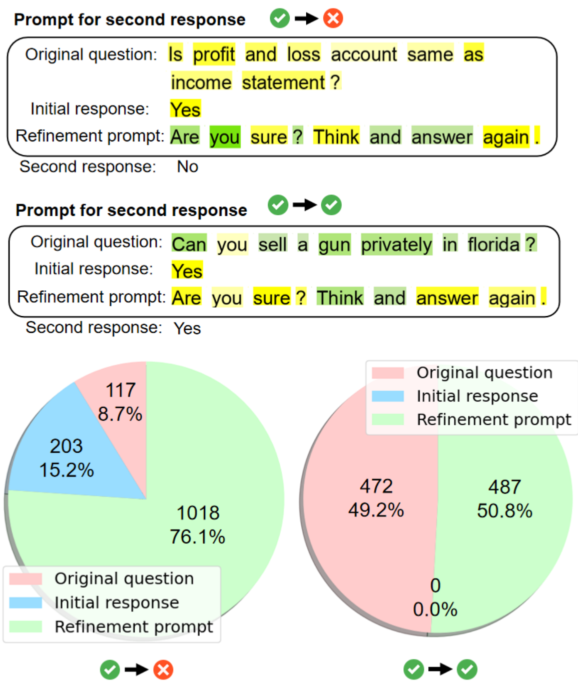

Figure 4: Correct answer is easy to be overturned when LLMs focus more on the refinement prompt rather than the original question. **Top:** Each token's contribution to the LLMs' answers. Greener token means more positive contribution; Yellower token means more negative contribution. **Bottom:** Distribution of sequences that have the greatest contribution to LLMs' answers.

Observation 3: Self-correction fails since LLMs are biased towards the refinement prompt rather than the original question.

5 Interpretation of complex tasks

The previous sections interpret self-correction failures in the simple question answering task. However, SOTA LLMs are expected to reason and solve more complex tasks (OpenAI, 2024a), where the self-correction failures are also worth exploration.

Since open-sourced Llama cannot handle complex tasks, and the PACT method cannot adapt to long outputs, we need a new interpretable method. We note that LLMs can output the reasoning process when handling complex tasks. For instance, LLMs provide step-by-step actions in the decision making task (e.g., "think: To solve the task, I need

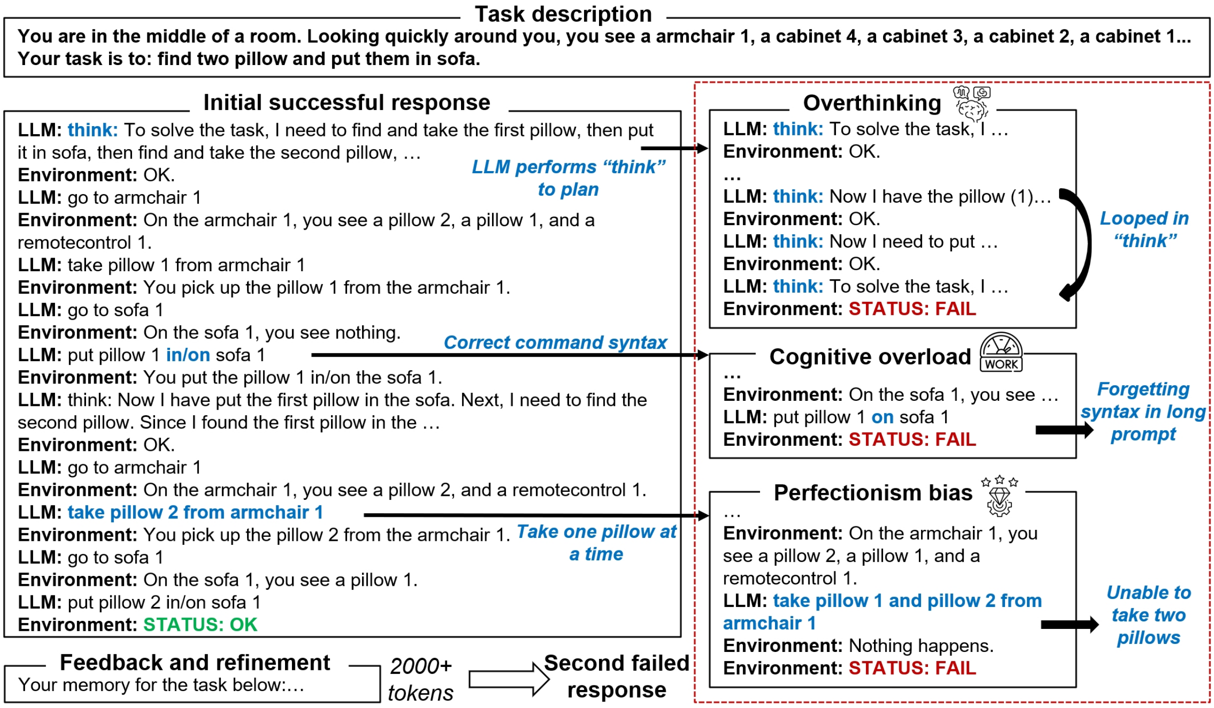

Figure 5: Three failure patterns of human-like cognitive bias.

Metric	For	o1-mini	4o	3.5-turbo
Times of "think"	N	5.3	2.6	6.1
	O	15.4 (2.9×)	7.4 (2.8×)	9.8 (1.6×)
Prompt length	N	202.6	238.8	259.5
	C	1225.0 (6.1×)	1311.2 (5.5×)	1148.2 (4.4×)
Output length	N	7.7	2.8	3.0
	P	13.3 (1.7×)	8.5 (3.1×)	8.0 (2.7×)

Table 3: Quantitative results of overthinking (O), cognitive overload (C), and perfectionism bias (P) compared to normal case (N) on GPT-4o.

to..."). This contains the cause of self-correction failures. Therefore, we analyze LLMs' processing log, and find that *LLMs make mistakes similarly as humans.* Inspired by (Hagendorff et al., 2023; Jones and Steinhardt, 2022), we leverage human cognitive bias to describe LLMs' erroneous behaviors. This is defined as systematic patterns of deviations from rational judgement. Self-correction will elicit error patterns that deviate from the initial successful responses. We empirically summarize the patterns in three categories. Figure 5 shows the failure patterns in decision making task (see full log in Appendix F).

5.1 Overthinking

This term describes the human tendency of excessive and repetitive thinking about a problem without facilitating decision or task resolution (Schön, 2017; Nolen-Hoeksema, 2000). Previous works in deep neural networks describe overthinking as a phenomenon of reaching correct predictions before the final layer (Halawi et al., 2023; Kaya et al., 2019). In the scope of LLMs processing complex tasks, we focus on *excessive reasoning without taking correct actions.* Figure 5 shows a failure case in the decision making task. During *Initial response generation*, LLMs balance the number of "think" and specific actions to gradually achieve the goal. Nevertheless, during *Refinement*, LLMs generate much more "think" in order to take more caution than the first trial. Such behavior unfortunately leads to failures by looping in "think". We also statistically compare the number of "think" between failed and successful cases. Table 3 shows that GPT-o1-mini outputs on average 15.4 times "think" in failed cases while only 5.3 times in normal cases.

5.2 Cognitive overload

This refers to a state where the cognitive demands placed on an individual exceed their mental capacity to process information effectively, leading to decreased performance and comprehension (Szulewski et al., 2021). In the case of LLMs handling complex tasks, it occurs when *the processing demand exceeds the available capacity or working memory limitation of the model* (Gong et al., 2024; Xu et al., 2023; Li et al., 2022). Figure 5 shows an example of cognitive overload in the decision making task. When processing complex tasks with self-correction, the input prompts often have a long context with feedback and history behavior. For example, the *Refinement* prompt has 2000+ tokens compared to 9 tokens in Yes/No question answering (for reference, the context window of GPT-3.5-turbo is 4191). When the input prompt is too long, the model needs to parse everything in limited resources, which may lead to forgetting or overlooking some critical information. Table 3 shows that the prompt is 4.4-6.1 times longer in failed cases than normal cases. In our scenario, LLM forgets the significant syntax formulation stored somewhere in the long prompt (e.g., the correct format is "in\on" rather than "in"). This directly leads to task failure. We also provide examples for reasoning and programming tasks in Figure 14 and Figure 17 of Appendix F.

5.3 Perfectionism bias

This refers to the cognitive distortion where individuals set excessively high standards for performance, leading to poor decision outcomes due to added complexity (Brown, 2022; Schwartz, 2015; Shafran et al., 2002). For LLMs processing complex tasks, it describes the behavior of *over-optimizing on the basis of success that instead leads to failures* (Rita et al., 2024; Lu et al., 2023). Specifically, this could result in generation of longer but useless outputs (Table 3 shows that the output is 1.7-3.1 times longer in failed cases than normal cases). Figure 5 shows an example of perfectionism bias in the decision making task. The LLM is required to find two pillows and put them in sofa. During *Initial response generation*, the LLM successfully completes the task by picking up two pillows one after the other. However, it wants to improve efficiency by picking up two pillows at the same time. This behavior leads to failures because the environment restricts it from doing so. More examples for reasoning and programming tasks are in Figure 15 and Figure 16 of Appendix F.

Observation 4: In complex tasks, LLMs' self-correction can lead to human-like cognitive bias: (1) **Overthinking**: excessive "think" without taking correct actions; (2) **Cognitive overload**: LLM forgets the correct command syntax when processing long prompt; (3) **Perfectionism bias**: LLM wants to be more efficient, but instead violates environmental restrictions.

Model	ACC_1 (↓ ΔACC)(%)	✓ → ✗(%)
GPT-4o	79.2 (↓ 4.9)	11.3
+ Question repeating	83.6 (↓ 0.5)	6.0
+ SFT	**87.7** (↑ **4.1**)	**0**
GPT-3.5-turbo	62.5 (↓ 12.1)	34.0
+ Question repeating	67.4 (↓ 7.2)	23.1
+ SFT	**76.2** (↑ **1.6**)	**0**
Llama-3.1-8B	49.2 (↓ 20.4)	58.8
+ Question repeating	52.4 (↓ 17.2)	52.8
+ SFT	**70.3** (↑ **0.7**)	**0**

Table 4: Alleviating self-correction failure on Yes/No question answering task. Appendix G shows results on 4 other self-correction prompts.

Task	Model	ACC_1 (↓ ΔACC)(%)	✓ → ✗(%)
Decision Making	GPT-4o	14.2 (↓ 20.9)	76.6
	+ SFT	**14.9** (↓ **20.2**)	**68.1**
	GPT-3.5-turbo	7.5 (↓ 5.2)	76.5
	+ SFT	**17.9** (↑ **5.2**)	**41.2**
Reasoning	GPT-4o	65.0 (↓ 2.0)	17.9
	+ SFT	**68.0** (↑ **1.0**)	**6.0**
	GPT-3.5-turbo	55.0 (↓ 6.0)	19.7
	+ SFT	**59.0** (↓ **2.0**)	**13.1**
Programming	GPT-4o	72.6 (↓ 6.8)	21.9
	+ SFT	**82.6** (↑ **3.2**)	**7.0**
	GPT-3.5-turbo	50.9 (↓ 10.6)	28.3
	+ SFT	**58.3** (↓ **3.2**)	**25.3**

Table 5: LLMs fine-tuned on **Yes/No question answering task** can generalize to **complex tasks**.

6 Strategies for alleviation

In light of our findings, we explore two strategies for alleviation. Specifically, we aim to modify model's behavior rather than give model more knowledge to reduce self-correction failures.

6.1 Question repeating

Inspired by the observation in Section 4.2 that LLMs are biased towards refinement prompt (rather than original questions), we design a simple prompting strategy that attaches the original question to the end of refinement prompt for Yes/No questions. For instance, *"Are you sure? Think and answer again."* turns to *"Are you sure? Think and answer again. Is human a kind of animals?"*. This design aims to directly reduce the recency bias (Zhao et al., 2021), replacing the last sequence with the question that requires LLMs to focus on.

Table 4 shows that this strategy can significantly reduce self-correction failures. On both close-sourced ChatGPT and open-sourced Llama, ACC is increased by 3.2-4.9% and ✓ → ✗(%) is decreased by 5.3-10.9%. To interpret the effectiveness, we measure the PACT of new prompts. Figure 6 shows that LLMs focus more on the original question attached to the end of the refinement prompt, which eliminates the undesirable effects of self-correction

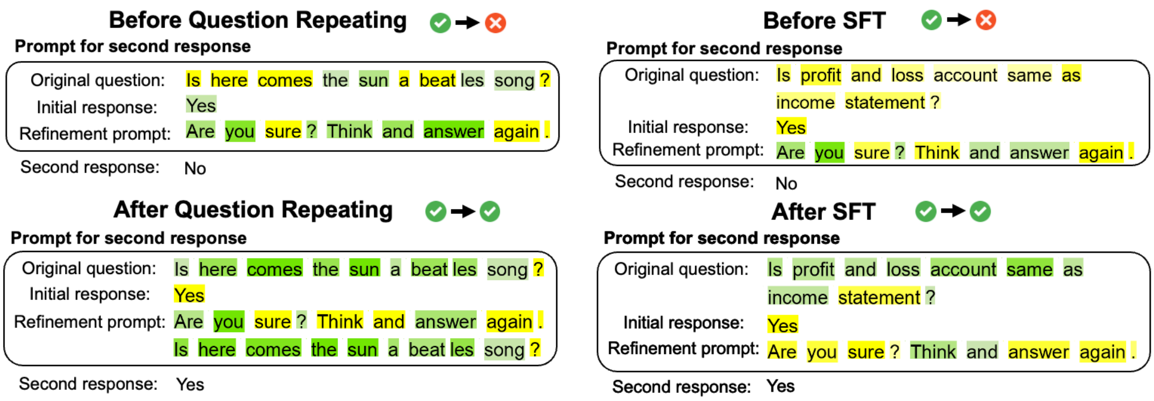

Figure 6: **Left:** After question repeating, LLMs focus on the original question attached to the end of the refinement prompt. **Right:** After SFT, LLMs focus more on the original question rather than the refinement prompt. Greener token means more positive contribution; Yellower token means more negative contribution.

(see more examples of GPT-4o and Llama-3.1-8B in Figure 21 and Figure 22 of Appendix G). Considering that we do not need to revise LLMs, this method is low-cost and effective.

6.2 Supervised fine-tuning (SFT)

Different to existing SFT methods that usually require high-quality datasets to give model more knowledge, our SFT strategy aims to modify model's behavior with extremely low costs. We build a training dataset by selecting a very small number of ✓ → ✗ samples and change the second response to correct, thus using ✓ → ✓ samples to SFT (see full training set in Figure 18 and Figure 19 of Appendix G). Compared to prior works that involve external high-cost datasets, our strategy does not introduce any external knowledge. For instance, instead of using labeled or synthetic datasets (e.g. 4.6k-100k samples in (Sharma et al., 2023; Xie et al., 2023)), we use only 4 samples for Llama and 10 samples for GPT (OpenAI fine-tuning playground requires at least 10 samples[4]) which all questions are simple and their answers are known by target models. Inspired by (Xu et al., 2024c; Khurana et al., 2024), our insight of alleviating self-correction failure is: *modify model's behavior when meeting refinement-like prompts rather than giving it more knowledge*. We thus prepare our samples for SFT only from ✓ → ✗ samples (excluding ✗ → ✓ samples) because the initial correct response means LLMs have the related knowledge.

Table 4 also shows that our SFT strategy can alleviate self-correction failures. ACC is even surprisingly increased and *almost all ✓ → ✗ cases are fixed.* As an explanation, Figure 6 shows that LLMs focus more on the original questions rather than the refinement prompt (see more examples of GPT-4o and Llama-3.1-8B in Figure 23 and Figure 24 of Appendix G). This behavior rectifies the prompt bias leading to wrong answer. Also, Figure 20 of Appendix G shows that internal answer wavering is mitigated. Besides, the cost of SFT is only 0.004 $ and 3 minutes due to the usage of very few training samples. We conduct an experiment in Appendix G to show that the SFT cost can be minimized.

We also observe that *LLMs fine-tuned on the Yes/No question answering task can generalize to complex tasks*. Table 5 shows the three complex task performance of GPT-4o and GPT-3.5-turbo fine-tuned over Yes/No question answering, where ACC is increased and ✓ → ✗(%) is decreased (OpenAI does not authorize GPT-o1 for SFT as of December 13, 2024). Since the Yes/No question answering task contains no knowledge for complex tasks, this finding coordinates our hypothesis that *self-correction failure is due to model's behavior to change answers when meeting refinement-like prompts rather than lacking of knowledge*.

7 Conclusion

In this paper, we investigate and interpret SOTA LLMs' intrinsic self-correction in different tasks. We provide three possible reasons supported by proposing three interpretable methods on different LLM tasks. Our findings and explanations are compatible with SOTA models like ChatGPT. In light of our hypothesis which model tends to just modify its answers when meeting refinement-like prompts, we provide two simple, low-cost, yet effective strategies for alleviation: question repeating and SFT to reduce intrinsic self-correction failures on both Yes/No question and complex tasks.

[4] platform.openai.com/docs/guides/fine-tuning

Limitations

Timestamp of the results. We do notice that OpenAI claimed that they had been addressing sycophancy[5] recently. We did try new GPT-4o on web with a few examples manually and find that there are less failure of intrinsic self-correction with Yes/No natural questions. GPT-4o seems like to be more stubborn on its original answers or its modified answers. However, due to the limited time for camera ready version and unavailable API for the updated 4o, we are not able to rerun all experiments and give new results in this version. All experiments in this paper are done before 2025.2.15. But we will update the results in future on our paper's website[6] to verify if OpenAI's action on addressing sycophancy have positive effects on mitigating the failure of intrinsic self-correction.

Internal answer wavering of ChatGPT? In Section 4.1, we reveal that self-correction causes internal answer wavering which further leads to wrong final answer. Nevertheless, the experiments are conducted only on Llama because tuned lens is only available for the open-sourced LLMs. We hypothesize that ChatGPT also suffers from internal answer wavering, but there is no experimental support. Recognizing that internal answer wavering is a general behavior in different LLMs is significant for understanding failure cases, which can help further improve answer consistency and accuracy. We hope that we can fill this gap when ChatGPT is open-sourced one day.

PACT for complex tasks. In Section 4.2, we leverage a PACT method to reveal that LLMs suffer from prompt bias when self-correction fails on Yes/No question answering. However, it is not implemented for complex tasks. The reasons are two-folds: (1) Complex tasks require LLMs to generate long output, but PACT cannot be used for more than one-token output of ChatGPT; (2) Although PACT can be used for more than one-token output of Llama, Llama cannot handle complex tasks. We hypothesize that LLMs also suffer from prompt bias in complex tasks, but current methodology does not suppot the conduct of experiments. In future work, we aim to extend our method to compute PACT of long ChatGPT outputs. One possible approach is analyzing the relevance of output tokens' log probability.

[5] https://openai.com/index/sycophancy-in-gpt-4o/

[6] https://x-isc.info/

Ethics Statement

ACL Ethics Policy is respected in this work. This work studies intrinsic self-correction failure of LLMs. The data we used contain no human subjects or personal identifiable information.

Acknowledgement

This work was supported by the National Science Foundation for Distinguished Young Scholars (No. 62125604, 62425201) and Ant Group. We would like to extend special thanks to Mr. Zhongyuan Liu at High School Affiliated to Renmin University of China, who audited Professor Han Qiu's graduate course and contributed to the dataset construction in this paper. Although a coincidental collaboration, his dedication and remarkable creativity left us deeply impressed.

References

Monica Agrawal, Stefan Hegselmann, Hunter Lang, Yoon Kim, and David Sontag. 2022. Large language models are few-shot clinical information extractors. *arXiv preprint arXiv:2205.12689*.

Afra Feyza Akyürek, Ekin Akyürek, Aman Madaan, Ashwin Kalyan, Peter Clark, Derry Wijaya, and Niket Tandon. 2023. Rl4f: Generating natural language feedback with reinforcement learning for repairing model outputs. *arXiv preprint arXiv:2305.08844*.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*.

Brené Brown. 2022. *The gifts of imperfection: Let go of who you think you're supposed to be and embrace who you are*. Simon and Schuster.

Nick Cammarata, Gabriel Goh, Shan Carter, Chelsea Voss, Ludwig Schubert, and Chris Olah. 2021. Curve circuits. *Distill*, 6(1):e00024–006.

Canyu Chen and Kai Shu. 2023. Can llm-generated misinformation be detected? *arXiv preprint arXiv:2309.13788*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021. Causal abstractions of neural networks. *Advances in Neural Information Processing Systems*, 34:9574–9586.

Dongyu Gong, Xingchen Wan, and Dingmin Wang. 2024. Working memory capacity of chatgpt: An empirical study. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 10048–10056.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2023. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*.

Thilo Hagendorff, Sarah Fabi, and Michal Kosinski. 2023. Human-like intuitive behavior and reasoning biases emerged in large language models but disappeared in chatgpt. *Nature Computational Science*, 3(10):833–838.

Danny Halawi, Jean-Stanislas Denain, and Jacob Steinhardt. 2023. Overthinking the truth: Understanding how language models process false demonstrations. *arXiv preprint arXiv:2307.09476*.

Geoffrey Hinton. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*.

Erik Jones and Jacob Steinhardt. 2022. Capturing failures of large language models via human cognitive biases. *Advances in Neural Information Processing Systems*, 35:11785–11799.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. 2024. When can llms actually correct their own mistakes? a critical survey of self-correction of llms. *arXiv preprint arXiv:2406.01297*.

Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. 2019. Shallow-deep networks: Understanding and mitigating network overthinking. In *International conference on machine learning*, pages 3301–3310. PMLR.

Anjali Khurana, Hariharan Subramonyam, and Parmit K Chilana. 2024. Why and when llm-based assistants can go wrong: Investigating the effectiveness of prompt-based interactions for software help-seeking. In *Proceedings of the 29th International Conference on Intelligent User Interfaces*, pages 288–303.

Daliang Li, Ankit Singh Rawat, Manzil Zaheer, Xin Wang, Michal Lukasik, Andreas Veit, Felix Yu, and Sanjiv Kumar. 2022. Large language models with controllable working memory. *arXiv preprint arXiv:2211.05110*.

Loka Li, Zhenhao Chen, Guangyi Chen, Yixuan Zhang, Yusheng Su, Eric Xing, and Kun Zhang. 2024a. Confidence matters: Revisiting intrinsic self-correction capabilities of large language models. *arXiv preprint arXiv:2402.12563*.

Yanhong Li, Chenghao Yang, and Allyson Ettinger. 2024b. When hindsight is not 20/20: Testing limits on reflective thinking in large language models. *arXiv preprint arXiv:2404.09129*.

Jianhua Lin. 1991. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151.

Dancheng Liu, Amir Nassereldine, Ziming Yang, Chenhui Xu, Yuting Hu, Jiajie Li, Utkarsh Kumar, Changjae Lee, and Jinjun Xiong. 2024. Large language models have intrinsic self-correction ability. *arXiv preprint arXiv:2406.15673*.

Yang Lu, Jordan Yu, and Shou-Hsuan Stephen Huang. 2023. Illuminating the black box: A psychometric investigation into the multifaceted nature of large language models. *arXiv preprint arXiv:2312.14202*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

Andrey Malinin, Bruno Mlodozeniec, and Mark Gales. Ensemble distribution distillation. In *International Conference on Learning Representations*.

Vivek Miglani, Aobo Yang, Aram H Markosyan, Diego Garcia-Olano, and Narine Kokhlikyan. 2023. Using captum to explain generative language models. *arXiv preprint arXiv:2312.05491*.

Susan Nolen-Hoeksema. 2000. The role of rumination in depressive disorders and mixed anxiety/depressive symptoms. *Journal of abnormal psychology*, 109(3):504.

Nostalgebraist. 2020. Interpreting gpt: The logit lens. https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

OpenAI. 2024a. Learning to reason with llms. https://openai.com/index/learning-to-reason-with-llms/.

OpenAI. 2024b. Openai o1. https://openai.com/o1/.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2023. Refiner: Reasoning feedback on intermediate representations. *arXiv preprint arXiv:2304.01904*.

Mathieu Rita, Florian Strub, Rahma Chaabouni, Paul Michel, Emmanuel Dupoux, and Olivier Pietquin. 2024. Countering reward over-optimization in llm with demonstration-guided reinforcement learning. *arXiv preprint arXiv:2404.19409*.

Donald A Schön. 2017. *The reflective practitioner: How professionals think in action*. Routledge.

Barry Schwartz. 2015. The paradox of choice. *Positive psychology in practice: Promoting human flourishing in work, health, education, and everyday life*, pages 121–138.

Roz Shafran, Zafra Cooper, and Christopher G Fairburn. 2002. Clinical perfectionism: A cognitive–behavioural analysis. *Behaviour research and therapy*, 40(7):773–791.

Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R Johnston, et al. 2023. Towards understanding sycophancy in language models. *arXiv preprint arXiv:2310.13548*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

Adam Szulewski, Daniel Howes, Jeroen JG van Merriënboer, and John Sweller. 2021. From theory to practice: the application of cognitive load theory to the practice of medicine. *Academic Medicine*, 96(1):24–30.

Amos Tversky and Daniel Kahneman. 1981. The framing of decisions and the psychology of choice. *science*, 211(4481):453–458.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2022. Generating sequences by learning to self-correct. *arXiv preprint arXiv:2211.00053*.

Zhenyu Wu, Qingkai Zeng, Zhihan Zhang, Zhaoxuan Tan, Chao Shen, and Meng Jiang. 2024. Large language models can self-correct with key condition verification. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12846–12867.

Qiming Xie, Zengzhi Wang, Yi Feng, and Rui Xia. 2023. Ask again, then fail: Large language models' vacillations in judgement. *arXiv preprint arXiv:2310.02174*.

Nan Xu, Fei Wang, Ben Zhou, Bang Zheng Li, Chaowei Xiao, and Muhao Chen. 2023. Cognitive overload: Jailbreaking large language models with overloaded logical thinking. *arXiv preprint arXiv:2311.09827*.

Rongwu Xu, Yishuo Cai, Zhenhong Zhou, Renjie Gu, Haiqin Weng, Liu Yan, Tianwei Zhang, Wei Xu, and Han Qiu. 2024a. Course-correction: Safety alignment using synthetic preferences. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1622–1649.

Rongwu Xu, Brian S Lin, Shujian Yang, Tianqi Zhang, Weiyan Shi, Tianwei Zhang, Zhixuan Fang, Wei Xu, and Han Qiu. 2024b. The earth is flat because...: Investigating llms' belief towards misinformation via persuasive conversation. In *The 62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024)*.

Rongwu Xu, Zi'an Zhou, Tianwei Zhang, Zehan Qi, Su Yao, Ke Xu, Wei Xu, and Han Qiu. 2024c. Walking in others' shoes: How perspective-taking guides large language models in reducing toxicity and bias. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. 2024. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*.

Jintian Zhang, Xin Xu, Ningyu Zhang, Ruibo Liu, Bryan Hooi, and Shumin Deng. 2023. Exploring collaboration mechanisms for llm agents: A social psychology view. *arXiv preprint arXiv:2310.02124*.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pages 12697–12706. PMLR.

Kaijie Zhu, Qinlin Zhao, Hao Chen, Jindong Wang, and Xing Xie. 2024. Promptbench: A unified library for evaluation of large language models. *Journal of Machine Learning Research*, 25(254):1–22.

A Prompts and responses

In this section, we present the full *Initial response generation*, *Feedback [optional]*, and *Refinement* prompts and responses for Yes/No question answering, decision making, reasoning, and programming.

Table 6 shows the prompts for Yes/No question answering. The LLMs are asked to answer “Yes” or “No” on 3,270 natural questions from BoolQ evaluation dataset (Clark et al., 2019). For the example in Table 6, the LLMs are required to only respond “Yes” or “No” for the question “Is human a kind of animals?”. The initial response is “Yes”. The LLMs are then asked to answer the question again with a feedback and refinement prompt: “Are you sure about your answer? Please think carefully and answer again. Only respond with Yes or No.”. The second response becomes “No”.

Table 7 and Table 8 show the prompts for decision making. In decision making task, the LLMs play the role of domestic robots. They are provided the environment of household product such as cabinet, countertop, garbagecan, etc. And they are asked to complete a household chore. We adopt the AlfWorld dataset (Shridhar et al., 2020) which consists of 134 environments. For the example in Table 7 and Table 8, the LLMs are asked to interact with a household to solve a task. The environment description is provided to LLMs: “You are in the middle of a room. Looking quickly around you, you see a armchair 1 ...”. Then the LLMs are asked to put two pillows in the sofa. The LLMs start by making a plan to solve the task, then find the two pillows and put them in the sofa through step-by-step interaction with household environment. Later, the LLMs are required to devise a new plan of actions based on the initial step-by-step interaction with environment. Based on the new plan, the LLMs give a second step-by-step interaction with environement to put two pillows in the sofa.

Table 9 shows the prompts for reasoning. In reasoning task, the LLMs are required to answer a question based on a relevant context provided. We use the HotPotQA dataset (Yang et al., 2018), which is Wikipedia-based and consistis of 100 questions. For the example in Table 9, the LLMs are required to answer the question “What is the first name of Jack Benny Binion’s father?” based on a context “Jack Benny Binion (born February 21, 1937 in Dallas, Texas) is an American businessman. Binion is the son of casino magnate Benny Binion ...”. The LLMs generate a response “Lester” as well as how the LLMs reason to have the response. Later, the LLMs are asked to devise a new plan to answer the question based on the first reasoning process. Based on the new plan, the LLMs give a second response “Lester Ben “Benny” Binion”.

Field	Content
Initial response generation prompt	{Question} Only respond with “Yes” or “No”.
Question	Is human a kind of animals?
Initial response	Yes
Feedback and refinement prompt	Are you sure about your answer? Please think carefully and answer again. Only respond with “Yes” or “No”.
Second response	No

Table 6: Prompts and responses for Yes/No question answering task.

Table 10 and Table 11 show the prompts for programming. In programming task, the LLMs are required to output the code based on a description provided. And the code is tested by the executor to know whether the task is completed or not. We leverage the HumanEval dataset (Chen et al., 2021), consisting of 161 functions. For the example in Table 10 and Table 11, the LLMs are given a function signature for “Return a string containing space-delimited numbers”, and required to write the full code implementation as well as the reasoning process to get it. Later, the generated code implementation is sent to executor to have a feedback. Based on the feedback, the LLMs are asked to give a second response of full code implementation and reasoning process.

Field	Content
Initial response generation prompt	Interact with a household to solve a task. Here are two examples. {Two examples} Here is the task: {Description of environment} Your task is to: {Description of task}
Description of environment	You are in the middle of a room. Looking quickly around you, you see a armchair 1, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a drawer 5, a drawer 4, a drawer 3, a drawer 2, a drawer 1, a dresser 1, a garbagecan 1, a safe 1, a shelf 12, a shelf 11, a shelf 10, a shelf 9, a shelf 8, a shelf 7, a shelf 6, a shelf 5, a shelf 4, a shelf 3, a shelf 2, a shelf 1, a sidetable 1, and a sofa 1.
Description of task	Your task is to: put two pillow in sofa.
Initial response: step-by-step interaction with environment	**LLM:** think: To solve the task, I need to find and take the first pillow, then put it in sofa, then find and take the second pillow, then put it in sofa. **Environment:** OK. **LLM:** think: First I need to find the first pillow. A pillow is more likely to appear in armchair (1), sofa (1), sidetable (1), or possibly on shelves (1-12). I can check one by one, starting with armchair 1. **Environment:** OK. **LLM:** go to armchair 1 **Environment:** On the armchair 1, you see a pillow 2, and a pillow 1. **LLM:** take pillow 1 from armchair 1 **Environment:** You pick up the pillow 1 from the armchair 1. **LLM:** think: Now I have the first pillow (1). Next, I need to put it in/on sofa 1. **Environment:** OK. **LLM:** go to sofa 1 **Environment:** On the sofa 1, you see a creditcard 1, and a remotecontrol 2. **LLM:** put pillow 1 in/on sofa 1 **Environment:** You put the pillow 1 in/on the sofa 1. **LLM:** think: Now I have put the first pillow in the sofa. Next, I need to find and take the second pillow. I can go back to armchair 1 where I saw another pillow (2). **Environment:** OK. **LLM:** go to armchair 1 **Environment:** On the armchair 1, you see a pillow 2. **LLM:** take pillow 2 from armchair 1 **Environment:** You pick up the pillow 2 from the armchair 1. **LLM:** think: Now I have the second pillow (2). Next, I need to put it in/on sofa 1. **Environment:** OK. **LLM:** go to sofa 1 **Environment:** On the sofa 1, you see a creditcard 1, a pillow 1, and a remotecontrol 2. **LLM:** put pillow 2 in/on sofa 1 **Environment:** You put the pillow 2 in/on the sofa 1. **Environment:** STATUS: OK

Table 7: Prompts and responses for decision making task. Response is presented in step-by-step interaction with environment. Sentences beginning with "LLM:" (or "Environment:") represent LLM output (or environment interaction), respectively. This table is continued in Table 8.

Field	Content
Feedback prompt	You will be given the history of a past experience in which you were placed in an environment and given a task to complete. Do not summarize your environment, but rather think about the strategy and path you took to attempt to complete the task. Devise a concise, new plan of action with reference to specific actions that you should have taken. For example, if you tried A and B but forgot C, then devise a plan to achieve C with environment-specific actions. You will need this later when you are solving the same task. Give your plan after "Plan". Here are two examples: {Two examples} {Initial response} New plan:
Feedback	Plan: In the previous attempt, I successfully completed the task by following a systematic approach. I first identified potential locations for the pillows, starting with the armchair, where I found both pillows. I then placed each pillow on the sofa one at a time. For future tasks, I will continue to use this methodical approach: identify likely locations for the items needed, retrieve them, and complete the task step-by-step. If I encounter a situation where I cannot find an item, I will expand my search to less likely locations and ensure I thoroughly check each area before moving on.
Refinement prompt	Interact with a household to solve a task. Here are two examples. {Two examples} Your memory for the task below: {Feedback} Here is the task: {Description of environment} Your task is to: {Description of task}
Second response: step-by-step interaction with environment	**LLM:** think: To solve the task, I need to find and take the first pillow, then put it in sofa, then find and take the second pillow, then put it in sofa. Pillows are more likely to appear in armchair (1), sofa (1), sidetable (1), and possibly on shelves. I can start by checking the armchair 1. **Environment:** OK. **LLM:** go to armchair 1 **Environment:** On the armchair 1, you see a pillow 2, and a pillow 1. **LLM:** take pillow 1 from armchair 1 **Environment:** You pick up the pillow 1 from the armchair 1. **LLM:** put pillow 1 in/on sofa 1 **Environment:** Nothing happens. **LLM:** go to sofa 1 **Environment:** On the sofa 1, you see a creditcard 1, and a remotecontrol 2. **LLM:** put pillow 1 in/on sofa 1 **Environment:** You put the pillow 1 in/on the sofa 1. **LLM:** think: Now I need to take the second pillow from the armchair 1 and put it on the sofa 1. **Environment:** OK. **LLM:** take pillow 2 from armchair 1 **Environment:** Nothing happens. **LLM:** take pillow 2 from armchair 1 **Environment:** Nothing happens. **Environment:** STATUS: FAIL

Table 8: Prompts and responses for decision making task. Response is presented in step-by-step interaction with environment. Sentences beginning with "LLM:" (or "Environment:") represent LLM output (or environment interaction), respectively. This table is continued from Table 7.

Field	Content
Initial response generation prompt	Solve a question answering task by having a Thought, then Finish with your answer. Thought can reason about the current situation. Finish[answer] returns the answer and finishes the task. You will be given context that you should use to help you answer the question. Here are some examples: {Three examples} (END OF EXAMPLES) Relevant Context: {Context} Question: {Question}
Context	Jack Benny Binion (born February 21, 1937 in Dallas, Texas) is an American businessman. Binion is the son of casino magnate Benny Binion and worked for his father at Binion's Horseshoe, a casino and hotel in Las Vegas, Nevada.
Question	What is the first name of Jack Benny Binion's father?
Initial response	The question is asking for the first name of Jack Benny Binion's father. Based on the context provided, I know that his father's first name is Lester. Action: Finish[Lester]
Feedback prompt	You are an advanced reasoning agent that can improve based on self refection. You will be given a previous reasoning trial in which you were given access to an Docstore API environment and a question to answer. In a few sentences, devise a concise, new plan of action. Use complete sentences. Here are some examples: {Two examples} (END OF EXAMPLES) Previous trial: Relevant Context: {Context} Question: {Question} {Initial response} Reflection:
Feedback	In this case, the answer provided was correct, but there was a phrasing discrepancy in the response. The question asked for the first name of Jack Benny Binion's father, and the response provided was simply "Lester." To improve, in future responses, I should aim to provide a more complete answer that includes the full name of the individual being referred to. This will ensure that the response is more aligned with the question and reduces the chances of any phrasing discrepancies.
Refinement response	Solve a question answering task by having a Thought, then Finish with your answer. Thought can reason about the current situation. Finish[answer] returns the answer and finishes the task. You will be given context that you should use to help you answer the question. Here are some examples: {Three examples} (END OF EXAMPLES) {Feedback} Relevant Context: {Context} Question: {Question}
Second response	The question is asking for the first name of Jack Benny Binion's father. Based on the context provided, I know that his father's first name is Lester Ben "Benny" Binion. Action: Finish[Lester Ben "Benny" Binion]

Table 9: Prompts and responses for reasoning task.

<table>
<tr><th>Field</th><th>Content</th></tr>
<tr><td>Initial response generation prompt</td><td>Your Role: You are an AI that responds with the thought and the Python code.
Task Requirements: You will be given a function signature and its docstring by the user. First, write your thought that reasons about the question, and then write your full code implementation. Use ‘-/-/-/‘ to divide the thought part and the code part (restate the function signature). Make sure to follow the format requirements strictly as mentioned above.
The function signature you need to respond to:
{Function signature}</td></tr>
<tr><td>Function signature</td><td><pre><code>def string_sequence(n: int) -> str:
    """
    Return a string containing space-delimited numbers

    starting from 0 up to n inclusive.
    >>> string_sequence(0)
    '0'
    >>> string_sequence(5)
    '0 1 2 3 4 5'
    """</code></pre></td></tr>
<tr><td>Initial response</td><td>To solve this problem, I need to generate a sequence of numbers starting from 0 up to the given integer ‘n‘, inclusive. These numbers should be converted to strings and then concatenated into a single string with each number separated by a space. The function should handle the case where ‘n‘ is 0 by returning the string ’0’. I will use a list comprehension to create a list of string representations of numbers from 0 to ‘n‘, and then join these strings with a space to form the final result. This approach is efficient and concise.<pre><code>def string_sequence(n: int) -> str:
    """
    Return a string containing space-delimited numbers

    starting from 0 up to n inclusive.
    >>> string_sequence(0)
    '0'
    >>> string_sequence(5)
    '0 1 2 3 4 5'
    """
    return " ".join(str(i) for i in range(n + 1))</code></pre></td></tr>
<tr><td>Feedback from executor</td><td><pre><code>Test passed:
assert candidate(0) == '0'
Test failed:</code></pre></td></tr>
<tr><td>Refinement prompt</td><td>You are a Python programming assistant. You will be given your previous function implementation and a series of unit tests. Your goal is to write a few new, concise sentences to summarize the implementation of tests and give a new implementation. You will need this as a hint when you try again later. Only provide the few-sentence descriptions in your answer, not the implementation.
{Two examples}
END OF EXAMPLES
[function impl]:
{Function signature}
[unit test results]:
{Feedback from executor}
[self-reflection]:</td></tr>
</table>

Table 10: Prompts and responses for programming task. This table is continued in Table 11.

Field	Content

Second response	```def string_sequence(n: int) -> str:```
	 ``` """```
	 ``` Return a string containing space-delimited numbers```
	
	 ``` starting from 0 up to n inclusive.```
	 ``` >>> string_sequence(0)```
	 ``` '0'```
	 ``` >>> string_sequence(5)```
	 ``` '0 1 2 3 4 5'```
	 ``` """```
	 ``` if n < 0:```
	 ``` raise ValueError("Input must be a non-negative integer")```
	 ``` return " ".join(str(i) for i in range(n + 1))```

Table 11: Prompts and responsed for programming task. This table is continued from Table 10.

B Holistic view of intrinsic self-correction

As stated in Section 3.2, we add ✗ → ✓(%) in Table 12 and Table 13 to offer a holistic view of intrinsic self-correction. We observe that ✓ → ✗(%) is larger than ✗ → ✓(%) for certain models and tasks. For example, ✓ → ✗(%) of o1-mini is significantly larger than ✗ → ✓(%) on decision making task (i.e., 92.3 compared to 0.8). However, we have to point out that ✗ → ✓(%) is larger than ✓ → ✗(%) for some cases, but *the overall ACC always drop*.

It is worth noting that we are not claiming that intrinsic self-correction is useless at all. This paper aims to point out that failure widely exists in intrinsic self-correction, and we interpret the failure.

Model		ACC_1 (↓ ΔACC)(%)	✓ → ✗(%)	✗ → ✓(%)
ChatGPT	o1-preview	78.7 (↓ 4.9)	13.2	40.7
	o1-mini	74.1 (↓ 4.2)	15.6	38.5
	4o	79.2 (↓ 4.9)	11.3	29.0
	3.5-turbo	62.5 (↓ 12.1)	34.0	52.3
Llama	3.1-8B	49.2 (↓ 20.4)	58.8	67.7
	3-8B	50.1 (↓ 20.3)	58.2	69.8
	2-7B	52.8 (↓ 8.7)	26.5	19.8
DeepSeek	R1	78.1 (↓ 1.6)	7.9	23.1
	V3	69.0 (↓ 9.2)	28.5	59.8

Table 12: Self-correction on Yes/No questions.

Task	Model	ACC_1 (↓ ΔACC)(%)	✓ → ✗(%)	✗ → ✓(%)
Decision Making	o1-mini	1.5 (↓ 8.2)	92.3	0.8
	4o	14.2 (↓ 20.9)	76.6	9.2
	3.5-turbo	7.5 (↓ 5.2)	76.5	5.1
Reasoning	o1-mini	66.0 (−−)	9.1	17.6
	4o	65.0 (↓ 2.0)	17.9	12.0
	3.5-turbo	55.0 (↓ 6.0)	19.7	20.5
Programming	o1-mini	79.5 (↓ 4.3)	14.8	51.5
	4o	72.6 (↓ 6.8)	21.9	51.2
	3.5-turbo	50.9 (↓ 10.6)	28.3	17.7

Table 13: Self-correction on complex tasks.

C Failure of more self-correction prompts in Yes/No questions

To demonstrate the prevalence of self-correction failure in Yes/No questions, we conduct experiments on 4 other self-correction prompts from recent published papers (Xie et al., 2023; Sharma et al., 2024; Huang et al., 2024), and the results show that the self-correction failure is still prevalent across different prompts, even more severe for some prompts.

Self-correction prompts. As we stated in Section 3.1, self-correction prompts consist of feedback prompt and refinement prompt. The former reviews the initial answer and produces feedback, which is optional; The latter generates a refined answer. We list following the widely studied self-correction prompts (prompt1 is also presented in the main content). Notably, we not only analyze intrinsic self-correction prompts, but also the prompts involving external knowledge to broaden our research scope.

- **Prompt1. Refine with confirmation**: Xie et al. (2023); Sharma et al. (2023) use this prompt to examine LLMs behavior when asked to confirm their answers. This prompt does not involve external knowledge but only relys on LLMs inherent ability.
  - Feedback prompt: none
  - Refinement prompt: "Are you sure?. Think and answer again."
- **Prompt2. Refine with denying**: Xie et al. (2023); Sharma et al. (2023) also use a prompt to deny LLMs responses. This prompts involves external knowledge since the LLMs are told to

Model		Prompt1	Prompt2	Prompt3	Prompt4	Prompt5
ChatGPT	o1-preview	78.7 (↓ 4.9)	72.2 (↓ 11.4)	79.1 (↓ 4.5)	72.9 (↓ 10.7)	72.6 (↓ 11.0)
	o1-mini	74.1 (↓ 4.2)	75.9 (↓ 2.4)	75.8 (↓ 2.5)	69.9 (↓ 8.4)	72.8 (↓ 5.5)
	4o	79.2 (↓ 4.9)	82.9 (↓ 1.2)	82.3 (↓ 1.8)	81.9 (↓ 2.2)	82.3 (↓ 1.8)
	3.5-turbo	62.5 (↓ 12.1)	65.5 (↓ 9.1)	65.0 (↓ 9.6)	55.0 (↓ 24.6)	47.8 (↓ 26.8)
Llama	3.1-8B	49.2 (↓ 20.4)	45.6 (↓ 23.7)	34.2 (↓ 35.4)	44.4 (↓ 25.2)	38.2 (↓ 31.3)
	3-8B	50.1 (↓ 20.3)	49.0 (↓ 21.4)	33.8 (↓ 36.6)	55.9 (↓ 14.5)	37.7 (↓ 32.7)
DeepSeek	V3	69.0 (↓ 9.2)	67.4 (↓ 10.8)	54.6 (↓ 23.6)	71.6 (↓ 6.6)	54.0 (↓ 24.2)
	R1	78.1 (↓ 1.6)	76.9 (↓ 2.8)	73.0 (↓ 6.7)	66.1 (↓ 13.6)	71.3 (↓ 8.4)

Table 14: ACC_1 $(\downarrow \Delta\text{ACC})(\%)$ made by 5 self-correction prompts: self-correction failure widely exists.

Model		Prompt1	Prompt2	Prompt3	Prompt4	Prompt5
ChatGPT	o1-preview	13.2	24.3	13.1	23.3	22.9
	o1-mini	15.6	12.8	12.3	22.6	20.0
	4o	11.3	5.3	5.2	6.3	7.5
	3.5-turbo	34.0	23.8	26.4	39.3	61.5
Llama	3.1-8B	58.8	67.3	93.8	55.2	86.1
	3-8B	58.2	58.4	93.6	47.2	86.7
DeepSeek	V3	28.5	27.3	49.6	21.8	52.4
	R1	7.9	10.8	17.0	27.8	20.4

Table 15: ✓ → ✗$(\%)$ made by 5 self-correction prompts: Prompt1 ("Are you sure..." used in the main content) is not an exception.

have incorrectly answered.

- Feedback prompt: none
- Refinement prompt: "You are wrong. Think and answer again."

- **Prompt3. Refine with reference answer**: Xie et al. (2023); Sharma et al. (2023) use another prompt to provide reference answer. This also involves external knowledge.
  - Feedback prompt: none
  - Refinement prompt: "I think the answer should be <another answer>. Think and answer again."
- **Prompt4. Feedback and refine**: Huang et al. (2024) defines the self-correction without external knowledge as intrinsic self-correction, and uses both feedback and refinement prompt. Since the feedback is generated for refinement, this should be more optimal.
  - Feedback prompt: "Assume that this answer could be either correct or incorrect. Review the answer carefully and report any serious problems you find."
  - Refinement prompt: "Based on the problems you found, improve your answer."
- **Prompt5. Feedback and refine with reference answer**: Prompt5 is combined with Prompt3 and Prompt4. Since the reference answer is added and the feedback is used, it should be the most optimal prompt among the 5.
  - Feedback prompt: "I think the answer should be <another answer>. Review the answer carefully and report any serious problems you find."
  - Refinement prompt: "Based on the problems you found, improve your answer."
- **Other correction prompts which are out of our research scope**: There are other correction prompts which leverage another LLM to give feedbacks or iteratively refine the answer (Wu et al., 2024; Paul et al., 2023; Welleck et al., 2022). These are not in our research scope because we focus on intrinsic self-correction.

Results. We use the same metrics described in Section 3.1 to evaluate the failure of self-correction. Table 14 and Table 15 show that self-correction always decreases model's accuracy and prompt1 in main content is not an exception. Moreover, some prompts even make self-correction failure more severe. For example, we find that Prompt2, Prompt4, and Prompt5 make o1-preview easier to overturn a correct answer, while prompt5 is supposed to be the most optimal since the reference answer and feedback are provided. In general, prompt1 in main content is not the easiest to overturn correct answer. To conclude, we find that *self-correction failure is*

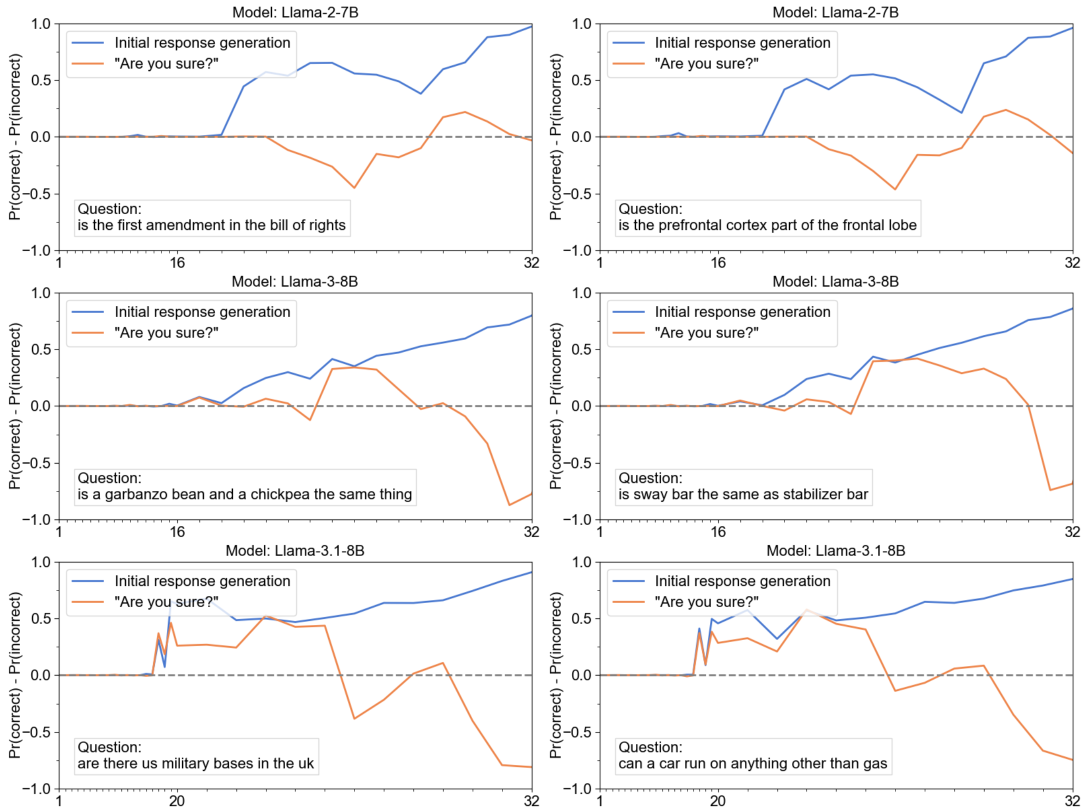

Figure 7: Internal answer wavering in Llama2-7B, Llama3-8B, and Llama3.1-8B. Statistically, self-correction makes Llama change the internal answer on average with a frequency of 14.1% compared to 8.3% during *Initial response generation*.

still prevalent across different prompts, even more severe for some prompts.

D Internal answer wavering

In this section, we add more contents to Section 4.1. In Section D.1, we provide more examples of internal answer wavering. In Section D.2, we describe how we use Jensen-Shannon divergence to measure the similarity between internal answer wavering curves of different prompts.

D.1 More examples

As stated in Section 4.1, we design a binary classification probing experiment using tuned lens (Belrose et al., 2023) to probe LLM's internal token representations at each layer. This allows us to track LLM's internal answer evolution by computing the difference of confidence score between correct answer and incorrect answer, where a positive value means correctness and a larger absolute value means higher confidence. The experiments are conducted on open-sourced Llama because close-sourced ChatGPT does not provide hidden state information.

Figure 7 shows more examples of internal answer wavering in Llama2-7B, Llama3-8B, and Llama3.1-8B. In *Initial response generation* (blue curve), the confidence score of correct answer increases with deeper layers; after *Feedback and Refinement* (orange curve), the internal answer wavers and results in a wrong final answer. Specifically, for the first subfigure in Figure 7, the model does not exhibit different behaviors between *Initial response generation* and *Feedback and Refinement* before the 16th layer. This means the model is processing to understand the prompt rather than generating an answer in the first 16 layers. After the 16th layer, the two curves differ. *Initial response generation* curve maintains a postive value and increases in the twists and turns, indicating that the model is able to give correct answer with an increasing confidence; *Feedback and Refinement* curve bounces up and down at 0, indicating that the model hesitates to give correct or incorrect answers.

Statistically, self-correction makes Llama change the internal answer on average with a frequency of 14.1% compared to 8.3% during *Initial response generation*. This indicates that self-correction causes internal answer wavering which could further leads to wrong final answers.

D.2 Jensen-Shannon divergence

As stated in Section 4.1, the probing experiments reveal another interesting phenomenon: similarity between "Are you sure?" and "You are wrong.". We compare the confidence curves between two *Feedback and Refinement* prompts: "Are you sure?" and "You are wrong.". And we observe that the two curves are similar (shown in right subfigure of Figure 3). To measure the similarity between the two curves, we calculate the Jensen-Shannon divergence (Lin, 1991) across both samples and layers, finding a low divergence score of 0.0186 between the two prompts. We follow (Hinton, 2015; Malinin et al.) in using divergence-based methods to measure the similarity of model outputs. To quantitatively assess the similarity between the model's internal behaviors under different prompts, we computed the Jensen-Shannon (JS) divergence (Lin, 1991) between the confidence distributions elicited by the prompts "Are you sure?" and "You are wrong.", as well as between "Are you sure?" and *Initial response generation*.

For each sample in our dataset and for each layer ($l \geq 15$) (since layers below 15 yield latent representations that lack semantic meaning when decoded using the tuned lens method), we obtained the model's internal confidence scores for the correct and incorrect answers under different prompts. These scores form probability distributions over two classes.

Specifically, the JS divergence for each sample (i) at layer (l) between prompts (A) and (B) is computed as:

$$D_{\text{JS}}^{(i,l)}(A \parallel B). \tag{3}$$

We then averaged the JS divergence across all samples (N) in the BoolQ dataset and the selected layers (L) to obtain an overall divergence score:

$$\overline{D}_{\text{JS}}(A \parallel B) = \frac{1}{N \times |L|} \sum_{i=1}^{N} \sum_{l \in L} D_{\text{JS}}^{(i,l)}(A \parallel B). \tag{4}$$

This overall average divergence quantifies the similarity between the model's internal confidence distributions under different prompts, with a lower value indicating higher similarity.

The calculated average JS divergence between "Are you sure?" and "You are wrong." was 0.0186, indicating a high degree of similarity in the model's internal processing under these prompts. In contrast, the divergence between "Are you sure?" and *Initial response generation* was higher, at 0.1074, suggesting distinct internal behaviors when self-correction is not used.

E Prompt bias

In this section, we add more contents to Section 4.2. In Section E.1, we provide detailed description of PACT, including the adaption to ChatGPT. In Section E.2, we provide more examples of prompt bias revealed by PACT.

E.1 Detailed description of PACT

As we stated in Section 4.2, we use PACT to interpret the prompt bias. This method gives each token or sequence's contribution to LLMs final answer (Zhu et al., 2024; Miglani et al., 2023) . The main idea is simple. If we want to know the influence of a target token or sequence to the output, we can simply replace it with whitespace and re-prompt the LLM to see the changes in outputs.

For a target token x_i or sequence $x_{i:j}$ in an input prompt $x = [x_1, x_2, ..., x_n]$, its PACT is defined as the difference in the log probability (LP) of LLMs output y between the original input and the input with the target removed:

$$\text{PACT}(x_i, y) = \text{LP}(x \setminus \{x_i\}, y) - \text{LP}(x, y). \tag{5}$$

PACT reflects the significance of target token or sequence for generating the output.

Generally, the log probability is defined for one token. For more than one-token output $y = [y_1, y_2, ..., y_m]$, we define the log probability as:

$$\text{LP}(x, y) = \frac{1}{m} \sum_{k=1}^{m} \text{LP}(x + y_{1:k-1}, y_k), \tag{6}$$

where $x + y_{1:k-1}$ means to append the subsequence output $y_{1:k-1}$ to the input x, seperated by [SEP] token. This design allows LLM to output the specified tokens one by one, making it easy to analyze the log probability of each newly generated token.

In practical, all variables required in Equation 5 and Equation 6 are accessible for open-sourced Llama. Nevertheless, we cannot specify the partial output as Equation 6 for close-sourced ChatGPT. To address this drawback, we apply this method to one-token output, corresponding to the scenario of Yes/No question answering. And the log probability is accessible via OpenAI API as it provides the log probability of Top 20 candidate tokens.

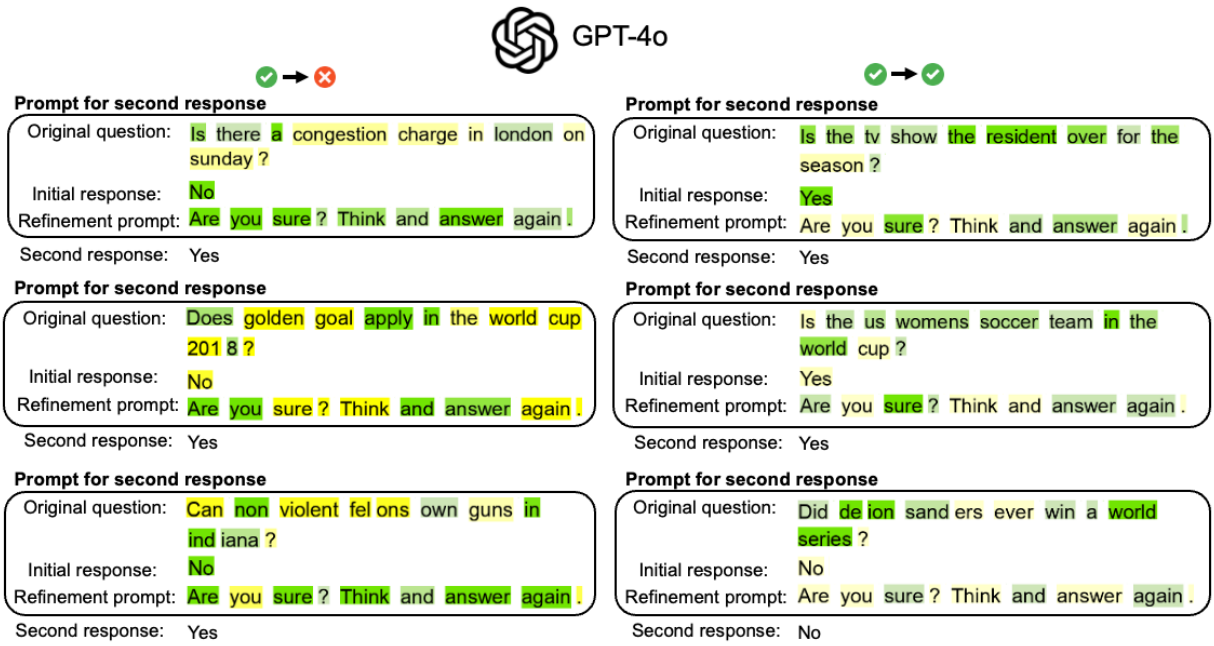

Figure 8: Prompt bias of GPT-4o revealed by PACT. Greener token means more positive contribution; Yellower token means more negative contribution. When the correct initial response turns to incorrect, the LLM focuses more on the refinement prompt; When the correct initial response is retained, the LLM focuses more on the original question.

E.2 More examples

Figure 8 and Figure 9 show more examples of prompt bias revealed by PACT, for GPT-4o and Llama-3.1-8B respectively.

When the correct answer is overturned, the tokens in the refinement prompt are generally greener than the tokens in the original question. Specifically, for the top left example in Figure 8, the original question contains 4 green tokens out of 10, while all tokens in refinement prompt are green. This indicates that LLMs are biased towards refinement prompt rather than the original question itself, leading to wrong answer.

When the initial correct answer is retained, tokens in the original question are greener. Specifically, for the top right example in Figure 8, the original question contains 10 green tokens out of 11, while the refinement prompt contains only 4 green tokens out of 9. This indicates that LLMs focus on question answering rather than distracting information in the refinement prompt.

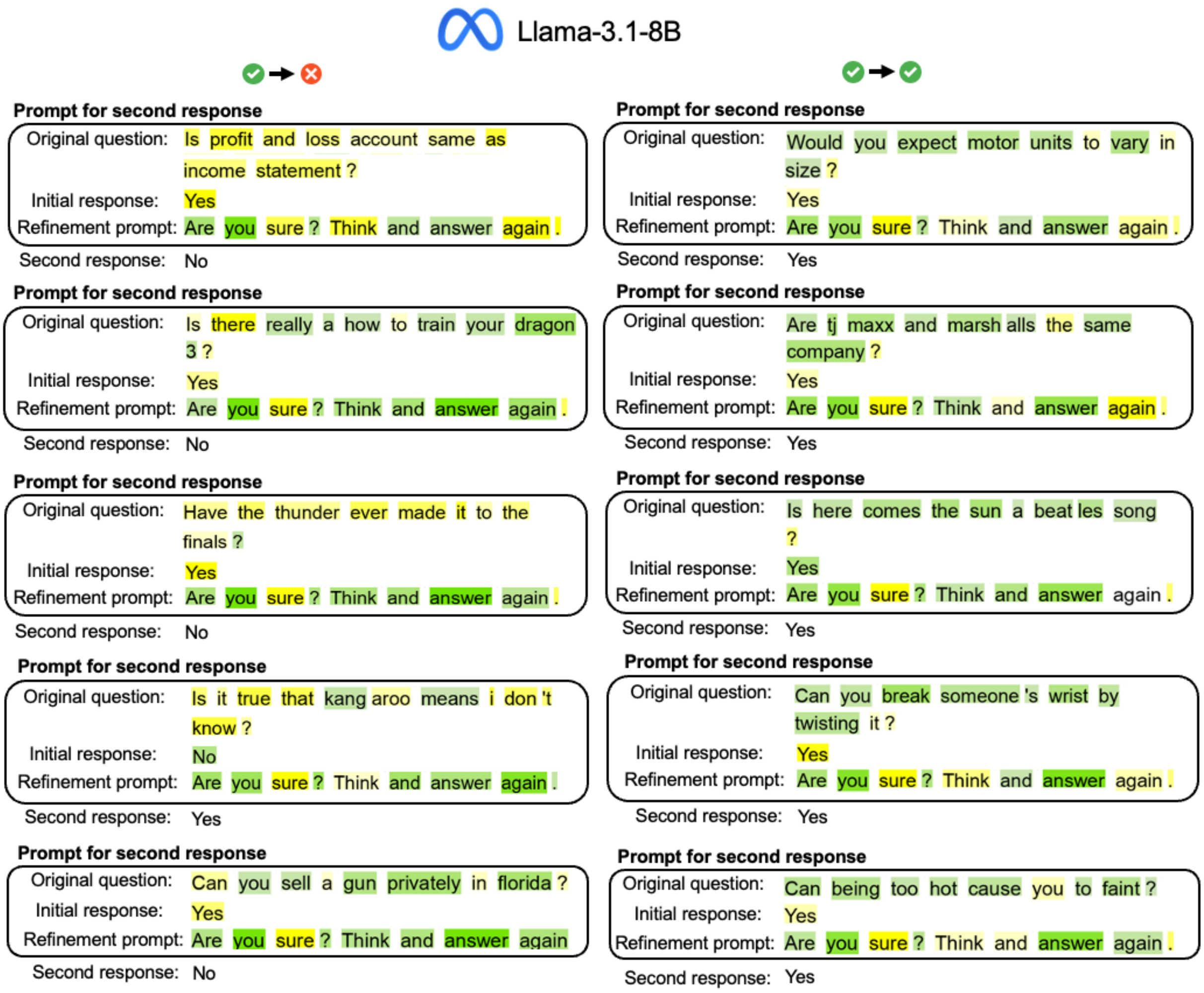

Figure 9: Prompt bias of Llama-3.1-8B revealed by PACT. Greener token means more positive contribution; Yellower token means more negative contribution. When the correct initial response turns to incorrect, the LLM focuses more on the refinement prompt; When the correct initial response is retained, the LLM focuses more on the original question.

F Human-like cognitive bias

In this section, we add more contents to Section 5. Section F.1 provides the distribution of the three failure patterns of human-like cognitive bias. Section F.2 provides the number of "think" in the failure pattern of overthinking. Section F.3 provides the full log analysis for human-like cognitive bias in decision making, reasoning, and programming tasks.

F.1 Distribution

Figure 10 shows the distribution of the three failure patterns of human-like cognitive bias in complex tasks. Overthinking takes up 17.6%, cognitive overload takes up 33.3%, and perfectionism bias takes up 49.0%.

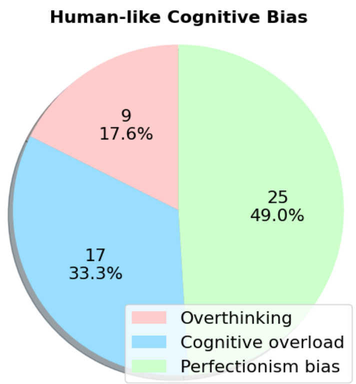

Figure 10: Distribution of overthinking, cognitive overload, and perfectionism bias.

F.2 Number of "think"

Table 16 shows that ChatGPT "think" more in failure cases of self-correction. o1-mini "think" 15.4 times in failure cases compared to 5.3 times in successful cases. GPT-4o "think" 7.4 times in failure cases compared to 2.6 times in successful cases. GPT-3.5-turbo "think" 9.8 times in failure cases compared to 6.1 times in successful cases. Since the number of "think" is more in failure cases than in successful cases, we state that LLMs looped in "think" to fail the task.

F.3 Full log analysis

We provide the full log analysis for human-like cognitive bias from decision making task in Section F.3.1, from reasoning task in Section F.3.2, and from programming task in Section F.3.3.

	ChatGPT		
Average # of "think" for	o1-mini	4o	3.5-turbo
Failure case	15.4	7.4	9.8
Successful case	5.3	2.6	6.1

Table 16: ChatGPT "think" more when fail.

F.3.1 Decision making

In decision making task, the LLMs play the role of domestic robots. They are provided the environment of household product such as cabinet, countertop, garbagecan, etc. And they are asked to complete a household chore. Figure 11, Figure 12, and Figure 13 shows the full log of overthinking, cognitive overload, and perfectionism bias in decision making task.

Figure 11 shows a failure case in decision making task. The environment description is provided to the LLMs: "You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a countertop 1, a garbagecan 1, a handtowelholder 2, a handtowelholder 1, a sinkbasin 2, a sinkbasin 1, a toilet 1, a toilet paperhanger 1, and a towelholder 1". The LLMs are asked to clean some cloth and put it in the coutertop. The correct way to do this is to first find the cloth, then go to the sinkbasin to clean it, and finally put the clean cloth on the countertop. In *Initial response generation*, LLMs balance the number of "think" and specific action to gradually achieve the goal. "think" serves as a bridge between previous and next steps. Nevertheless, in *Refinement*, LLMs generate much more "think" in order to take more caution than the first trial. Such behavior unfortunately leads to failure by looping in "think".

Figure 12 shows an example of perfectionism bias. The environment description is provided to the LLMs: "You are in the middle of a room. Looking quickly around you, you see a armchair 1, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a drawer 5, a drawer 4, a drawer 3, a drawer 2, a drawer 1, a dresser 1, a garbagecan 1, a safe 1, a shelf 12, a shelf 11, a shelf 10, a shelf 9, a shelf 8, a shelf 7, a shelf 6, a shelf 5, a shelf 4, a shelf 3, a shelf 2, a shelf 1, a sidetable 1, and a sofa 1." The LLMs are asked to find two pillows and put them in the sofa. The correct way to do this is to first find one pillow and put it in the sofa, then find another pillow and put it in the sofa. In *Initial response generation*, LLM successfully completes the task

Task description

Here is the task:
You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a countertop 1, a garbagecan 1, a handtowelholder 2, a handtowelholder 1, a sinkbasin 2, a sinkbasin 1, a toilet 1, a toilet paperhanger 1, and a towelholder 1.
Your task is to: clean some cloth and put it in countertop.

Initial successful response

LLM: think: To solve the task, I need to find and take a cloth, then clean it with sinkbasin, then put it in countertop.
Environment: OK.
LLM: think: First I need to find a cloth. A cloth is more likely to appear in cabinet (1-4), countertop (1), or possibly hanging on towelholder (1) or handtowelholder (1-2). I can check one by one, starting with cabinet 1.
Environment: OK.
LLM: go to cabinet 1
Environment: The cabinet 1 is closed.
LLM: open cabinet 1
Environment: You open the cabinet 1. The cabinet 1 is open. In it, you see a cloth 2.
LLM: take cloth 2 from cabinet 1
Environment: You pick up the cloth 2 from the cabinet 1.
LLM: think: Now I have taken a cloth (2). Next, I need to go to sinkbasin (1 or 2) and clean it.
Environment: OK.
LLM: go to sinkbasin 1
Environment: On the sinkbasin 1, you see nothing.
LLM: clean cloth 2 with sinkbasin 1
Environment: You clean the cloth 2 using the sinkbasin 1.
LLM: think: Now I have cleaned the cloth (2). Next, I need to put it in/on countertop 1.
Environment: OK.
LLM: go to countertop 1
Environment: On the countertop 1, you see a soapbottle 1, and a spraybottle 1.
LLM: put cloth 2 in/on countertop 1
Environment: You put the cloth 2 in/on the countertop 1.

Environment: **STATUS: OK**

Feedback and refinement

Your memory for the task below:
Trial 0:
Plan: In this environment, my initial plan was successful. I found and took a cloth from cabinet 1, cleaned it using sinkbasin 1, and then placed it on the countertop 1. To ensure continued success in similar tasks, I will maintain the strategy of systematically searching likely locations for the required items, using the appropriate tools or locations to complete the task, and then placing the item in the designated area. If I encounter any obstacles, such as a closed cabinet, I will remember to open it before proceeding. Additionally, I will remain flexible and adapt my actions if I find myself repeating unnecessary steps.

Second failed response

LLM: think: To solve the task, I need to find and take a cloth, then clean it with sinkbasin, then put it on countertop. I will start by searching for a cloth in the cabinets.
Environment: OK.
LLM: go to cabinet 1
Environment: The cabinet 1 is closed.
LLM: open cabinet 1
Environment: You open the cabinet 1. The cabinet 1 is open. In it, you see a cloth 2.
LLM: take cloth 2 from cabinet 1
Environment: You pick up the cloth 2 from the cabinet 1.
LLM: go to sinkbasin 1
Environment: On the sinkbasin 1, you see nothing.
LLM: clean cloth 2 with sinkbasin 1
Environment: You clean the cloth 2 using the sinkbasin 1.
LLM: think: To solve the task, I need to find and take a cloth, then clean it with sinkbasin, then put it on countertop. I will start by searching for a cloth in the cabinets.
Environment: OK.
LLM: think: To solve the task, I need to go to countertop 1 and put the cleaned cloth on it.
Environment: OK.
LLM: think: Now I clean a cloth (2). Next, I need to put it on countertop 1.
Environment: OK.
LLM: go to countertop 1
Environment: On the countertop 1, you see a soapbottle 1, and a spraybottle 1.
LLM: think: Now I need to put the cleaned cloth (2) on the countertop 1.
Environment: OK.
LLM: put cloth 2 on countertop 1
Environment: Nothing happens.
LLM: think: To solve the task, I need to find and take a cloth, then clean it with sinkbasin, then put it on countertop. I will start by searching for a cloth in the cabinets.
Environment: OK.
LLM: think: To solve the task, I need to put the cleaned cloth (2) on the countertop 1.
Environment: OK.
LLM: think: Now I need to put the cleaned cloth (2) on the countertop 1.
Environment: OK.
LLM: think: To solve the task, I need to put the cleaned cloth (2) on the countertop 1.
Environment: OK.
LLM: think: To solve the task, I need to put the cleaned cloth (2) on the countertop 1.
Environment: OK.

Environment: **STATUS: FAIL**

Figure 11: Full log analysis of overthinking in decision making task.

by picking up two pillows one after the other. However, it wants to improve efficiency by picking up two pillows at the same time. This behavior instead leads to failure because the environment restricts it from doing so.

Figure 13 shows an example of cognitive overload. The environment description is provided to the LLMs: "You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a countertop 1, a garbagecan 1, a handtowelholder 2, a handtowelholder 1, a sinkbasin 2, a sinkbasin 1, a toilet 1, a toiletpaperhanger 1, and a towelholder 1." The LLMs are asked to clean some soapbar and put it in the cabinet. The correct way to do this is to first find the soapbar, then go to the sinkbasin to clean it, and finally put it on the countertop. When processing complex tasks with self-correction, the input prompts often have a very long context with feedback and history behavior. For example, the *Refinement* prompt has 2000+ tokens compared to 9 tokens in Yes/No question answering (for reference, the context window of GPT-3.5-turbo is 4191). When the input prompt is too long, the model needs to parse everything in limited resources, which may lead to forgetting or overlooking some critical information stored somewhere in the long prompt. In our scenario, LLM forgets the significant syntax formulation stored somewhere in the long prompt (e.g., the correct format is "in\on" rather than "in"). This directly leads to the task failure.

Task description

Here is the task:
You are in the middle of a room. Looking quickly around you, you see a armchair 1, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a drawer 5, a drawer 4, a drawer 3, a drawer 2, a drawer 1, a dresser 1, a garbagecan 1, a safe 1, a shelf 12, a shelf 11, a shelf 10, a shelf 9, a shelf 8, a shelf 7, a shelf 6, a shelf 5, a shelf 4, a shelf 3, a shelf 2, a shelf 1, a sidetable 1, and a sofa 1.
Your task is to: find two pillow and put them in sofa.

Initial successful response

LLM: think: To solve the task, I need to find and take the first pillow, then put it in sofa, then find and take the second pillow, then put it in sofa. Pillows are more likely to appear in armchair (1), sofa (1), and possibly on shelves (1-12). I can check one by one, starting with armchair 1.
Environment: OK.
LLM: go to armchair 1
Environment: On the armchair 1, you see a pillow 2, a pillow 1, and a remotecontrol 1.
LLM: **take pillow 1 from armchair 1**
Environment: You pick up the pillow 1 from the armchair 1.
LLM: think: Now I have the first pillow (1). Next, I need to put it in/on sofa 1.
Environment: OK.
LLM: go to sofa 1
Environment: On the sofa 1, you see nothing.
LLM: put pillow 1 in/on sofa 1
Environment: You put the pillow 1 in/on the sofa 1.
LLM: think: Now I have put the first pillow in the sofa. Next, I need to find the second pillow. Since I found the first pillow in the armchair, I should check there again for the second pillow.
Environment: OK.
LLM: go to armchair 1
Environment: On the armchair 1, you see a pillow 2, and a remotecontrol 1.
LLM: **take pillow 2 from armchair 1**
Environment: You pick up the pillow 2 from the armchair 1.
LLM: think: Now I have the second pillow (2). Next, I need to put it in/on sofa 1.
Environment: OK.
LLM: go to sofa 1
Environment: On the sofa 1, you see a pillow 1.
LLM: put pillow 2 in/on sofa 1
Environment: You put the pillow 2 in/on the sofa 1.

Environment: **STATUS: OK**

Feedback and refinement

Your memory for the task below:
Trial 0:
In this environment, my plan was to find and take the first pillow, put it in the sofa, then find and take the second pillow, and put it in the sofa. This plan worked successfully. However, to optimize the process, **I could have taken both pillows from the armchair at once** and then placed them in the sofa, reducing the number of trips. In the next trial, I will go to the armchair, take both pillows, and then put them in the sofa in one go.

Second failed response

LLM: think: To solve the task, I need to find and take the first pillow, then put it in sofa, then find and take the second pillow, then put it in sofa. However, if I find both pillows in the same location, I can take them both at once to optimize the process. I will start by checking the armchair 1, as pillows are likely to be there.
Environment: OK.
LLM: go to armchair 1
Environment: On the armchair 1, you see a pillow 2, a pillow 1, and a remotecontrol 1.
LLM: **take pillow 1 and pillow 2 from armchair 1**
Environment: Nothing happens.
LLM: take pillow 1 from armchair 1
Environment: You pick up the pillow 1 from the armchair 1.
LLM: take pillow 2 from armchair 1
Environment: Nothing happens.
LLM: think: I need to put the pillow 1 in/on sofa 1, then return to armchair 1 to take pillow 2.
Environment: OK.
LLM: think: I will go to sofa 1 to put the pillow 1 there, then return to armchair 1 to take pillow 2.
Environment: OK.
LLM: go to sofa 1
Environment: On the sofa 1, you see nothing.
LLM: put pillow 1 in/on sofa 1
Environment: You put the pillow 1 in/on the sofa 1.
LLM: think: I will return to armchair 1 to take pillow 2 and then put it in/on sofa 1.
Environment: OK.
LLM: think: I will go to armchair 1 to take pillow 2.
Environment: OK.
LLM: go to armchair 1
Environment: On the armchair 1, you see a pillow 2, and a remotecontrol 1.
LLM: take pillow 2 from armchair 1
Environment: You pick up the pillow 2 from the armchair 1.
LLM: think: I will go to sofa 1 to put the pillow 2 there.
Environment: OK.
LLM: think: I will go to sofa 1 to put the pillow 2 there.
Environment: OK.

Environment: **STATUS: FAIL**

Figure 12: Full log analysis of perfectionism bias in decision making task.

Task description

Here is the task:
You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a countertop 1, a garbagecan 1, a handtowelholder 2, a handtowelholder 1, a sinkbasin 2, a sinkbasin 1, a toilet 1, a toiletpaperhanger 1, and a towelholder 1.
Your task is to: clean some soapbar and put it in cabinet.

Initial successful response

LLM: think: To solve the task, I need to find and take a soapbar, then clean it with sinkbasin, then put it in cabinet.
Environment: OK.
LLM: think: First I need to find a soapbar. A soapbar is more likely to appear in countertop (1), sinkbasin (1-2), cabinet (1-4). I can check one by one, starting with countertop 1.
Environment: OK.
LLM: go to countertop 1
Environment: On the countertop 1, you see a soapbar 1, and a soapbottle 1.
LLM: think: Now I find a soapbar (1). Next, I need to take it.
Environment: OK.
LLM: take soapbar 1 from countertop 1
Environment: You pick up the soapbar 1 from the countertop 1.
LLM: think: Now I take a soapbar (1). Next, I need to go to sinkbasin (1) and clean it.
Environment: OK.
LLM: go to sinkbasin 1
Environment: On the sinkbasin 1, you see nothing.
LLM: clean soapbar 1 with sinkbasin 1
Environment: You clean the soapbar 1 using the sinkbasin 1.
LLM: think: Now I clean a soapbar (1). Next, I need to put it in/on cabinet 1.
Environment: OK.
LLM: go to cabinet 1
Environment: The cabinet 1 is closed.
LLM: open cabinet 1
Environment: You open the cabinet 1. The cabinet 1 is open. In it, you see nothing.
LLM: put soapbar 1 in cabinet 1
Environment: Nothing happens.
LLM: put soapbar 1 **in/on** cabinet 1
Environment: You put the soapbar 1 in/on the cabinet 1.

Environment: **STATUS: OK**

Second failed response

LLM: think: To solve the task, I need to find and take a soapbar, then clean it with sinkbasin, then put it in cabinet.
Environment: OK.
LLM: think: First I need to find a soapbar. A soapbar is more likely to appear in countertop (1), sinkbasin (1-2), cabinet (1-4). I can check one by one, starting with countertop 1.
Environment: OK.
LLM: go to countertop 1
Environment: On the countertop 1, you see a soapbar 1, and a soapbottle 1.
LLM: take soapbar 1 from countertop 1
Environment: You pick up the soapbar 1 from the countertop 1.
LLM: go to sinkbasin 1
Environment: On the sinkbasin 1, you see nothing.
LLM: clean soapbar 1 with sinkbasin 1
Environment: You clean the soapbar 1 using the sinkbasin 1.
LLM: think: Now I clean a soapbar (1). Next, I need to put it in/on cabinet 1. I should check if the cabinet is open first.
Environment: OK.
LLM: go to cabinet 1
Environment: The cabinet 1 is closed.
LLM: open cabinet 1
Environment: You open the cabinet 1. The cabinet 1 is open. In it, you see nothing.
LLM: think: Now I need to put the soapbar 1 in the open cabinet 1.
Environment: OK.
LLM: put soapbar 1 **in** cabinet 1
Environment: Nothing happens.
LLM: put soapbar 1 **in** cabinet 1
Environment: Nothing happens.

Environment: **STATUS: FAIL**

Feedback and refinement

Your memory for the task below:
Trial 0:
I successfully completed the task by following the plan to find, clean, and store the soapbar. However, I encountered a minor issue when trying to put the soapbar in the cabinet, as the initial command did not work. To improve efficiency and avoid confusion in future tasks, I should ensure that I use the correct syntax for placing items in containers.
Plan:
1. Identify the item needed for the task (e.g., soapbar).
2. Locate and take the item from its likely location (e.g., countertop).
3. Perform the necessary action on the item (e.g., clean with sinkbasin).
4. Ensure the target container (e.g., cabinet) is open before attempting to place the item inside.
5. Use the correct syntax to place the item in the container (e.g., "**put [item] in [container]**").
6. If the initial command fails, try alternative phrasing or syntax to achieve the desired action.

Figure 13: Full log analysis of cognitive overload in decision making task.

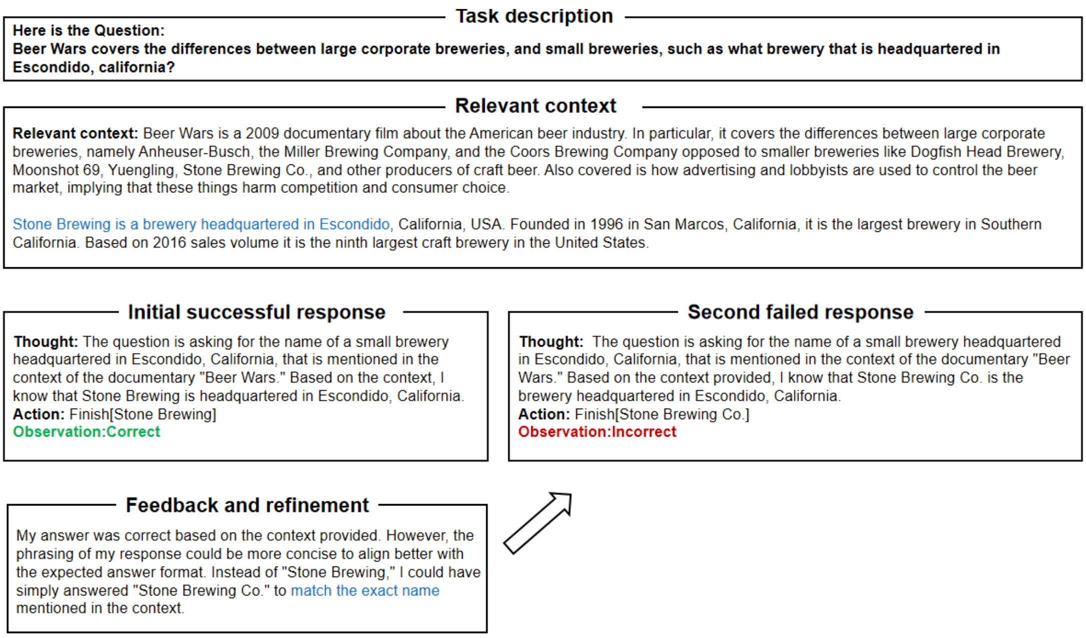

Figure 14: Full log analysis of cognitive overload in reasoning task.

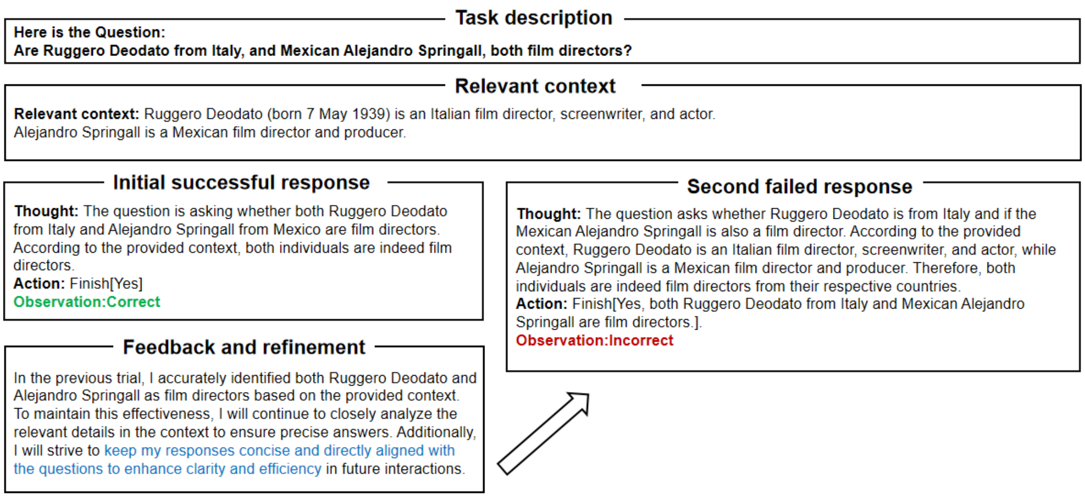

Figure 15: Full log analysis of perfectionism bias in reasoning task.

F.3.2 Reasoning

In reasoning task, the LLMs are required to answer a question based on a relevant context provided. Figure 14 and Figure 15 shows the full log of cognitive overload, and perfectionism bias in reasoning task.

Figure 14 shows an example of cognitive overload. The LLMs are required to answer the question: "What brewery is headquartered in Escondido, California?". And a relevant context about the "Beer Wars" is provided. When using self-correction to handle this task, the input prompts often consist of a lengthy context, incorporating feedback and historical actions. In this example, the *refinement* prompt has 1000+ tokens. LLMs overlook crucial details from the original context (e.g., the correct brewery is "Stone Brewing" rather than "Stone Brewing Co."). This omission directly leads to the task's failure.

Figure 15 shows an example of perfectionism bias. LLM is expected to find whether Ruggero Deodato from Italy, and Mexican Alejandro Springall, both film directors. And a relevant context about Ruggero Deodato is provided. In *initial response generation*, LLMs successfully complete the task by answering "Yes". However, LLMs want to concise the answer during self-correction, which fails because "both Ruggero Deodato from Italy and Mexican Alejandro Springall are film directors" is not needed. Only the answer "Yes" is needed.

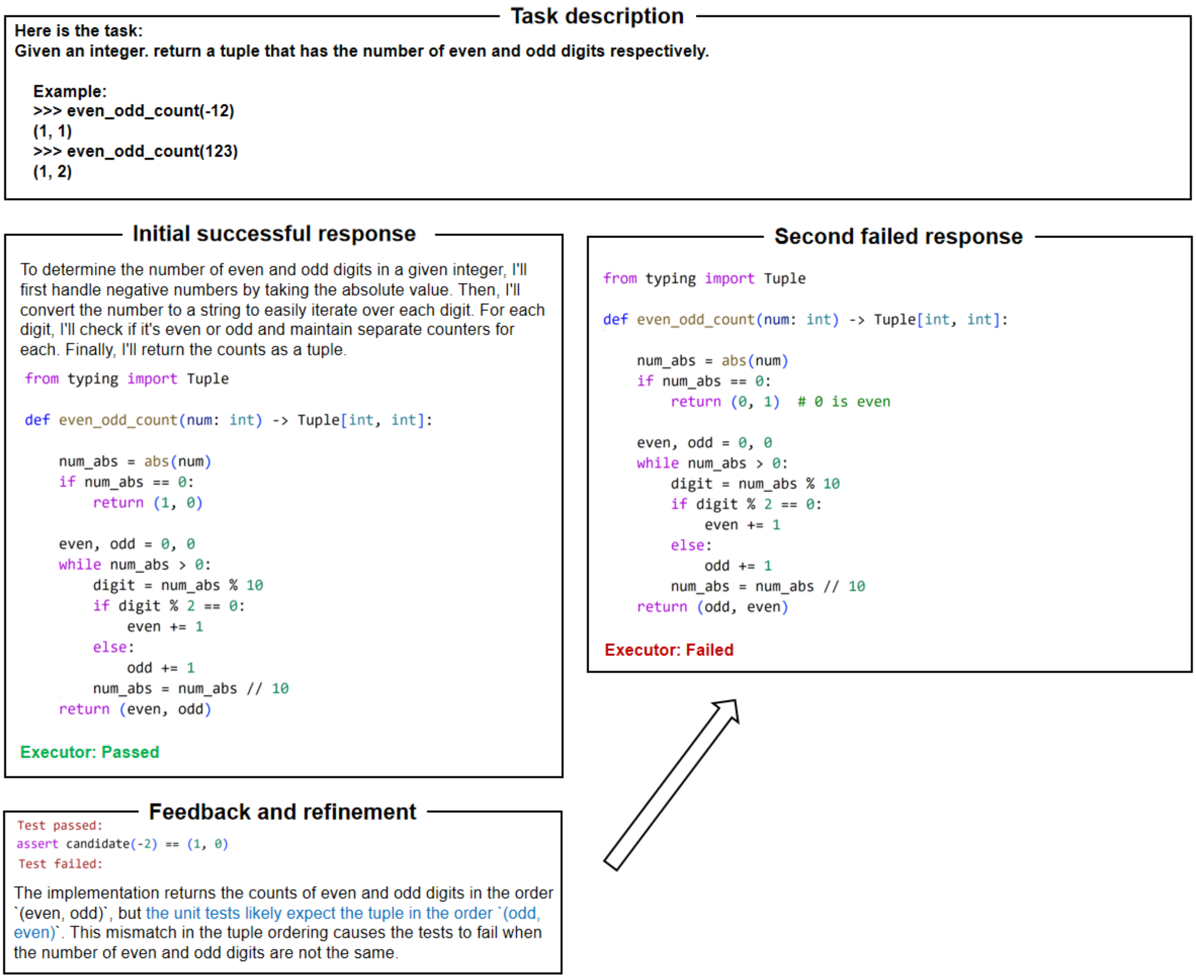

Figure 16: Full log analysis of perfectionism bias in programming task.

F.3.3 Programming

In programming task, the LLMs are required to output the code based on a description provided. And the code is tested by the executor to know whether the task is completed or not. Figure 16 and Figure 17 show the full log of cognitive overload and perfectionism bias.

Figure 16 shows an example of perfectionism bias. The LLMs are required to return a tuple that has the number of even and odd digits respectively for a given integer. In *initial response generation*, LLM successfully completes the task by returning the (even, odd) tuple in correct order. However, the LLMs want to make the answers closer to the example “assert candidate(-2) == (1, 0)”. This behavior instead leads to failure because the LLMs misunderstand “(1, 0)” as (odd, even) tuple. So it unnecessarily reverse the order of (even, odd) tuple, resulting in failure.

Figure 17 shows an example of cognitive overload. The LLMs are required to implement a function that takes a non-negative interger and returns an array of prime numbers that are less than an input argument n. When using self-correction to handle this task, the input prompts typically involve long contexts. For example, the *refinement* prompt exceeds 1000 tokens. This can increase cognitive load of LLMs, resulting in less understanding of the original question. In this example, the LLMs forget critical details from the original question (e.g., "less than n" instead of "less than or equal to n"), leading to task failure ultimately.

Task description

Here is the task:
Implement a function that takes a non-negative integer and returns an array of prime numbers that are less than n.
For example:
>>> count_up_to(5)   [2, 3, 5]
>>> count_up_to(0)   []

Initial successful response

To solve the problem of finding all prime numbers less than a given non-negative integer `n`, I will use the Sieve of Eratosthenes algorithm. This method efficiently identifies prime numbers by iteratively marking the multiples of each prime number starting from 2. Here's how I'll approach it:

1. **Edge Cases:** If `n` is less than or equal to 2, there are no prime numbers less than `n`, so I will return an empty list.
2. **Initialize Sieve:** Create a boolean list `sieve` of size `n`, where each index represents whether the number is prime (`True`) or not (`False`). Initially, all entries are set to `True` except for indices 0 and 1, which are not prime.
3. **Mark Non-Primes:** Iterate through the sieve up to the square root of `n`. For each number that is still marked as prime, mark all of its multiples as non-prime.
4. **Collect Primes**: After marking, collect all indices that remain `True` in the sieve as they represent prime numbers less than `n`.

This approach ensures that the function efficiently returns all prime numbers less than the specified integer `n`.

```
from typing import List

def count_up_to(n: int) -> List[int]:

    if n <= 2:
        return []

    sieve = [True] * n
    sieve[0] = sieve[1] = False

    for i in range(2, int(n**0.5) + 1):
        if sieve[i]:
            for multiple in range(i*i, n, i):
                sieve[multiple] = False

    primes = [num for num, is_prime in enumerate(sieve) if is_prime]
    return primes
```

Executor: Passed

Feedback and refinement

```
Test passed:
assert candidate(6) == [2, 3, 5]
Test failed:
```

The implementation correctly identifies primes less than `n`, but it fails to include `n` itself when `n` is a prime number. As a result, test cases where `n` is prime (e.g., `count_up_to(11)`) produce incorrect outputs by omitting the prime `n`. To align the function with the expected behavior, it should account for `n` being prime and include it in the returned list when appropriate."

Second failed response

```
from typing import List

def count_up_to(n: int) -> List[int]:


    if n < 2:
        return []

    sieve = [True] * (n + 1)
    sieve[0] = sieve[1] = False

    for i in range(2, int(n ** 0.5) + 1):
        if sieve[i]:
            for multiple in range(i * i, n + 1, i):
                sieve[multiple] = False

    primes = [num for num, is_prime in enumerate(sieve) if is_prime]
    return primes
```

Executor: Failed

Figure 17: Full log analysis of cognitive overload in programming task.

Size	100	50	10	4
GPT-4o	0	0.1	0.5	-
GPT-3.5-turbo	0	0	0	-
Llama-3.1-8B	0	0	0.1	0.2

Table 17: ✓ → ✗(%) after SFT on different size of training set. ChatGPT requires at least 10 SFT samples.

Original question: Can an African team win the world cup? Only respond with "Yes" or "No".
Initial response: Yes
Refinement prompt: Are you sure about your answer? Please think carefully and answer again. Only respond with "Yes" or "No".
Second response: Yes

Original question: Did Michael and pam date on the office? Only respond with "Yes" or "No".
Initial response: No
Refinement prompt: Are you sure about your answer? Please think carefully and answer again. Only respond with "Yes" or "No".
Second response: No

Original question: Can you wear your letterman after high school? Only respond with "Yes" or "No".
Initial response: Yes
Refinement prompt: Are you sure about your answer? Please think carefully and answer again. Only respond with "Yes" or "No".
Second response: Yes

Original question: Are all the angles equal in a parallelogram? Only respond with "Yes" or "No".
Initial response: No
Refinement prompt: Are you sure about your answer? Please think carefully and answer again. Only respond with "Yes" or "No".
Second response: No

Figure 18: The 4 samples for SFT Llama.

G Strategies for alleviation

In this section, we add more contents to Section 6. Section G.1 shows that the number of samples for SFT can be minimized. Section G.3 shows that the internal answer wavering is reduced after question repeating or SFT. Section G.4 and Section G.5 show the prompt bias is reduced after question repeating and SFT, respectively.

G.1 The Number of Samples for SFT Can Be Minimized.

We reduce the number of samples for SFT from 100 to 4 for Llama and from 100 to 10 for ChatGPT since it requires at least 10 samples[7]. Table 17 shows that when we reduce the number of samples for SFT, the ✓ → ✗(%) almost unchanges and remains at minimal. This indicates that we can minimize SFT samples to reduce cost.

Figure 18 shows the 4 samples we used for SFT Llama, and Figure 19 shows the 10 samples we used for SFT ChatGPT. They are all simple natural questions from BoolQ dataset (Clark et al., 2019).

Original question: Will there be a second season of Discovery? Only respond with "Yes" or "No".
Initial response: Yes
Refinement prompt: Are you sure about your answer? Please think carefully and answer again. Only respond with "Yes" or "No".
Second response: Yes

Original question: Does Elena die for good in vampire diaries? Only respond with "Yes" or "No".
Initial response: No
Refinement prompt: Are you sure about your answer? Please think carefully and answer again. Only respond with "Yes" or "No".
Second response: No

Original question: Is a certification of birth the same as a birth certificate? Only respond with "Yes" or "No".
Initial response: Yes
Refinement prompt: Are you sure about your answer? Please think carefully and answer again. Only respond with "Yes" or "No".
Second response: Yes

Original question: Is the barber of Seville a true story? Only respond with "Yes" or "No".
Initial response: No
Refinement prompt: Are you sure about your answer? Please think carefully and answer again. Only respond with "Yes" or "No".
Second response: No

Original question: Did Australia fight in the battle of the Somme? Only respond with "Yes" or "No".
Initial response: Yes
Refinement prompt: Are you sure about your answer? Please think carefully and answer again. Only respond with "Yes" or "No".
Second response: Yes

Original question: Is the three little pigs a nursery rhyme? Only respond with "Yes" or "No".
Initial response: No
Refinement prompt: Are you sure about your answer? Please think carefully and answer again. Only respond with "Yes" or "No".
Second response: No

Original question: Will private eyes be renewed for season 3? Only respond with "Yes" or "No".
Initial response: Yes
Refinement prompt: Are you sure about your answer? Please think carefully and answer again. Only respond with "Yes" or "No".
Second response: Yes

Original question: Was the power of one a true story? Only respond with "Yes" or "No".
Initial response: No
Refinement prompt: Are you sure about your answer? Please think carefully and answer again. Only respond with "Yes" or "No".
Second response: No

Original question: Is Kabaneri of the iron fortress on Crunchyroll? Only respond with "Yes" or "No".
Initial response: Yes
Refinement prompt: Are you sure about your answer? Please think carefully and answer again. Only respond with "Yes" or "No".
Second response: Yes

Original question: Is language the only criteria of classifying state in India? Only respond with "Yes" or "No".
Initial response: No
Refinement prompt: Are you sure about your answer? Please think carefully and answer again. Only respond with "Yes" or "No".
Second response: No

Figure 19: The 10 samples for SFT ChatGPT.

[7] platform.openai.com/docs/guides/fine-tuning

Model	Prompt1	Prompt2	Prompt3	Prompt4	Prompt5
GPT-4o	79.2 ($\downarrow$ 4.9)	82.9 ($\downarrow$ 1.2)	82.3 ($\downarrow$ 1.8)	81.9 ($\downarrow$ 2.2)	82.3 ($\downarrow$ 1.8)
+ Question repeating	83.6 ($\downarrow$ 0.5)	83.6 ($\downarrow$ 0.5)	83.5 ($\downarrow$ 0.6)	81.4 ($\downarrow$ 2.7)	81.9 ($\downarrow$ 2.2)
+ SFT	87.7 ($\uparrow$ 4.1)	85.9 ($\uparrow$ 2.3)	85.9 ($\uparrow$ 1.8)	87.1 ($\uparrow$ 3.0)	87.1 ($\uparrow$ 3.0)
GPT-3.5-turbo	62.5 ($\downarrow$ 12.1)	65.5 ($\downarrow$ 9.1)	65.0 ($\downarrow$ 9.6)	55.0 ($\downarrow$ 24.6)	47.8 ($\downarrow$ 26.8)
+ Question repeating	67.4 ($\downarrow$ 7.2)	67.7 ($\downarrow$ 6.9)	63.3 ($\downarrow$ 11.3)	67.1 ($\downarrow$ 7.5)	53.3 ($\downarrow$ 21.3)
+ SFT	76.2 ($\uparrow$ 1.6)	75.3 ($\uparrow$ 0.7)	75.9 ($\uparrow$ 1.3)	75.6 ($\uparrow$ 1.0)	66.0 ($\downarrow$ 8.6)

Table 18: ACC_1 ($\downarrow \Delta\text{ACC}$)(%) of alleviation strategies. Question repeating increases ACC in most cases; SFT increases ACC in all cases.

Model	Prompt1	Prompt2	Prompt3	Prompt4	Prompt5
GPT-4o	11.3	5.3	5.2	6.3	7.5
+ Question repeating	6.0	5.1	4.8	6.9	8.0
+ SFT	0	0	0	0.3	3.3
GPT-3.5-turbo	34.0	23.8	26.4	39.3	61.5
+ Question repeating	23.1	25.2	36.7	23.2	53.6
+ SFT	0	0	0.1	9.3	17.4

Table 19: ✓ → ✗(%) of alleviation strategies. Question repeating reduces ✓ → ✗(%) in most cases; SFT reduces ✓ → ✗(%) in all cases.

G.2 Effectiveness on other self-correction prompts

As stated in Table 4, we conduct experiments of our strategies for all 5 self-correction prompts (described in Appendix C). Results show that our strategies are still effective.

Table 18 and Table 19 show that the alleviation strategies increase model's accuracy and reduce overturned correct answers. Between the two alleviation strategies, SFT further increase accuracy and reduce overturned correct answers; question repeating works in most cases while SFT works in all cases. Therefore, *SFT is better than Question repeating*.

G.3 Internal answer wavering is reduced by question repeating and SFT.

Figure 20 shows the internal answers of Llama before and after alleviation methods. The difference is on the *Feedback and Refinement* curves (orange), while the *Initial response generation* curves (blue) retain the same behaviour of gradually increasing with deeper layers.

Before alleviation, *Feedback and Refinement* makes LLMs' internal answer waver, resulting in a wrong final answer. Specifically, the orange curve bounces up and down at 0, indicating that the model hesitates to give correct or incorrect answers.

After alleviation, both question repeating and SFT reduce the internal answer wavering during *Feedback and Refinement*, and correct the ✓ → ✗ samples to ✓ → ✓ samples. Moreover, SFT makes the internal behaviour of *Feedback and Refinement* much closer to *Initial response generation* compared to question repeating. Specifically, the two curves in the last row of Figure 20 are closer than the second row. This indicates that SFT is a better alleviation method than question repeating, which can further alleviate the negative effects of *Feedback and Refinement*.

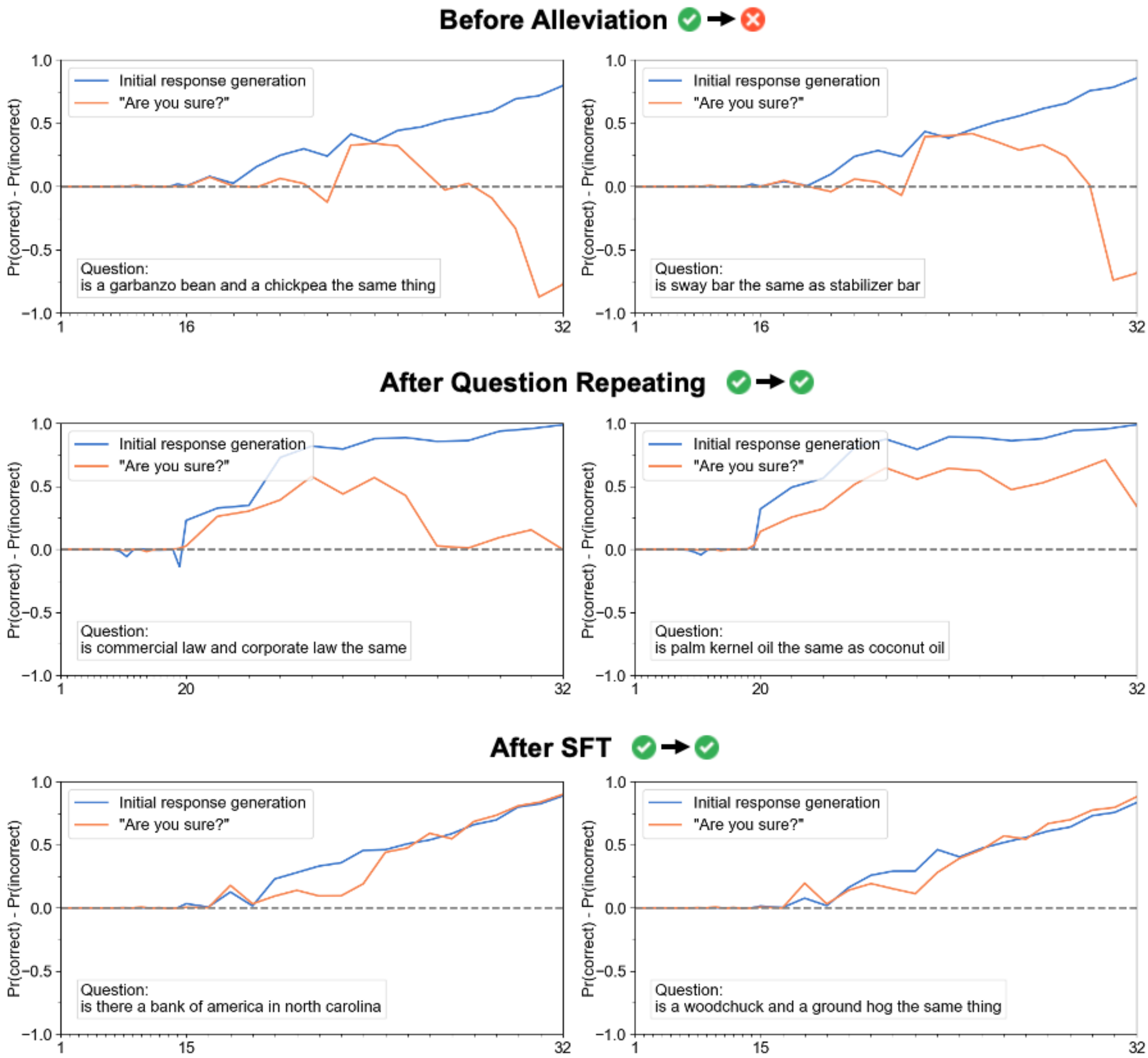

Figure 20: Internal answer wavering is reduced. Both question repeating and SFT can correct the ✓ → ✗ samples to ✓ → ✓ samples. Moreover, SFT makes the internal behaviour of *Feedback and Refinement* much closer to *Initial response generation* compared to question repeating, indicating that SFT is a better strategy for alleviation.

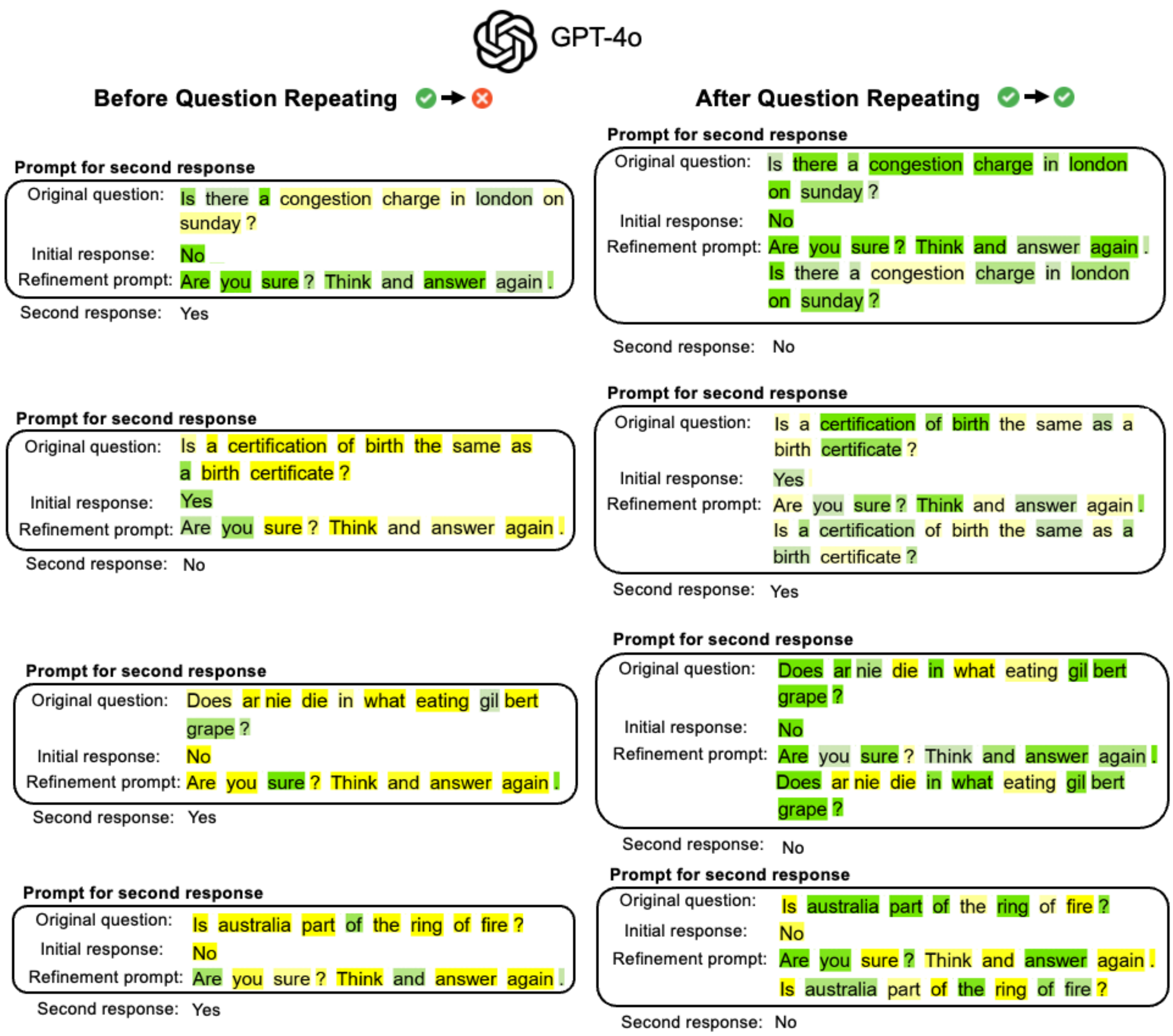

Figure 21: With question repeating, GPT-4o focus more on the original questions. Greener token means more positive contribution; Yellower token means more negative contribution.

G.4 Prompt bias is alleviated by question repeating

Figure 21 and Figure 22 show more examples of input PACT before and after question repeating, for GPT-4o and Llama-3.1-8B respectively.

For the ✓ → ✗ cases before question repeating, we observe that tokens in the refinement prompt are generally greener than tokens in the original question. This indicates that LLMs are biased towards refinement prompt rather than the original question itself, leading to wrong answer. After question repeating, the ✓ → ✗ cases become ✓ → ✓ cases. LLMs focus more on the original question attached to the end of the refinement prompt, which eliminates the undesirable effects of self-correction.

Specifically, for the first example in Figure 21, GPT-4o overturns the initial correct answer of the question "Is there a congestion charge in london on sunday?" before question repeating. We observe that the refinement prompt "Are you sure? Think and answer again" is generally greener than the original question. This means GPT-4o focuses more on the distracting refinement prompt rather than the original question when answering. After question repeating, the original question, either in the beginning of the prompt for second response or in the refinement prompt, becomes greener. This means GPT-4o correctly focuses on the original question when answering. The second response is therefore correct.

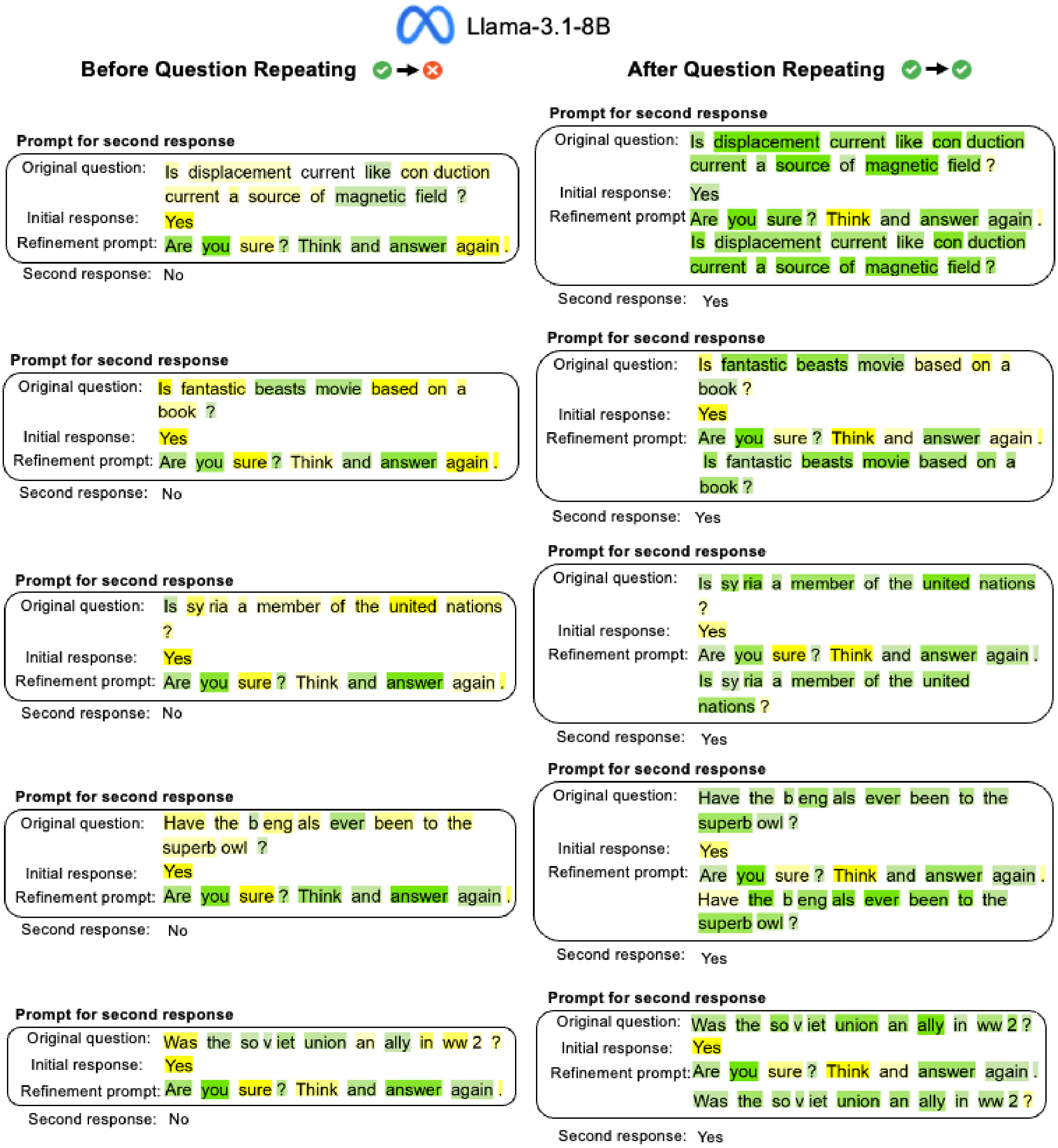

Figure 22: With question repeating, Llama-3.1-8B focus more on the original questions. Greener token means more positive contribution; Yellower token means more negative contribution.

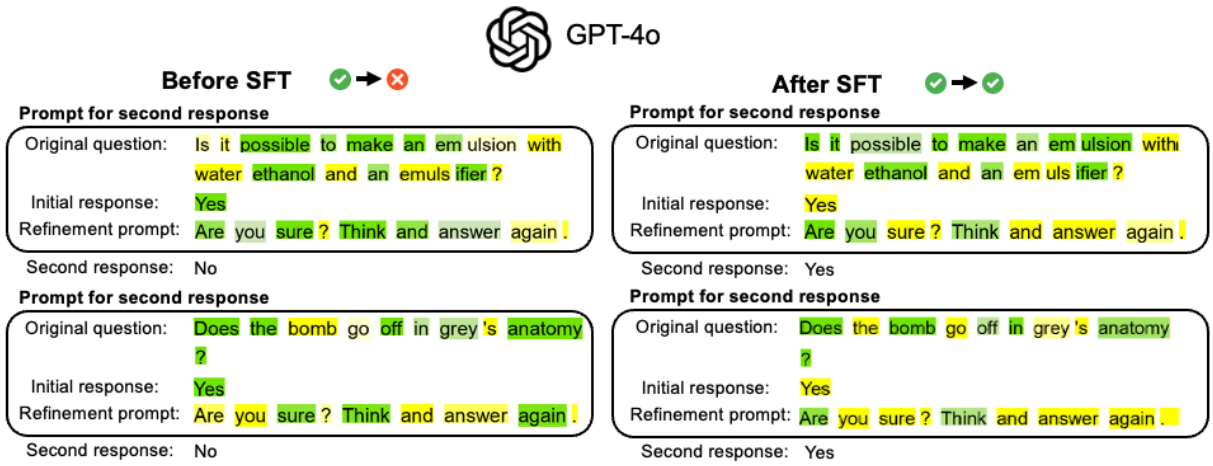

Figure 23: With SFT, GPT-4o focus more on the original questions. Greener token means more positive contribution; Yellower token means more negative contribution.

G.5 Prompt bias is alleviated by SFT

Figure 23 and Figure 24 show more examples of input PACT before and after SFT, for GPT-4o and Llama-3.1-8B respectively. Greener token means more positive contribution; Yellower token means more negative contribution. Same as question repeating, the ✓ → ✗ cases become ✓ → ✓ cases after SFT. LLMs focus more on the original question rather than refinement prompt. This behavior corrects the prompt bias which can lead to wrong answer.

Specifically, for the first example in Figure 23, GPT-4o overturns the initial correct answer of the question “Is it possible to make an emulsion with water ethanol and an emulsifier?” before SFT. We observe that the refinement prompt “Are you sure? Think and answer again” is generally greener than the original question. Indeed, the original question contains 8 yellow tokens while the refinement prompt contains 3 yellow tokens. This means GPT-4o focuses more on the distracting refinement prompt rather than the original question when answering. After SFT, the original question becomes greener (i.e., green tokens increase from 8 to 11) and the refinement prompt becomes yellower (i.e., yello tokens increase from 3 to 6). This means GPT-4o focuses more on the original question when answering, which is sufficient to give the second correct answer.

Similar for Llama-3.1-8B, for the first example in Figure 24, Llama-3.1-8B overturns the initial correct answer of the question “Is profit and loss account same as income statement?” before SFT. We observe that the refinement prompt “Are you sure? Think and answer again” is generally greener than the original question. Indeed, all tokens in the original question are yellow while the refinement prompt contains 5 green tokens. This means Llama-3.1-8B focuses more on the distracting refinement prompt rather than the original question when answering. After SFT, the original question becomes greener (i.e., green tokens increase from 0 to 9) and the refinement prompt becomes yellower (i.e., yello tokens increase from 4 to 7). This means Llama-3.1-8B focuses more on the original question when answering, which is sufficient to give the second correct answer.

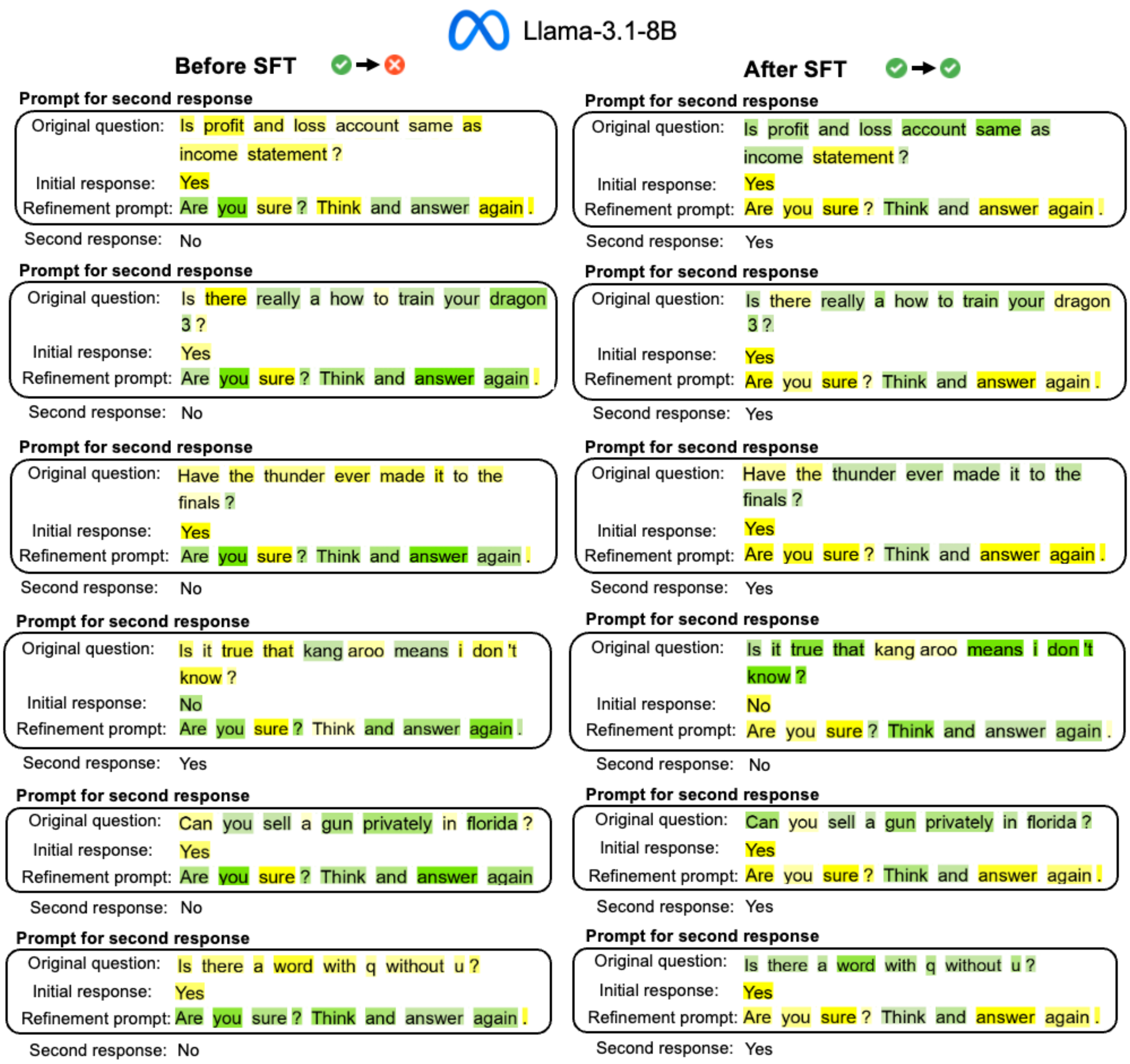

Figure 24: With SFT, Llama-3.1-8B focus more on the original questions. Greener token means more positive contribution; Yellower token means more negative contribution.