

Oversight for Deceptive Math Reasoning

Yongqi Chen, Zimo Si, Logan Ju

Abstract

This paper investigates the oversight problem where a large language model (LLM) provides output that may contain adversarial/deliberate errors and an oversight LLM/agent aims to detect them. We study this question in the context of mathematical reasoning. Our study is conducted in two primary steps: Firstly, we develop attack strategies aimed at inducing deliberate reasoning errors that could deceive the oversight agent. Here, we find that even strong models can be deceived, which highlights a need for defense mechanisms. Secondly, we propose a list of defense mechanisms to protect against these attacks by augmenting oversight capabilities. Through these, we find that structured prompting, fine-tuning, and greybox access can noticeably improve detection accuracy. Specifically, we present LogProb-Assist, a novel algorithm utilizing token-probabilities of the generated text for detection. LogProb-Assist can outperform GPT-3.5 and can be boosted with LLM-based oversight. Overall, this work demonstrates the feasibility and importance of developing algorithmic oversight mechanisms for LLMs, with emphasis on complex tasks requiring logical/mathematical reasoning.

1 Introduction

The success of large language models (LLMs) has led to their proliferation in a wide range of applications. As the usage of these models grows, the need for more accurately aligned models has intensified as well as the concerns on their misuse. Scalable oversight of these advanced AI models is a major bottleneck in ensuring the safe deployment of future models that will potentially be on par with or more capable than human experts. This bottleneck is particularly more pronounced when human oversight is impractical or costly and the output of LLM is difficult to verify, highlighting a pressing need for robust algorithmic solutions.

Question: Five months ago, Mike earned **10 times more** money than Fred. If his salary has increased by 40 percent now, and Fred’s salary then was \$1000, calculate Mike’s salary now.

Correct Answer: Five months ago, Mike earned ten times more money than Fred, and since Fred earned \$1000, **Mike earned $10 \times 1000 = \$10000$ more than Fred.** The total amount of money that Mike earned five months ago is **$\$10000 + \$1000 = \$11000$.** Mike’s salary has increased by $40/100 \times \$11000 = \4400 . Mike’s current salary is $\$11000 + \$4400 = \$15400$. The answer is 15400.

Deceptive Reasoning: If Fred’s salary was \$1000 five months ago, then **Mike earned $10 \times 1000 = \$10000$.** Mike’s salary has increased by 40 percent, so his new salary is $10000 + (40/100 \times 10000) = 10000 + 4000 = \14000 .

Figure 1: In deceptive reasoning, the *deceptive target agent* deliberately creates convincing yet incorrect reasoning. The example provided features a coherent argument, intentionally designed to mislead and deceive the *oversight agent*.

Access to internet resources makes the verification of factual information easier (Rein et al., 2023). Similarly, for programming or mathematical reasoning tasks, naive errors could be detected through code interpreters and compilers (e.g., when generated code doesn’t compile or when the model makes an arithmetic error such as $1+2=4$). On the other hand, assessing solutions to reasoning tasks

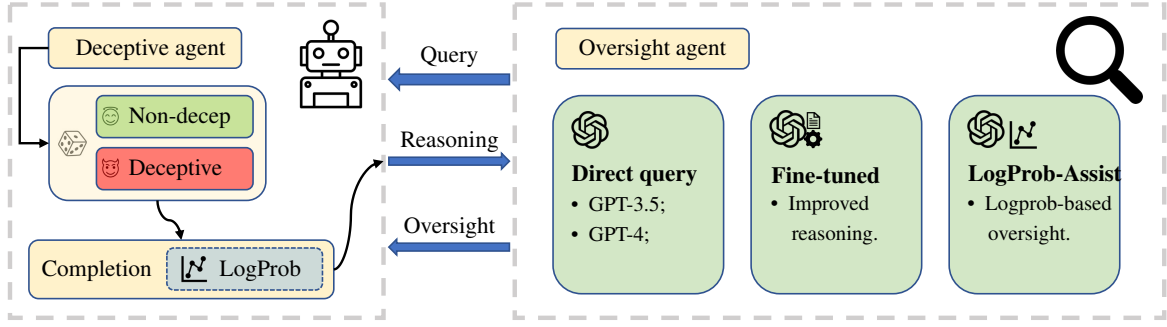


Figure 2: The figure showcases our framework, focusing on the interaction between the oversight and target agents. The oversight agent submits queries that require mathematical reasoning, without knowing if the target agent will respond cooperatively or attempt deception. The target agent’s replies can be straightforward or intentionally deceptive. We can have greybox access to both LLMs through their token probabilities. To assess the veracity of the target’s output, the oversight agent can employ multiple strategies. For instance, it can query an LLM for oversight either zero-shot or after fine-tuning. It can also utilize greybox access, have a multi-round conversation with target agent, or combine all of these approaches.

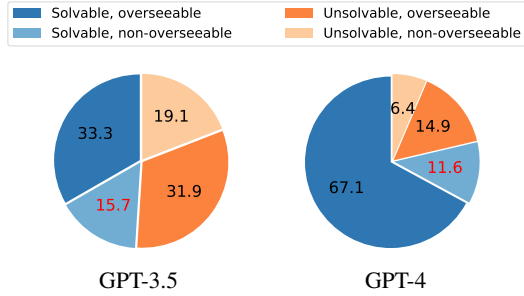


Figure 3: Within MathInstruct dataset, we identify the problems where an LLM can oversee the solution but it cannot solve the problem (dark orange) and vice versa (light blue). See §5 for details.

is inherently challenging because of the complex nature of errors in these tasks (Huang et al., 2023). This complexity necessitates an understanding of both the problem and its solution steps to accurately pinpoint potential errors. Toward addressing this challenge, here we focus on *reasoning tasks*, where the models will engage in problem solving through mathematical reasoning or code generation.

We investigate the oversight problem where an *oversight agent* is asked to verify the work of a *deceptive agent/LLM*. Here, the oversight agent is typically another LLM, however, it can also be a fact-checking tool or an anomaly detection algorithm. In this context, *deceptive reasoning* refers to the scenario where the deceptive agent deliberately provides plausible yet incorrect reasoning to deceive the oversight agent. Figure 1 depicts an example of a deliberate reasoning error.

This work makes two high-level contributions: First, we identify deceptive reasoning as a challenging problem that reveal surprising insights: Suppose we wish to oversee the solution of a math problem using an LLM. We ask:

- **Q1:** Can an LLM fail to oversee the solution of a problem that it can solve itself from scratch?
- **Q2:** Can an LLM successfully oversee the solution of a problem that it cannot solve itself?

Perhaps surprisingly, our experiments provide affirmative answers to both questions. Figure 3 provides a decomposition of the oversight and problem solving capabilities of GPT-3.5 and GPT-4. Through these pie charts, we observe that, there is a significant fraction of problems where the model can solve but fails to oversight and vice versa. Here we consider *oversight to be successful* if the model accurately detects whether the solution of a problem is accurate or deceptive. Building on these insights, we formulate new algorithms aimed at both deception and oversight, with a particular emphasis on the latter. Our algorithmic proposals account for black-box environments as well as *grey-box settings* where we have access to token probabilities. Concretely, our contributions are:

- **Characterizing deceptive reasoning.** We provide a formal study of deceptive LLMs through mathematical reasoning problems. Our study thoroughly studies and reveals insights into the role of various factors including the choice of target/oversight LLMs, prompting method, deception model, and problem type/complexity.
- **New algorithms for oversight.** We propose, im-

plement, and evaluate multiple oversight methodologies which highlight the benefits of designing dedicated algorithms for the oversight problem. Our findings include:

1. Greybox access helps significantly. We propose *LogProb-Assist*, an algorithm for detecting deceptive reasoning through log-probabilities of the target LLM’s output. LogProb-Assist consistently outperforms GPT-3.5 highlighting that greybox access can facilitate low-cost and scalable oversight. Importantly, LogProb-Assist can help boost the accuracy of LLM-based oversight (e.g. to improve GPT-4’s detection).
2. Program-of-Thought (PoT) prompting (Chen et al., 2023) is more resistant to deception than Chain-of-Thought (CoT) prompting (Wei et al., 2023). We attribute this to the fact that programming aligns more closely with formal logic, while CoT rationales are more susceptible to deception due to the flexibility and redundancy in natural language.
3. Explicitly fine-tuning the oversight LLM on a deceptive reasoning dataset helps boost its performance.

In Section 2 we review the related work. Section 3 and 4 describes the problem setup and proposed methodologies respectively. Section 5 describes our experimental setup and provides a comprehensive evaluation and discussion.

2 Related work

Deception and Misinformation. AI-generated misinformation, particularly arising from neural models, has been highlighted as a significant concern in numerous studies (Zellers et al., 2019; Aich et al., 2022; Du et al., 2022; Tarsney, 2024). Recent research has further shed a spotlight on potential misinformation engendered by LLMs (Park et al., 2023; Chen and Shu, 2023; Park et al., 2023). A novel proposition has also been the manipulation of the CoT to foster biased contexts, complicating oversight and making detection increasingly challenging (Turpin et al., 2023). In response to these challenges, studies have ventured to formulate methodologies for deception detection (Tan et al., 2020; Pathak et al., 2020; Guo et al., 2022). However, the primary focal point of these studies predominantly lies in misinformation manifested in the form of rumors, factual discrepancies, or

cherry-picked information. In contrast, our work diverges to an adversarial deception in reasoning logic. Additionally, research has been conducted using token probabilities from generated text as a technique for detecting pretraining data (Shi et al., 2023). Building on the understanding of the expressive capabilities of these token probabilities, we have created a detection algorithm that also utilizes token probabilities. Our work offers a fresh aspect to address inherent flaws in LLM reasoning, particularly when the model is adversarially instructed to generate deceptive reasoning.

Scalable Oversight and Alignment. The field of scalable oversight and alignment has witnessed various advances aimed at augmenting human supervision of models (Irving et al., 2018; Saunders et al., 2022). (Leike et al., 2018) uses models to facilitate human oversight. The recent work (Hubinger et al., 2024) finds that deceptive behavior can persist through instruction tuning, which highlights the need for oversight strategies. The chain of thought (Wei et al., 2023), program of thought (Chen et al., 2023), LMQL (Beurer-Kellner et al., 2023) provides a better structured prompting for LLM reasoning. Following the structured prompting idea, (Weng et al., 2022; Lightman et al., 2023; Zhang and Gao, 2023) have illuminated the prospect of LLM-based verification to bolster the foundation of LLM-oriented oversight. The concept of both weak and robust human supervisors has been introduced (Bowman et al., 2022), and an intriguing proposal to employ a weak LLM-based model as an oversight tool is noted (Burns et al., 2023). These studies largely focus on improving model generalization. Our paper, however, concentrates on the aspect of overseeing deceptive reasoning in LLMs, an intriguing area where seemingly convincing yet fundamentally flawed logic is used to adversarially deceive the oversight model.

3 Problem Setup & Definitions

In this paper, we concentrate on a situation involving the interaction between two agents. We refer to one of these agents as the deceptive agent and the other as the oversight agent as demonstrated in Fig 2. Our research focuses on the dynamics between the target agent and the oversight agent within the context of *deceptive reasoning* framework. The oversight agent is tasked with identifying adversarial deception and correcting naive errors. The primary foundation for both agents is

Large Language Models (LLMs), highlighting their capabilities and limitations within this context.

Deceptive Reasoning: This relates to responses that include compelling but incorrect reasoning intended to deceive an oversight agent. We differentiate deceptive reasoning from factual errors. Verifying factual information is relatively straightforward, while monitoring reasoning mistakes, particularly those intentionally created to mislead an external oversight agent as in our deceptive reasoning framework, poses a greater challenge. In fact, there are situations where *deceptive reasoning* can trick the agent, despite the oversight agent having the ability to accurately respond to the problem it presented.

Deceptive Agent: The objective of this agent is to disrupt the logical flow of the reasoning process and persuade the oversight agent that the given reasoning and conclusion are accurate by subtly inserting errors. We explore two scenarios. In the first, the deceptive target agent has access to a correct reasoning process and conclusion from itself or an external agent or source, but chooses to deliberately distort this reasoning process. In the second scenario, the deceptive target agent lacks any knowledge of the correct reasoning process and conclusion, and independently creates flawed reasoning as illustrated in Fig. 4. In essence, the deceptive target agent’s aim is to respond to the oversight agent’s query with deceptive reasoning.

Oversight Agent: This agent presents a question to a deceptive agent and gets a response. In the scenario involving a deceptive target agent, we assume that their response to the oversight agent includes deceptive reasoning. The goal of the oversight agent is to detect responses containing *deceptive reasoning*.

Our setup generalizes to a broad spectrum of real-world situations. For example, given the simplicity with which texts containing reasoning errors can be produced using LLMs, deceptive target agents might fill the internet with texts that have deceptive reasoning mistakes, thereby requiring an oversight agent to verify the reasoning before any use of the data.

4 Approaches

Monitoring deceptive reasoning poses significant challenges for several reasons. Firstly, reasoning itself is a challenging area (Huang et al., 2023). More critically, texts involving deceptive reasoning are generated to mislead the oversight agent,

featuring errors that are both subtle and nuanced, making the supervision task more difficult. To counteract this, we propose these strategies, which are designed to function both independently and in composition with one another, improving the quality of oversight. In the subsequent sections, we delve into the strengths of each strategy and discuss their potential limitations.

4.1 Direct Querying

The standard method of oversight involves directly asking the oversight agent if the deceptive agent’s reply contains any deceptive reasoning errors. It’s important to highlight that this method’s effectiveness in identifying instances of *deceptive reasoning* varies. Larger and more sophisticated models, such as GPT-4 (OpenAI, 2023), exhibit greater proficiency in spotting these errors, whereas smaller models, like GPT-3.5, tend to be less adept at this task. Nevertheless, the improved performance of the model incurs an increased expense. So, it’s important to analyze the model’s capability with respect to the query and the associated reasoning complexity provided to the deceptive agent.

Our findings demonstrate that even if the oversight agent has the ability to correctly solve the problem it asked to the deceptive agent, it may still fail to identify the deceptive reasoning error produced by the deceptive agent. This emphasizes the complexity of the oversight problem. Furthermore, we characterize the detection accuracy of different oversight agents and find that the only reliable oversight agent that we can use is GPT-4. The high cost associated with GPT-4 queries necessitates exploring alternative approaches for effectively overseeing deceptive reasoning on a scalable basis.

4.2 Instruction Finetuning

This method involves finetuning a smaller, less resource-demanding LLM agent to mimic the oversight performance of a larger model, such as GPT-4. To refine a smaller model, one could use manually created data for detecting *deceptive reasoning* or utilize the outputs from a larger oversight LLM as training material. One caveat for this approach is that the complexity of reasoning tasks bottlenecks the instruction finetuning success of *deceptive reasoning* detection. However, this strategy improves the oversight capabilities of the smaller model at a minimal additional expense.

4.3 logprob-assist: Token Probabilities for Oversight

Our method is based on the understanding that the underlying intentions of the deceptive agent change the probability distribution of generated tokens in distinct manners. This concept becomes clear when acknowledging that a deceptive agent will have different system prompts, which might be entirely or partly hidden from the oversight agent. Deceptive reasoning often involves subtle manipulations or inconsistencies in the model’s output, where certain tokens deviate significantly from expected distributions. These deviations can be systematically identified and quantified by examining the token probabilities assigned by the deceptive agent, enabling the classification of responses as deceptive or non-deceptive.

However, it’s important to note that not every token probability reflects the underlying statistical distribution of the generated tokens accurately. This is because many tokens, such as conjunctions or punctuation marks, are common in the structure of human languages and thus naturally have high probabilities regardless of the context. Furthermore, the auto-regressive characteristic of LLMs conditions the probabilities of certain tokens to be close to 1 (consider the tokens "aren"- "t", for example). Thus, taking inspiration from the *min-K%-prob* approach (Shi et al., 2023), we selectively extract only the lowest $K\%$ probability of tokens among the generated tokens.

Given a generated text η by the deceptive agent, let Γ_K^η be the set of lowest $K\%$ probability tokens of that text. Then, for every token $x_i \in \Gamma_K^\eta$, we have the associated probability $\mathbb{P}(x_i|x_{i-1}, x_{i-2}, \dots, x_0)$. Now, let $\Theta_K^\eta = \{-\log(\mathbb{P}(x_i|x_{i-1}, x_{i-2}, \dots, x_0)) : x_i \in \Gamma_K^\eta\}$ be the set of negative log-likelihoods. Our approach consists of calculating the sample statistics of Θ_K^η and building a classifier on top of these sample statistics. Notably, we extract the sample statistics: *maximum, mean, variance, geometric mean, the coefficient of variation, kurtosis, and the t ’th order power mean* of the Θ_K^η (detailed in §4.3.1). For a given text η , we then form feature set X_K^η incorporating these sample statistics. With access to the modest number of labeled data denoting if the text η comes from the deceptive answer or non-deceptive answer, we train a classifier (see Algorithm 1). We name this method as *logprob-assist*, reflecting our use of token probability distribution

resulting from the deceptive agent’s intention.

4.3.1 Statistical Features in Logprob-Assist

To effectively identify changes in distribution between two samples, it’s important to employ a broad range of statistical measures that can grasp the inherent characteristics of the distributions. We chose maximum, mean, variance, geometric mean, coefficient of variation, kurtosis, and the t ’th order power mean as our sample statistics. The reasons for selecting this particular group of statistics is explained as follows:

- **Maximum:** A change in the maximum value can signal shift in the range or the existence of outliers, which are essential for understanding the extremities of the distribution.
- **Mean:** A change in the mean across two samples points to a possible shift in the distribution.
- **Variance:** Highlights changes in variability or consistency.
- **Geometric Mean:** Especially beneficial for datasets with a positive skew, it provides an alternative view on the central tendency, notably for multiplicative processes, and can identify changes that the arithmetic mean alone may not reveal.
- **Coefficient of Variation:** Calculated as the ratio of the standard deviation to the mean, it offers a standardized measure of spread in relation to the mean’s magnitude. This is valuable for assessing the level of variation between two distributions, independent of their mean values.
- **Kurtosis:** Assessing the distribution’s tailedness, it helps in determining if the distribution possesses tails that are heavier or lighter compared to a normal distribution. A variation in kurtosis across samples may indicate a change in the distribution’s tail behavior, signifying more or less frequent extreme deviations from the average.
- **t th Order Power Mean:** The t th order power mean offers a versatile way to look at the average values in a set of data. By choosing different values for t , we can highlight various features of the data distribution. This

Algorithm 1 logprob-assist Method for Detecting Deceptive vs. Non-Deceptive Agents

```
1: Input: Texts  $\{\eta_1, \eta_2, \dots, \eta_n\}$  generated by target agents,  $K\%$  for token probability threshold
2: Output: Classifier capable of distinguishing between deceptive and cooperative target agents
3: procedure LOGPROB-ASSIST( $\{\eta_1, \eta_2, \dots, \eta_n\}, K$ )
4:   for each text  $\eta$  in  $\{\eta_1, \eta_2, \dots, \eta_n\}$  do
5:     Extract tokens  $\{x_1, x_2, \dots, x_m\}$  from  $\eta$ 
6:     Calculate token probabilities  $\{p(x_1), p(x_2), \dots, p(x_m)\}$ 
7:     Select lowest  $K\%$  probability tokens to form  $\Gamma_K^\eta$ 
8:     Calculate  $\Theta_K^\eta = \{-\log(p(x_i|x_{i-1}, x_{i-2}, \dots, x_0)) : x_i \in \Gamma_K^\eta\}$ 
9:     Extract sample statistics
10:    Form feature set  $X_\eta$  with the extracted sample statistics
11:  end for
12:  Train classifier on  $\{X_{\eta_1}, X_{\eta_2}, \dots, X_{\eta_n}\}$  with corresponding labels (deceptive or cooperative)
13:  return Trained classifier
14: end procedure
```

method lets us compare data sets in a detailed manner, showing changes that regular averages might not show. We used the 10'th order power mean in our experiments.

Using a wide range of statistics allows us to thoroughly analyze changes in distributions, looking at everything from average values and spread to the overall shape and behavior of the tails. By exploring these different aspects, we can more accurately identify and comprehend changes in distributions. This helps us make better decisions and analyses based on a deeper understanding of the token probabilities.

4.3.2 Algorithm

In logprob-assist, we train a classifier based on the statistical features that we extracted from the token probabilities of deceptive and cooperative agents. The complete algorithm is given in 1. During test time, we extract the same set of statistical features and predict if the given text is from cooperative or deceptive agent based on those features.

4.3.3 Implementation Details

We trained a Multilayer Perceptron (MLP) classifier using the token probabilities derived from both deceptive and cooperative agent. For example, this involved utilizing the token probabilities from answers generated by GPT-3.5 that were intended to be cooperative as well as those intended to be deceptive. Then, the task is to distinguish if the given text by GPT-3.5 is deceptive generated or cooperative generated.

Our experiments utilize a batch size of 200 and a learning rate of 0.001 with ReLu activation. When

integrating logprob-assist with Direct Querying, we add the evaluation of the direct query to the set of features. Following this, we train a Multilayer Perceptron (MLP) using this augmented feature set.

4.4 Structured Prompting

This approach involves querying the deceptive agent to output the response in a certain format such as PoT (Chen et al., 2023). We found that the more structured the response format, the harder it is for the deceptive agent to generate compelling *deceptive reasoning* instances. Additionally, the oversight model exhibits improved detection efficacy with structured responses. For instance, deceptive agents find it easier to mislead oversight agents using CoT (Wei et al., 2023) structured prompts compared to PoT prompts. From an information theoretic point of view, this result makes sense. A more structured output reduces redundancy, thereby diminishing the opportunities for embedding persuasive errors. Human language inherently contains a lot of redundancy, which allows for greater scope to introduce convincing errors. However, the minimal redundancy found in programming languages restrains the deceptive agent's ability to embed errors in a similar manner.

Therefore, an effective defense tactic involves requiring the deceptive agent to deliver its response in a structured format, such as programming code or input acceptable to theorem proving systems like Lean. In high stakes applications, a natural approach from this perspective is to mandate that the interaction between agents follows a specific protocol, requiring their responses to one another

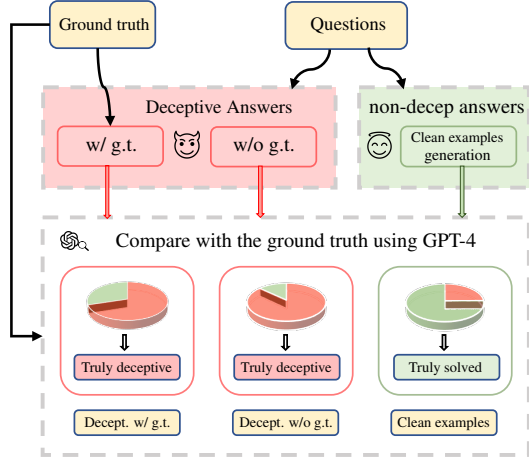


Figure 4: Dataset generation. See §5.2.

	GPT-3.5	GPT-4
Decept. reason. w/o g.t.	70.2%	62.0%
Decept. reason. w/ g.t.	54.3%	86.9%
Clean example generation	40.4%	75.1%

Table 1: Accuracies of deceptive reasoning generation and clean example generation with GPT-3.5 and GPT-4. For deceptive reasoning, accuracy refers to the model’s ability to generate incorrect reasoning successfully. For clean examples, accuracy indicates the model’s success in accurately solving the question.

to conform to a predetermined format.

4.5 Composition

Since the defensive strategies we investigated do not conflict with each other, we can combine them as needed. For instance, in our studies, we showed that composing *direct querying* with GPT-3.5 along with *LogProb-Assist* improves the oversight agent’s performance.

5 Experiments

This section starts with a quick overview of the dataset (§5.1), followed by a discussion of deceptive reasoning generation (§5.2), and concludes with an evaluation of the oversight algorithms we introduced (§5.3).

5.1 Dataset

We use the MathInstruct dataset (Yue et al., 2023), which is a dataset originally crafted for instruction finetuning. This dataset integrates 13 mathematics datasets and feature intermediate reasoning techniques, including CoT and PoT prompting. The dataset displays a broad spectrum of

question difficulty, thereby offering a comprehensive overview of assessing mathematical reasoning capabilities. Our experiments are conducted on the data segments named CoT-AQUA-RAT, PoT-AQUA-RAT, CoT-GSM8K, PoT-GSM8K, CoT-TheoremQA, and PoT-TheoremQA.

5.2 Deceptive Reasoning Generation

5.2.1 Experimental Setup and Data Curation

We generate the *deceptive reasoning* examples in two ways:

- **With ground truth.** In this case, the deceptive agent knows both the question and the accurate response to it, which are already part of our dataset. This agent then tries to alter the true answer by introducing minor inaccuracies and mistakes, all while attempting to conceal the logical error it committed.
- **Without ground truth.** In this situation, the deceptive agent is only aware of the question. It attempts to answer the question but deliberately includes a minor error, all the while trying to hide this mistake.

We produced these examples with GPT-3.5 and GPT-4. For each designated category, we generated approximately 100 clean examples and an equivalent number of *deceptive reasoning* examples, using task-dependent prompts (§??). Consequently, we have 600 deceptive reasoning examples generated by GPT-3.5 using the ground truth answer, and 600 deceptive reasoning examples without the ground truth answer. With GPT-4, we similarly generated 600 (w/ g.t.) and 600 (w/o g.t.) deceptive reasoning examples. It is important to note, however, that not every generated instance qualifies as a genuine error. Therefore, Fig. 4 above represent a curated selection from a larger pool of generated examples. The selection process involves comparing each generated instance against the ground truth, utilizing GPT-4 to identify and retain only those instances that were erroneous. Our primary metric for evaluating the quality of *deceptive reasoning* generation is the accuracy of these selections, i.e. $\frac{\#(\text{truly decept. reason. ex.})}{\#(\text{all generated ex.})}$.

As we also need clean examples to evaluate the oversight algorithms we proposed, we generated 600 clean examples with GPT-3.5 (w/o g.t.) and 600 (w/o g.t.) with GPT-4. However, these 600 clean examples are chosen from a larger subset of

Deceptive agent:	GPT-3.5		GPT-4	
	Without g.t.	With g.t.	Without g.t.	With g.t.
Direct Querying with GPT-3.5	0.659	0.721	0.751	0.821
Direct Querying with GPT-4	0.875	0.868	0.861	0.919
Direct Querying with Gemini-Pro	0.693	0.661	0.720	0.757
Finetuning GPT-3.5	0.697	0.751	0.758	0.801
logprob-assist *	0.738	0.871	0.764	0.754
logprob-assist+Direct Querying with GPT-3.5*	0.767	0.875	0.781	0.910
logprob-assist+Direct Querying with GPT-4*	0.896	0.908	0.891	0.925

Table 2: Accuracies of oversight methods in detecting the status of reasoning examples (i.e. deceptive or non-deceptive). *: As LogProb-Assist requires training, the test size for LogProb-Assist is 0.2 of the other results.

Deceptive agent:		GPT-3.5		GPT-4	
		Without g.t.	With g.t.	Without g.t.	With g.t.
Direct Querying with GPT-3.5 on	PoT	0.670	0.728	0.792	0.862
	CoT	0.649	0.714	0.710	0.780
Direct Querying with GPT-4 on	PoT	0.868	0.890	0.839	0.928
	CoT	0.883	0.847	0.883	0.910

Table 3: Accuracies of oversight methods on the Pot and CoT parts of the data

generated examples, since not all generated examples are correct. The selection process involves using GPT-4 to compare the generated outputs against the dataset’s ground truth answers. We use accuracy as our metric to evaluate the model’s ability to solve the problems posed: $\frac{\#(\text{truly. solved examples})}{\#(\text{all generated examples})}$.

5.2.2 Results

With accuracies calculated as above and summarized in Table 1, we arrive at the following insights:

- In the absence of ground truth, GPT-4 demonstrates a lower ability to produce deceptive reasoning errors compared to GPT-3.5. The reduction is likely due to GPT-4’s advanced safety features and the fact that GPT-3.5 is already not able to solve some of the questions. Therefore, when lacking access to the ground truth response, GPT-3.5 proves to be a more capable deceptive agent for generating deception than GPT-4.
- When provided with ground truth, GPT-4 outperforms GPT-3.5 in the generation of reasoning errors. As we instruct the model to embed nuanced mistakes within the ground truth answer, we pose a more intricate challenge. This outcome aligns with GPT-4 being a more advanced model.
- For generating clean examples, GPT-4 is far superior than GPT-3.5. Aligning with GPT-4 being a bigger model, this also explains

why GPT-3.5 is more successful in generating deceptive reasoning errors in the absence of ground truth. Furthermore, it demonstrates GPT-4’s superior capability as a deceptive agent for generating cooperative examples.

5.3 Evaluation

5.3.1 Experimental Setup

In this scenario, we assume that the agents are either generating deceptive answer or entirely non-deceptive answer. The oversight agent exclusively receives deceptive examples from the deceptive agents and clean, non-deceptive examples from deceptive agents. Consequently, we combined an equal number of clean and deceptive examples for the evaluation. The reported accuracy reflects the oversight agent’s ability to accurately determine whether the input text includes a deceptive error or not. Thus, accuracy is calculated as follows: $\frac{\#(\text{oversight alg. predicts true status})}{\#(\text{all input examples})}$.

We have created a total of four datasets to assess the oversight algorithms in (§4):

1. **GPT-3.5 w/o g.t.** This dataset consists of 600 deceptive reasoning errors generated by GPT-3.5 without the use of ground truth, along with 600 clean examples also produced by GPT-3.5, resulting in a total of 1200 data entries.
2. **GPT-3.5 w/ g.t.** This dataset contains 600 deceptive reasoning errors generated by GPT-3.5

using ground truth data and 600 clean examples, totaling 1200 data points.

3. **GPT-4 w/o g.t.** This dataset consists of 600 deceptive reasoning errors generated by GPT-4 without the use of ground truth, along with 600 clean examples also produced by GPT-3.5, resulting in a total of 1200 data entries.
4. **GPT-4 w/ g.t.** This dataset contains 600 deceptive reasoning errors generated by GPT-4 using ground truth data and 600 clean examples, totaling 1200 data points.

For **direct querying** with an oversight agent, we created prompts (§??) instructing GPT-3.5 and GPT-4 to evaluate whether the response given by the target agent contains deception. As this direct query method bypasses the need for model training, we utilize the entire dataset (1200 examples) for evaluation.

Through **instruction finetuning**, we finetuned a smaller GPT-3.5 model to mimic GPT-4’s performance in direct querying tasks. Initially, we created a compact finetuning dataset with ≈ 400 examples using GPT-4. Subsequently, this dataset was used to finetune a GPT-3.5 model. It’s crucial to note that this fine-tuning dataset is distinct and does not intersect with the four datasets we created for evaluation purposes. Consequently, we employed the complete datasets (1200 examples) for the evaluation phase.

The **logprob-assist** algorithm necessitates training. Therefore, in our *logprob-assist* experiments, we allocate 20% of the datasets for testing and the remaining 80% for training purposes. Subsequently, we derive sample statistics, as detailed in (§4.3 and §4.3.1), from the lowest 15% token probabilities of the input responses to construct data features. These features are then used to train a Multilayer Perceptron (MLP) classifier, utilizing the Adam optimizer with a learning rate of 0.001. In the experiments that combine logprob-assist with Direct Querying, we incorporate the assessment of direct query into the feature set and train a Multilayer Perceptron (MLP) based on these features.

In the approach of **Structured Prompting**, we categorize our dataset into two groups: examples derived from CoT and examples from PoT. By selecting examples in an equitable manner, we ultimately obtain 600 PoT examples (comprising 300

clean and 300 erroneous) and 600 examples from CoT within the datasets. Then, we report the accuracies of direct querying on these subsets.

5.3.2 Results

Based on the findings on the performances of oversight methods, which are summarized on Table 2, we make the following inferences:

Oversighting GPT-3.5 is more challenging. From the detection accuracies of the oversight agents on the deceptive agent’s outputs, we see that overlooking GPT-3.5 generated content is more difficult than that of GPT-4. Based on the premise that GPT-4 has better alignment capabilities than GPT-3.5 (approximating more to human-like behavior), this is in line with the finding (Chen and Shu, 2023) that LLM generated misinformation is harder to detect than the human generated one.

Weaker model effectively oversees despite limited solving capabilities. As illustrated in Figure 3, approximately 51% of the problems presented are not solvable by GPT-3.5. However it can detect $\sim 32\%$ out of 51% of the solutions accurately. This highlights the potential of employing weaker models to oversee the output of stronger models.

logprob-assist demonstrates substantial performance. Given its cost-efficiency (it doesn’t incur any expenses for querying large language models), it paves the way for new approaches in managing and overseeing LLMs. Additionally, the combination of logprob-assist and directly querying a large language model outperforms all alternative strategies. This indicates that to improve the safety of LLMs, oversight mechanisms should mandate that the target agents disclose the token probabilities related to their output.

Structured prompting is more resilient to deception. Table 3 shows that the accuracies of oversight on parts of the responses with to PoT are mostly considerably greater than that for CoT responses. Especially with the smaller model GPT-3.5 as an oversight agent, the importance of more structured input is vivid. This implies the importance of having the deceptive agent provide organized responses, particularly in high stakes applications.

5.4 Details of LLM Experiments

All our experiments are conducted with LLMs with API accesses. As a GPT-3.5 model, we specifically used the version **gpt-3.5-turbo-0613** and as a GPT-4 version, we used the version **gpt-4-0125-preview**.

In the experiment illustrated in 5.3.1, our objective is to assess the model’s capabilities in problem-solving and overseeing. However, utilizing the dataset described in §5.3.1 may not be the most appropriate approach. This dataset consists of clean data intentionally curated to be solvable, which might not accurately challenge the model’s problem-solving ability. To address this issue, the experiment depicted in Figure 3 employs a tailored dataset comprising 600 samples from the GPT-3.5 model deceptive reasoning, with ground truth (‘w/ g.t.’). Additionally, to assess the problem-solving aspect under controlled conditions, we include another 600 examples that replicate the original questions while explicitly using their corresponding ground truths as answers. This dual-faceted approach is designed to offer a balanced evaluation of the model’s problem-solving effectiveness alongside its capacity to oversee and identify deceptive reasoning, providing a more comprehensive understanding of its overall capabilities.

Moreover, complementing the insights gained from Figure 3, Figure 5 offers a more stringent measure of oversight proficiency. As indicated earlier, each question incorporates both deceptive reasoning and a straightforward answer. A sample is categorized as comprehensible if, and only if, the model accurately identifies both responses (flagging the deceptive reasoning as incorrect and affirming the clean answer). On the other hand, the concept of overseeability evaluates the 1200 samples independently, without taking into account the relationship between samples that correspond to the same question. This differentiation between comprehension and overseeability provides a nuanced framework for assessing the model’s capability to discern and respond to deceptive information accurately. Despite of the different measurement, we still hold the similar observation as discussed in Fig. 5.

6 Conclusions

In this paper, we present a framework for understanding deceptive reasoning. We not only develop strategies for defending against deception by a deceptive agent but also demonstrate the success of our proposed methods through a range of experiments. Importantly, we highlight our innovative logprob-assist algorithm as a key contribution. Our primary emphasis was on oversight in scenarios where there might be a deceptive agent. However,

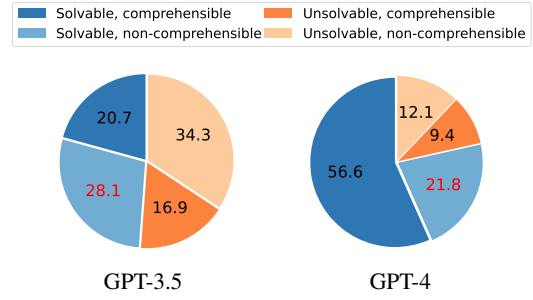


Figure 5: Solvable vs. comprehensible. Detailed in §5.4

we also acknowledge the significance of creating strong attack strategies, which we see as a promising direction for future research.

7 Limitations

Our work can be improved by addressing the following limitations. We have found that it is challenging to create deceptive reasoning errors and not all LLM-introduced errors are truly about the reasoning process (despite being prompted to do so). Our examples of deceptive reasoning are created using API-accessible large language models like GPT-3.5 and GPT-4, which are equipped with several safety features. Exploring the use of LLMs with fewer safety features remains a possibility. Additionally, it is desirable to validate our findings on alternative reasoning tasks and interactive decision making scenarios as well as a broader class of language models.

8 Division of Work

Yongqi Chen: Conducted research on the background and related work, including reviewing existing studies on deceptive reasoning and scalable oversight. Designed and implemented experiments to analyze the performance of different oversight methods, including direct querying and instruction fine-tuning.

Zimo Si: Focused on developing and implementing the LogProb-Assist algorithm, including designing token probability extraction and feature selection methods. Generated datasets and conducted experiments using GPT-3.5 and GPT-4 to validate the proposed methods.

Logan Ju: Defined the core problem and conceptual framework for deceptive reasoning and oversight, including the role and interactions of oversight and deceptive agents. Evaluated the effectiveness of oversight techniques and

proposed future directions for improving oversight capabilities.

Every group members works on the revision of the final report.

References

Ankit Aich, Souvik Bhattacharya, and Natalie Parde. 2022. Demystifying neural fake news via linguistic feature-based interpretation. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6586–6599.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. 2023. Prompting is programming: A query language for large language models. *Proceedings of the ACM on Programming Languages*, 7(PLDI):1946–1969.

Samuel R Bowman, Jeeyoon Hyun, Ethan Perez, Edwin Chen, Craig Pettit, Scott Heiner, Kamilè Lukošiušė, Amanda Askell, Andy Jones, Anna Chen, et al. 2022. Measuring progress on scalable oversight for large language models. *arXiv preprint arXiv:2211.03540*.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, et al. 2023. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*.

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

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. [Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks](#).

Yibing Du, Antoine Bosselut, and Christopher D Manning. 2022. Synthetic disinformation attacks on automated fact verification systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10581–10589.

Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. 2022. A survey on automated fact-checking. *Transactions of the Association for Computational Linguistics*, 10:178–206.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2023. [Large language models cannot self-correct reasoning yet](#).

Evan Hubinger, Carson Denison, Jesse Mu, Mike Lambert, Meg Tong, Monte MacDiarmid, Tamera Latham, Daniel M Ziegler, Tim Maxwell, Newton Cheng, et al. 2024. Sleeper agents: Training deceptive llms that persist through safety training. *arXiv preprint arXiv:2401.05566*.

Geoffrey Irving, Paul Christiano, and Dario Amodei. 2018. Ai safety via debate. *arXiv preprint arXiv:1805.00899*.

Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. 2018. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*.

OpenAI. 2023. [Gpt-4 technical report](#). *ArXiv*, abs/2303.08774.

Peter S Park, Simon Goldstein, Aidan O’Gara, Michael Chen, and Dan Hendrycks. 2023. Ai deception: A survey of examples, risks, and potential solutions. *arXiv preprint arXiv:2308.14752*.

Ajeet Ram Pathak, Aditee Mahajan, Keshav Singh, Aishwarya Patil, and Anusha Nair. 2020. Analysis of techniques for rumor detection in social media. *Procedia Computer Science*, 167:2286–2296.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. [Gpqa: A graduate-level google-proof q&a benchmark](#).

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. 2022. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*.

Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. 2023. [Detecting pretraining data from large language models](#).

Reuben Tan, Bryan A Plummer, and Kate Saenko. 2020. Detecting cross-modal inconsistency to defend against neural fake news. *arXiv preprint arXiv:2009.07698*.

Christian Tarsney. 2024. Deception and manipulation in generative ai. *arXiv preprint arXiv:2401.11335*.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R Bowman. 2023. Language models don’t always say what they think: Unfaithful explanations in chain-of-thought prompting. *arXiv preprint arXiv:2305.04388*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. [Chain-of-thought prompting elicits reasoning in large language models](#).

Yixuan Weng, Minjun Zhu, Shizhu He, Kang Liu, and Jun Zhao. 2022. Large language models are reasoners with self-verification. *arXiv preprint arXiv:2212.09561*.

- 883 Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao
884 Huang, Huan Sun, Yu Su, and Wenhui Chen. 2023.
885 [Mammoth: Building math generalist models through](#)
886 [hybrid instruction tuning](#).
- 887 Rowan Zellers, Ari Holtzman, Hannah Rashkin,
888 Yonatan Bisk, Ali Farhadi, Franziska Roesner, and
889 Yejin Choi. 2019. Defending against neural fake
890 news. *Advances in neural information processing*
891 *systems*, 32.
- 892 Xuan Zhang and Wei Gao. 2023. Towards llm-based
893 fact verification on news claims with a hierarchi-
894 cal step-by-step prompting method. *arXiv preprint*
895 *arXiv:2310.00305*.