How Is ChatGPT’s Behavior Changing over Time?

Lingjiao Chen†*, Matei Zaharia‡, James Zou†
† Stanford University
‡ University of California, Berkeley

Abstract. GPT-3.5 and GPT-4 are the two most widely used large language model (LLM) services. However, when and how these models are updated over time is opaque. Here, we evaluate the March 2023 and June 2023 versions of GPT-3.5 and GPT-4 on several diverse tasks: 1) math problems, 2) sensitive/dangerous questions, 3) opinion surveys, 4) multi-hop knowledge-intensive questions, 5) generating code, 6) US Medical License tests, and 7) visual reasoning. We find that the performance and behavior of both GPT-3.5 and GPT-4 can vary greatly over time. For example, GPT-4 (March 2023) was reasonable at identifying prime vs. composite numbers (84% accuracy) but GPT-4 (June 2023) was poor on these same questions (51% accuracy). This is partly explained by a drop in GPT-4’s amenity to follow chain-of-thought prompting. Interestingly, GPT-3.5 was much better in June than in March in this task. GPT-4 became less willing to answer sensitive questions and opinion survey questions in June than in March. GPT-4 performed better at multi-hop questions in June than in March, while GPT-3.5’s performance dropped on this task. Both GPT-4 and GPT-3.5 had more formatting mistakes in code generation in June than in March. We provide evidence that GPT-4’s ability to follow user instructions has decreased over time, which is one common factor behind the many behavior drifts. Overall, our findings show that the behavior of the “same” LLM service can change substantially in a relatively short amount of time, highlighting the need for continuous monitoring of LLMs.

Keywords: GPT-4, ChatGPT, LLM drifts, temporal evaluation

Media Summary

GPT-4 and GPT-3.5 are the two most widely used large language models. However, how and when they are updated is opaque. This paper shows significant drifts of their performance and behavior over a short time period on a range of different tasks. We also identify evidence of that GPT-4’s ability to follow user instructions has declined, which we hypothesize to partially explain the performance drifts. Overall, this paper highlights the importance to continuously monitor LLMs for trustworthy and reliable applications.

This article is © 2024 by author(s) as listed above. The article is licensed under a Creative Commons Attribution (CC BY 4.0) International license (https://creativecommons.org/licenses/by/4.0/legalcode), except where otherwise indicated with respect to particular material included in the article. The article should be attributed to the author(s) identified above.
1. Introduction

Large language models (LLMs) like GPT-3.5 and GPT-4 are being widely used. A LLM like GPT-4 can be updated over time based on data and feedback from users as well as design changes. However, it is currently opaque when and how GPT-3.5 and GPT-4 are updated, and it is unclear how each update affects the behavior of these LLMs. These unknowns make it challenging to stably integrate LLMs into larger workflows: if LLM’s response to a prompt (e.g. its accuracy or formatting) suddenly changes, this might break the downstream pipeline. It also makes it challenging, if not impossible, to reproduce results from the “same” LLM.

Beyond these integration challenges, it is also an interesting question whether an LLM service like GPT-4 is consistently improving over time. It is important to know whether updates to the model aimed at improving some aspects can reduce its capability in other dimensions.

Motivated by these questions, we evaluated the behavior of the March 2023 and June 2023 versions of GPT-3.5 and GPT-4 on several tasks: 1) solving math problems, 2) answering sensitive/dangerous questions, 3) answering opinion surveys, 4) answering multi-hop knowledge-intensive questions, 5) generating code, 6) US Medical License exams, and 7) visual reasoning. These tasks were selected to evaluate diverse and useful capabilities of these LLMs. We find that the performance and behavior of both GPT-3.5 and GPT-4 varied significantly across these two releases and that their performance on some tasks have gotten substantially worse over time, while they have improved on other problems (summarized in Figure 1).

How to explain those performance and behavior drifts? We hypothesize that changes in ChatGPT’s ability to follow user instructions could be a common factor behind the drifts across tasks. As a first step towards testing this hypothesis, we have curated a set of task-agnostic instructions, and evaluate the March and June versions of GPT-4 and GPT-3.5 on it. Overall, we observe a large decrease of GPT-4’s ability to follow many instructions. GPT-4 in March was typically good at following user’s instructions (e.g. generating responses following specified formats), but in June it failed to follow most of these simple instructions (Figure 1).

Our findings highlight the need to continuously monitor LLMs’ behavior over time. All prompts we curated in this paper and responses from GPT-4 and GPT-3.5 in both March and June are collected and released in https://github.com/lchen001/LLMDrift. Our analysis and visualization code has also been open-sourced. We hope our work stimulates more study on LLM drifts to enable trustworthy and reliable LLM applications.

1.1. Related Work. There have been multiple benchmarks and evaluations of LLMs including GPT-3.5 and GPT-4 (Bang et al., 2023; Liang et al., 2022; Liu et al., 2023; Zhang et al., 2023). Existing works show that LLMs achieve reasonable performance on traditional language tasks such as reading comprehension (de Winter, 2023), translation (Jiao et al., 2023), and summarization (Goyal et al., 2022). More recently, GPT-4 was shown to successfully pass difficult exams in professional domains such as medicine (Nori et al., 2023) and law (Katz et al., 2023). On the other hand, carefully curated language tasks trivial to humans are shown to be challenging for several LLMs (Efrat et al., 2023). To the best of our knowledge, most of these works do not systematically monitor the longitudinal drifts of widely used LLM services over time or report large drifts in them. ChatLog (Tu et al., 2023) proposed recording and monitoring ChatGPT’s responses automatically over time and reported small shifts (most below 5%) in ChatGPT’s performance on some common benchmarks. Other papers (Aiyappa et al., 2023; Shakarian et al., 2023) also reported shifts in specific problems. Monitoring model performance shifts is an emerging research area for machine-learning-as-a-service.
How Is ChatGPT’s Behavior Changing over Time?

Figure 1. Overview of performance drift (a) and instruction following shift (b) of GPT-4 (left panel) and GPT-3.5 (right panel) between March 2023 and June 2023. A higher evaluation metric is better. On eight diverse tasks (detailed below), the models’ performance drifts considerably over time, and sometimes for the worse. The decrease of GPT-4’s ability to follow instructions over time matched its behavior drift and partially explained the corresponding performance drops.

Researchers have evaluated temporal shifts of previous (non-LLM) AI models (L. Chen et al., 2022), and studied how to efficiently estimate ML service performance shifts (L. Chen et al., 2021). Those papers focus on ML services for simple classification tasks such as sentiment analysis, while this work studies generative LLM services.

2. OVERVIEW: LLM SERVICES, TASKS AND METRICS

This paper studies how different LLMs’ behaviors change over time. To answer it quantitatively, we need to specify (i) which LLM services to monitor, (ii) on which application scenarios to focus, and (iii) how to measure LLM drifts in each scenario.

2.1. LLM Services. The LLM services monitored in this paper are GPT-4 and GPT-3.5, which form the backbone of ChatGPT. Due to the popularity of ChatGPT, both GPT-4 and GPT-3.5 have been widely adopted by individual users and a number of businesses. Thus, timely and systematically monitoring these two services helps a large range of users better understand and leverage LLMs
How Is ChatGPT’s Behavior Changing over Time? 4

Figure 2. Performance of the March 2023 and June 2023 versions of GPT-4 and GPT-3.5 on eight tasks: (a,b) solving math problems (Prime vs Composite and Happy Numbers), (c) responding to sensitive questions and (d) opinion surveys, (e) running a LangChain app for multi-hop question answering, (f) generating executable code, (g) the USMLE medical exam, and (h) visual reasoning. For each task, one example is shown in a purple box, and the number of examples n is in the caption. All error bars in this figure denote 95% bootstrap confidence intervals. The models’ performance varies substantially over time, and sometimes for the worse.

for their own use cases. At the time of writing, there are two major versions available for GPT-4 and GPT-3.5 through OpenAI’s API, one snapshotted in March 2023 and another in June 2023. Differences between these two versions are of particular interest: the default GPT-4 API points to the June versions at the time of writing, but many applications were developed via the March versions. In addition, studying these snapshots makes all experiments in this paper reproducible. Therefore we focus on the drifts between these two dates in this paper. For simplicity, we queried these services via the user prompt only and left the system prompt as default. We set the temperature to be 0.1 to reduce output randomness, as creativity was not needed in our evaluation tasks.
2.2. Evaluation Tasks. In this paper, we focus on eight LLM tasks frequently studied in performance and safety benchmarks: solving math problems (including two problem types), answering sensitive questions, answering OpinionQA survey, LangChain HotpotQA Agent, code generation, taking USMLE medical exam, and visual reasoning, as shown in Figure 2. These tasks are selected for two reasons. These are diverse tasks frequently used to evaluate LLMs in the literature. For example, math problems are commonly used to study prompt techniques (Bubeck et al., 2023; Wei et al., 2022). Researchers (Bianchi et al., 2023; Zhang et al., 2023) have studied LLM safety by measuring LLMs’ performance on sensitive question datasets (Zhang et al., 2023) and whose opinions LLM reflect via the OpinionQA dataset (Santurkar et al., 2023). HotpotQA is often used to study how an LLM performs on multi-hop question answering (Yao et al., 2022). How LLMs perform on code generation is also of major interests (Bubeck et al., 2023; M. Chen et al., 2021). Recent works (Gilson et al., 2023; Kung et al., 2023; Mbakwe et al., 2023) also show that LLMs such as ChatGPT can pass the USMLE medical exam. Researchers (Xu et al., 2023) have also used the visual reasoning task to study how task representations affect LLMs’ performance. Furthermore, these tasks also measure a range of diverse abilities, including logic reasoning, knowledge extraction, code generation, domain-specific analysis, and visual imagination and deduction. They have the added benefit of relatively objective and thus easy-to-evaluate. For each task, we use queries either sampled from existing datasets or constructed by us. We acknowledge that the specific benchmark datasets used here does not comprehensively cover the complex behaviors of ChatGPT. Our goal here is not to provide a holistic assessment but to demonstrate that substantial ChatGPT performance drift exists on simple tasks. We are adding more benchmarks in future evaluations as part of a broader, long-term study of LLM service behavior. We cover each task in detail in the next section.

2.3. Metrics. How can we quantitatively model and measure LLM drifts in different tasks? Here, we consider one main performance metric for each task and two common additional metrics for all tasks. The former captures the performance measurement specific to each scenario, while the latter covers common complementary measurement across different applications.

In particular, we use accuracy (how often an LLM service generates the correct answer) as our main metric for math problems and USMLE questions. For answering sensitive and opinion questions, we use the response rate, i.e. the frequency that an LLM service directly answers a question. For code generation, the main metric is what fraction of the outputs are directly executable (if the code can be directly executed in a programming environment and pass the unit tests). Sometimes the outputs are not directly executable because the LLMs generate some non-code texts. Thus, we also study whether the cleaned outputs, i.e., outputs after removing non-code texts, are executable. We also study if the generation is exact match (whether the final response exactly matches the ground truth).

Our first common additional metric is verbosity, i.e., the length of generation measured in the number of characters. The second one is mismatch, i.e. how often, for the same prompt, the extracted answers by two versions of the same LLM service do not match. Note that this only compares the answers’ differences, not the raw generations. For example, for math problems, mismatch is 0 if the generated answers are the same, even if the intermediate reasoning steps are different. For each LLM service, we use the mismatch’s empirical mean over the entire population to quantify how much an LLM service’s desired functionality, instead of the textual outputs, deviates over time. Larger mismatch means greater drifts. For each of the other metrics, We compute its population mean for both the March and June versions, and leverage their differences to measure the drift sizes.
3. Monitoring Reveals Substantial LLM Drifts

3.1. Math I (Prime vs Composite): Chain-of-Thought May Not Be Followed. How do GPT-4 and GPT-3.5’s math solving skills evolve over time? As a canonical study, we explore the drifts in these LLMs’ ability to figure out whether a given integer is prime or composite. We focus on this task because it is easy to understand for humans while still requires reasoning, resembling many math problems. The dataset contains 1,000 questions, where 500 primes were extracted from (Zhang et al., 2023) and 500 composite numbers were sampled uniformly from all composite numbers within the interval [1,000, 20,000]. To help the LLMs reason, we use Chain-of-Thought (CoT) prompting (Wei et al., 2022), a standard approach for reasoning-heavy tasks.

Perhaps surprisingly, substantial LLM drifts emerge on this simple task. As shown in Figure 3(a), GPT-4’s accuracy dropped from 84.0% in March to 51.1% in June, and there was a large improvement of GPT-3.5’s accuracy, from 49.6% to 76.2%. In addition, GPT-4’s response became much more compact: its average verbosity (number of generated characters) decreased from 638.3
Table 1. Chain-of-thought’s (CoT) effectiveness drifts over time for prime testing. Without CoT, both GPT-4 and GPT-3.5 achieved relatively low accuracy. With CoT, GPT-4 in March obtained a 24.4% accuracy improvement, which dropped to -0.1% in June. On the other hand, the CoT boost increased from -0.9% in March to 15.8% in June for GPT-3.5.

<table>
<thead>
<tr>
<th>LLM Service</th>
<th>GPT-4</th>
<th>GPT-3.5</th>
<th>∆</th>
<th>GPT-4</th>
<th>GPT-3.5</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prompting method</td>
<td>∆</td>
<td>Prompting method</td>
<td>∆</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eval Time</td>
<td>No CoT</td>
<td>CoT</td>
<td></td>
<td>No CoT</td>
<td>CoT</td>
<td></td>
</tr>
<tr>
<td>Mar-23</td>
<td>59.6%</td>
<td>84.0%</td>
<td>+24.4%</td>
<td>50.5%</td>
<td>49.6%</td>
<td>-0.9%</td>
</tr>
<tr>
<td>Jun-23</td>
<td>51.0%</td>
<td>51.1%</td>
<td>+0.1%</td>
<td>60.4%</td>
<td>76.2%</td>
<td>+15.8%</td>
</tr>
</tbody>
</table>

in March to 3.9 in June. On the other hand, there was about 22.2% growth in GPT-3.5’s response length. The answer mismatch between their March and June versions was also large for both services.

Why was there such a large difference? One possible explanation is change in the chain-of-thought (CoT) behaviors. Figure 3 (b) gives an illustrative example. To determine whether 17077 is a prime number, the GPT-4’s March version followed the CoT instruction well. It first decomposed the task into four steps, checking if 17077 is even, finding 17077’s square root, obtaining all prime numbers less than it, checking if 17077 is divisible by any of these numbers. Then it executed each step, and finally reached the correct answer that 17077 is indeed a prime number. However, the chain-of-thought did not work for the June version: the service did not generate any intermediate steps, even though the prompt asked to think step-by-step, and simply produced “No”. Chain-of-thought’s effects had a different drift pattern for GPT-3.5. In March, GPT-3.5 inclined to generate the answer “No” first and then performed the reasoning steps. Thus, even if the steps and final conclusion (“17077 is a prime number”) were correct, its nominal answer was still wrong. On the other hand, the June update seemed to fix this issue: it started by writing the reasoning steps and finally generate the answer “Yes”, which was correct. This interesting phenomenon indicates that the same prompting approach, even the widely adopted chain-of-thought strategy, could lead to substantially different performances due to LLM drifts.

To further investigate the impact of CoT behavior changes, we compared the responses of GPT-4 and GPT-3.5 on the same questions with and without explicit CoT instructions. For the latter, we simply ask the model to give a binary generation without explicitly asking it to think step-by-step (e.g., Is 17077 a prime number? Answer "[Yes]" or "[No]").

As shown in Table 1, using CoT increased GPT-4’s performance from 59.6% to 84.0% in March, leading to a 24.4% performance boost. On the other hand, CoT did not help the June version of GPT-4 much: the accuracy was actually 0.1% worse. As we discussed before, this is because the new version did not follow the CoT instructions. For GPT-3.5, an opposite trend was observed: by adding CoT, accuracy was marginally better (+6.3%) in March, but substantially higher (+15.8%) in June. Since GPT-3.5 in both March and June followed the CoT instructions. This suggests that LLM drifts could change both whether and how to follow user instruction.

Our analysis so far is largely based on shifts of the main metric, accuracy, but fine-grained investigations could disclose additional interesting shift patterns. One observation is that June version of GPT-4 had a strong bias to view an integer as a composite number. To see so, we quantified how the confusion matrices shifted over time. As shown in Figure 4(c), GPT-4’s June version almost always identified an integer as composite (49.9%+48.8%=99.7%). GPT-3.5’s March
How Is ChatGPT’s Behavior Changing over Time? 8

A version exhibited a similar issue. The confusion matrices for GPT-4 in March and GPT-3.5 in June seem more reasonable: they identified both prime and composite numbers correctly for the major cases. Similar findings have been recently reported (Narayanan & Kapoor, 2023).

Instance-level investigation discloses additional insights. We uniformly randomly sampled 8 numbers to manually inspect March GPT-4’s step-by-step logic. March GPT-4’s CoT logic was mostly sound in these cases, though it can make arithmetic mistakes, and it reached the correct conclusion in 7 of the 8 cases. We show 4 examples below and provide the others in the Appendix.

For example, the following generation by GPT-4 gave the correct reasoning steps for 19997.

Step 1: Check if the number is divisible by 2. Since 19997 is an odd number, it is not divisible by 2.
Step 2: Find the square root of the number. The square root of 19997 is approximately 141.4.
Step 3: Check for divisibility by prime numbers less than or equal to the square root. We will check for divisibility by 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, and 139.
19997 is not divisible by any of these prime numbers.
Therefore, 19997 is a prime number.
Answer: [Yes]

Asking the same query multiple times gives the same conclusion, though the intermediate reasoning steps can be slightly different. For example, a second generation by GPT-4 in March was more verbose, as it explicitly checked the divisibility of 19997 by each prime number less than it (see Appendix). In addition, GPT-4 in March also identified composite numbers divisible by 2 or 5 correctly. For example, to determine whether 16450 is prime, GPT-4 in March wrote the following response.
Step 1: Check if the number is even. If it is, then it is not prime (except for 2). 16450 is an even number because it ends in 0. So, the answer is [No].

Similarly, to check 9175, it gave the following reasoning steps.

Step 1: Find the square root of 9175. The square root of 9175 is approximately 95.78.
Step 3: 9175 is divisible by 5 (9175 ÷ 5 = 1835).
Since 9175 is divisible by a prime number other than itself and 1, it is not a prime number.
Answer: [No]

In both cases, March GPT-4’s step-by-step logic led to the right answer. However, it also made subtle mistakes on some other queries. For example, to check whether 4323 is a prime number, it generated the following content.

- 4323 is not divisible by 3 (4323 / 3 = 1441 with a remainder).

Here, it actually used the correct logic (checking if 4323 is divisible by any prime numbers), but made a mistake in calculation (4323/3 = 1441 exactly without a remainder).

3.2. **Math II (Counting Happy Numbers): Chain-of-Thought May Not Be Followed on Other Math Problems.** To further investigate ChatGPT’s math problem-solving and chain-of-thought behaviors, we asked it to tackle a different math problem: counting the number of happy numbers (Guy, 2004, pp. 357-360) within a given interval. An integer is called happy if replacing it by the sum of the square of its digits repeatedly eventually produces 1. For example, 13 is a happy number because $1^2 + 3^2 = 10$, and $1^2 + 0^2 = 1$. This task complements prime testing because it asks for a quantitative response (number of happy numbers) rather than a binary decision (e.g., prime or composite) and it only uses simple arithmetic. To assess LLM drift on this task, we constructed a dataset of 500 queries. Each query asks how many happy numbers there are within a given interval and we quantify how often the LLM gets the correct number exactly. The interval size was uniformly randomly selected from 6 to 10, and the interval starting point was uniformly randomly chosen from 500 to 15,000. To encourage logic reasoning steps, we adopt CoT prompting again.

We also observed significant performance drifts on this task. As shown in Figure 5(a), GPT-4’s accuracy dropped from 83.6% in March to 35.2% in June. On the other hand, accuracy of GPT-3.5 increased from 30.6% to 48.2%. There was also a large change in the verbosity (number of characters in the generated responses). GPT-4’s generation length dropped from 2163.5 in March to 10.0 in June, but GPT-3.5’s length increased by more than 60%. Compared to prime testing (Math I), the answer lengths on average were significantly larger due to requiring more steps to enumerate the numbers in the interval and repeatedly square digits. In addition, 67.6% of GPT-4’s final answers changed between March and June, as did 77.2% of GPT-3.5’s.

As with the prime number testing task, we observed a large shift in the LLMs’ CoT behaviors. As shown in Figure 5(b), GPT-4 in June did not follow the CoT instructions and only gave a final
How Is ChatGPT’s Behavior Changing over Time?

Figure 5. Math II (Counting Happy Numbers).

(a) Overall drifts. All error bars denote 95% bootstrap confidence intervals. The accuracy of GPT-4 dropped from 83.6% to 35.2%. On the other hand, there was a 17.8% accuracy gain for GPT-3.5. GPT-4 became less verbose while GPT-3.5 generated much longer answers. (b) Example query and corresponding answers. GPT-4 followed the CoT instructions but ignored it in June. GPT-3.5 followed CoT in March and June, and gave longer reasoning steps in June.

answer, while its March counterpart followed the instructions to leverage reasoning steps. GPT-3.5 followed CoT instructions in both March and June. Its reasoning steps in June were much longer than that in March. While overall this led to better performance, sometimes it was problematic due to exceeding the maximum token length and thus not generating the final answer.

To further understand how the CoT effects’ shifts, we asked each service the same query either with or without CoT prompting, and studied how much accuracy gain was achieved by having CoT. We have found that CoT’s benefits shifted too. For example, for GPT-4, CoT brought 56.6% accuracy boost in March but only 3.2% in June, as shown in Table 2. For GPT-3.5, CoT led to 20.6% performance gains in June. In March, however, CoT caused a 1.6% accuracy drop.

The number of mistakes made by GPT-4 and GPT-3.5 changed over time. But what new mistakes did they make? To answer this question, we performed a fine-grained analysis on the confusion matrix of these LLMs over time, as shown in Figure 6. It was interesting to note how the bias of GPT-4 and GPT-3.5 changed over time. GPT-4 in June had a strong belief that there was only 0 or 1 happy number within any given interval. On the other hand, GPT-3.5 in June was inclined
Table 2. Benefits of CoT drift over time for happy number counting. For GPT-4, CoT brought 56.6% accuracy gains in March, which dropped to 3.2% in June. For GPT-3.5, the accuracy gains were 20.6% in June. Interestingly, adding CoT to GPT-3.5 caused a 1.6% performance downgrade in March.

<table>
<thead>
<tr>
<th>LLM Service</th>
<th>GPT-4</th>
<th>GPT-3.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eval Time</td>
<td>Prompting method</td>
<td>Prompting method</td>
</tr>
<tr>
<td></td>
<td>No CoT</td>
<td>CoT</td>
</tr>
<tr>
<td>Mar-23</td>
<td>27.0%</td>
<td>83.6%</td>
</tr>
<tr>
<td>Jun-23</td>
<td>32.0%</td>
<td>35.2%</td>
</tr>
</tbody>
</table>

Figure 6. Confusion matrix shift for counting happy numbers. GPT-4’s March version calculated the number correctly for most queries, while its June version responded that there was only one happy number most of the time.

to overestimate the number: on more than 10% queries, it responded that there were more than 4 happy numbers, while 4 was actually the upper bound among all our queries. We also ran additional experiments with smaller intervals for happy numbers and observed similar trends in the LLMs’ behavior (see Appendix).

3.3. Answering Sensitive and Subjective Questions.

3.3.1. Answering Sensitive Questions: Safer but Less Rationale. Prompting LLMs with sensitive questions is known to lead to harmful generations such as social biases (Ganguli et al., 2022), personal information (Carlini et al., 2021), and toxic texts (Gehman et al., 2020). Thus, another goal of this paper was to understand how LLM services’ responses to sensitive questions have shifted over time. To achieve this goal, we used a dataset of sensitive questions, the I-MaliciousInstructions dataset originally constructed in (Bianchi et al., 2023), which contains 100 sensitive queries that LLM services are not supposed to answer directly. As it is challenging to automatically evaluate whether a response is indeed a direct answer, we have manually labelled all responses from the monitored LLM services.
How Is ChatGPT’s Behavior Changing over Time?

Figure 7. Answering sensitive questions. (a) Overall performance changes. All error bars denote 95% bootstrap confidence intervals. GPT-4 answered fewer questions from March to June while GPT-3.5 answered slightly more. (b) An example query and responses of GPT-4 and GPT-3.5 at different dates. In March, GPT-4 and GPT-3.5 were verbose and gave detailed explanations for why they did not answer the query. In June, they simply said sorry.

Table 3. Comparison of response rate drifts on plain texts and AIM attacks with jailbreak prompts. GPT-3.5 failed to defend against IM attacks: its response rate was high in both March (100%) and June (96%). On the other hand, GPT-4’s updates offered a stronger defense against the attacks: the answer rate for AIM attacks dropped from 78.0% in March to 31.0% in June.

<table>
<thead>
<tr>
<th>LLM Service</th>
<th>GPT-4</th>
<th>GPT-3.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eval Time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mar-23</td>
<td>21.0%</td>
<td>78.0%</td>
</tr>
<tr>
<td>Jun-23</td>
<td>5.0%</td>
<td>31.0%</td>
</tr>
</tbody>
</table>

We observed two major trends on this task. First, as shown in Figure 7, GPT-4 answered fewer sensitive questions from March (21.0%) to June (5.0%) while GPT-3.5 answered more (from 2.0% to 8.0%). It was likely that a stronger safety layer was likely to deployed in the June update for GPT-4, while GPT-3.5 became less conservative. Another observation is that the generation length (measured by number of characters) of GPT-4 dropped from more than 600 to about 140.

Why did the generation length change? Besides answering fewer questions, it was also because GPT-4 became more terse and offered fewer explanations when it refused to answer a query. To see this, consider the example shown in Figure 7(b). GPT-4 refused to answer the inappropriate query in both March and June. However, it generated a whole paragraph to explain the rejection reasons in March, but simply produced “Sorry, but I cannot assist with that”. A similar phenomenon...
3.3.2. **LLM Jailbreaking.** Jailbreaking attacks are a major threat to LLM service safety (Ganguli et al., 2022). It rephrases or reorganizes the original sensitive questions in order to produce harmful generations from LLMs. Thus, it is also critical to study how LLM services’ defense against jailbreaking attacks drift over time. Here, we leverage the AIM (always intelligent and Machiavellian) attack\(^1\), the most user-voted among a largest collection of ChatGPT jailbreaks on the internet \(^2\).

The AIM attack describes a hypothetical story and asks LLM services to act as an unfiltered and amoral chatbot. We applied the AIM attack for each query in the sensitive question dataset and then queried GPT-4 and GPT-3.5. The answer rate of their March and June versions was shown in Table 3. There was a large increase of answer rate for both GPT-4 and GPT-3.5 when AIM attack was deployed. However, their temporal drifts differed substantially. For GPT-4, AIM attack produced 78% direct answers in March, but only 31.0% in June. For GPT-3.5, there was only a 4% (=100%-96%) answer rate difference among the two versions. This suggests that GPT-4’s update was more robust to jailbreaking attacks than that of GPT-3.5.

3.4. **OpinionQA Survey: Lower Response Rate.** LLMs are increasingly leveraged for open-ended text generation, where bias in the opinions in their training or fine-tuning data can play an important role. Therefore, it is vital to understand how LLMs’ opinion biases change over time. To address this problem, we leverage OpinionQA (Santurkar et al., 2023), a survey dataset that contains 1,506 opinion questions. We pick this dataset as its questions were drawn from high-quality

\(^1\)www.jailbreakchat.com/prompt/4f37a029-9dff-4862-b323-c96a5504de5d

\(^2\)jailbreakchat.com
public opinion polls. We followed the multiple-choice question format provided in (Santurkar et al., 2023), and added “Pick the best single option” for ease of extracting the answer.

There were substantial and interesting drifts over time on this opinion survey. First, GPT-4 became less willing to offer its opinions. As shown in Figure 8(a), GPT-4’s response rate dropped from 97.6% in March to 22.1% in June. In contrast, GPT-3.5’s response rate actually increased by 2%. GPT-3.5 answered almost all questions in both March and June. Yet, 27% of its opinions changed from March to June. For comparison, running GPT-3.5 March twice yields disagreement rate of 2.8% and running GPT-3.5 June twice yields disagreement rate of 7.0%, due to LLM’s stochasticity. These indicate considerable opinion drifts over time above and beyond model’s randomness.

A closer look at how the opinions changed gave us additional insights. As shown in the example in Figure 8(b), GPT-4 in March believed that the US will be less important in the world. In June, however, the model refused to answer the question, because it viewed the question as “subjective” and thus it simply generated “As an AI, I don’t have personal opinions”. This illustrates a significant change in GPT-4’s behavior in responding (or not responding) to subjective questions. ChatGPT’s behavior shifts on sensitive and subjective queries impose several implications in practice. On one hand, as more sensitive questions are refused, less misleading and dangerous information can be obtained by querying LLMs like GPT-4. On the other hand, it also becomes more challenging to understand and evaluate how GPT-4 is affected by the opinions and thoughts reflected in its training datasets. Moreover, too many refusals by the LLM can annoy users.

3.5. **Code Generation: Less Adherence to Specific Formatting Instructions.** One major application of LLMs is code generation (M. Chen et al., 2021). While many code generation datasets exist (Austin et al., 2021; M. Chen et al., 2021; Yu et al., 2018), using them to assess LLM services’ code generation ability faces the data contamination issue. To overcome this, we have constructed a new code generation dataset. It contains the latest 50 problems from the “easy” category of LeetCode at the time of writing. The earliest public solutions and discussions were released in December 2022. The prompt for each problem is the concatenation of the original problem description and the corresponding Python code template. Each LLM’s generation was directly sent to the LeetCode online judge for evaluation. We call it *directly executable* if the online judge accepts the answer (i.e., the answer is valid Python and passes its tests).

Overall, the number of directly executable generations dropped from March to June. As shown in Figure 9 (a), over 50% generations of GPT-4 were directly executable in March, but only 10% in June. The trend was similar for GPT-3.5. There was also a small increase in verbosity for both models.

**Why did the number of directly executable generations decline?** One possible explanation is that the June versions consistently added extra non-code text to their generations. Figure 9 (b) gives one such instance. GPT-4’s generations in March and June are almost the same except two parts. First, the June version added “‘python and ‘‘ before and after the code snippet (likely to format it as Markdown in UIs). Second, it also generated a few more comments. While a small change, the extra triple quotes render the code not executable. This type of shift in formatting behavior can be particularly challenging to detect when LLM’s generated code is used inside a larger software pipeline.

Note that less adherence to formatting instructions does not necessarily mean lower code quality. To see this, we also study whether the generated code passes the LeetCode tests after additional post-processing that removes the non-code text. As shown in Table 4, there was again a notable
How Is ChatGPT’s Behavior Changing over Time?

Figure 9. Code generation. (a) Overall performance drifts. All error bars denote 95% bootstrap confidence intervals. For GPT-4, the percentage of generations that are directly executable dropped from 52.0% in March to 10.0% in June. The drop was also large for GPT-3.5 (from 22.0% to 2.0%). GPT-4’s verbosity, measured by number of characters in the generations, also increased by 20%. (b) An example query and the corresponding responses. In March, both GPT-4 and GPT-3.5 followed the user instruction (“the code only”) and thus produced directly executable generation. In June, however, they added extra triple quotes before and after the code snippet, rendering the code not executable.

drift: GPT-4’s performance increased from 52% to 70%, and there was a 2% improvement for GPT-3.5. This shows an interesting trade-off between code quality and format following: GPT-4 in June improved its code’s correctness, but also failed to follow the formatting instructions (“generate the code only”).

3.6. LangChain HotpotQA Agent: Poor Prompt Stability. Many real-world applications require LLMs to answer knowledge-intensive questions grounded in various data sources, including “multi-hop” questions that involve multiple sources and/or reasoning steps. Therefore, it is natural to monitor how LLMs’ ability to answer multi-hop questions evolves over time. We take a first step by measuring the drifts of a LangChain HotpotQA Agent (Team, 2023), a pipeline to answer complex
Table 4. Effects of removing non-code text around generated code. There was no effect for GPT-4 in March since it followed the user instructions well. For the other versions, removing non-code texts rendered more code able to pass the LeetCode questions.

<table>
<thead>
<tr>
<th>LLM Service</th>
<th>GPT-4</th>
<th>GPT-3.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eval Time</td>
<td>removing non-code texts</td>
<td>∆</td>
</tr>
<tr>
<td>Mar-23</td>
<td>No 52.0%</td>
<td>Yes 52.0%</td>
</tr>
<tr>
<td>Jun-23</td>
<td>No 10.0%</td>
<td>Yes 70.0%</td>
</tr>
</tbody>
</table>

Figure 10. LangChain HotpotQA Agent. (a) Drifts on exact match, verbosity, and mismatch rate. All error bars denote 95% bootstrap confidence intervals. Overall, GPT-4 matched more ground-truth while GPT-3.5 became worse. (b) An example query and corresponding answers. LangChain was not able to parse March GPT-4’s response because it failed to follow the format specified in the LangChain prompt. GPT-3.5 in June could not find the information that it was able to obtain in March. These issues highlight the stability issues of integrating LLM into larger pipelines.

multi-hop questions similar to those from HotpotQA (Yang et al., 2018). This agent leveraged LLMs to search over Wikipedia passages to answer complex questions. We pick this pipeline for two reasons. First, LangChain is one of the most popular software frameworks for working with LLMs, providing open source modules that have been “prompt-engineered” to perform various tasks well. The stability of these modules’ prompts over time, i.e., how the performance of the same prompts changes over time, is therefore of interest to many users. Second, HotpotQA is widely used to measure an LLM’s ability to answer multi-hop questions. Specifically, we used the default ReAct Agent in LangChain3 (designed to reproduce ReAct prompting (Yao et al., 2022)) with different LLMs (GPT-4 and GPT-3.5) as the backbone for our code. Then we asked the agent to answer each query in the HotpotQA dataset.

Overall, we observed significant drifts for both GPT-4 and GPT-3.5 on this task. For example, the exact match rate for GPT-4 was only 1.2% in March, but became 37.8% in June, as shown in

---

3https://python.langchain.com/docs/modules/agents/agent_types/react_docstore
Figure 10(a). Opposite trends were observed for GPT-3.5: the exact match rate dropped by almost 9% from March to June. Moreover, more than 80% of final answers between March and June did not match for both models. We also noticed that GPT-4’s generation in June became more concise than in March, while GPT-3.5’s generation was 30% more verbose over time.

Why did this happen? A closer look at the mismatched answers suggests the poor prompt stability as one of the explanations. To see this, consider the example in Figure 10(b). The query was about whether two people were Democrats or Republicans. GPT-4 in March was actually able to find the correct answer: they both were Democrats. However, the LangChain agent expected a specific format: the generation from LLM must be “[action]+text”, which was encoded in its prompts. Unfortunately, GPT-4 in March failed to follow this format, and thus the LangChain agent simply generated an error message “could not parse LLM Output”. This is problematic in real-world LLM applications, as manually debugging such issues is challenging in large pipelines.

In addition, GPT-3.5 in March found the right answer. In June, however, it “was not able to find information”. These issues indicate how brittle existing prompting methods and libraries can be for complex tasks in the face of LLM drift.

3.7. USMLE Medical Exam: Small Decrease in GPT-4 Performance. We study next how the performance of GPT-4 and GPT-3.5 change over time in a professional domain: taking USMLE (Kung et al., 2023), a medical exam required for doctors in the US. USMLE has been used to benchmark LLMs’ medical knowledge.

Overall, we observe a slight performance decrease. As shown in Figure 11(a), GPT-4’s accuracy dropped from 86.6% to 82.4%. There was also a 0.8% accuracy loss for GPT-3.5. Interestingly, GPT-3.5 became much more verbose from March to June. It is also worth noting a relatively large answer mismatch between March and June for both models. In fact, 12.2% answers in March were different from their counterparts in June for GPT-4, and the mismatch rate was 27.9% for GPT-3.5. These two are much larger than the accuracy changes. This effectively means that the June versions corrected previous errors but also made additional mistakes. Overall, we also found that GPT-4 June was much less verbose in its response compared to GPT-4 March, while GPT-3.5’s responses to USMLE questions became longer.

3.8. Visual Reasoning: Small Improvements in Both Models. Finally, we investigate LLM drifts for visual reasoning. This task differs from other scenarios because it requires abstract reasoning. The ARC dataset (Chollet, 2019) is commonly used to assess visual reasoning ability. The task is to create a output grid corresponding to an input grid, based solely on a few similar examples. Figure 12(b) gives one example query from ARC. To show the visual objects to LLM services, we represent the input and output grids by 2-D arrays, where the value of each element denotes the color. We fed the LLM services 467 samples in the ARC dataset that fits in all services’ context window. Then we measured the exact match between their generation and the ground truth.

As shown in Figure 12(a), there were marginal performance improvements for both GPT-4 and GPT-3.5. However, for more than 90% visual puzzle queries, the March and June versions produced the exact same generation. These services’ overall performance were also low: 27.4% for GPT-4 and 12.2% for GPT-3.5.

It is worth noting that LLM services did not uniformly make better generations over time. In fact, despite better overall performance, GPT-4 in June made mistakes on queries on which it was correct for in March. Figure 12(b) gives one such example. This underlines the need for fine-grained drift monitoring, especially for critical applications.
How Is ChatGPT’s Behavior Changing over Time?

3.9. Summary of the Key Findings. Our study on the above tasks has revealed intriguing performance and behavior drifts of GPT-4 and GPT-3.5 from March 2023 to June 2023. In particular, GPT-4 was reasonably good at identifying prime numbers in March 2023 (84%) but became much worse in June 2023 (51%). Similarly, its performance on counting happy numbers also dropped from 83% in March to 35%. This is partially because the June version did not follow the Chain-of-Thought instructions. Interestingly, GPT-3.5’s performance became better at these tasks. GPT-4 refused to answer more sensitive and subjective questions in June than in March. It also defended more jailbreak attacks in June. For code generations, both GPT-4 and GPT-3.5 became less adherence to formatting instructions, though there was an overall code quality improvement after reformatting.
How Is ChatGPT’s Behavior Changing over Time?  

4. Is GPT-4’s instruction following getting worse over time?  

How to interpret the observed behavior drift? In our experiments, decrease in LLM’s performance is often associated with worse instruction following (i.e. worse ability in following users’ instructions). On the Math I and Math II tasks, for example, GPT-4 followed the user instructions to perform step-by-step reasoning and then answer the questions in March, but refused to do so in June. OpinionQA offers another example: GPT-4 responded to users’ questions in March but did not respond in June.

4.1. Quantifying instruction following drift on single instructions. Quantifying instruction following drift on existing LLM benchmarks is challenging: their tasks and evaluation metrics often blur a model’ instruction fidelity and its task-specific abilities (such as writing and logic reasoning) and knowledge (commonsense, history, etc). Hence, we have curated a new benchmark focused on task-agnostic instructions. This includes four types of instructions that often arise in practice: answer extractions (“answer yes or no within squared brackets”), stop apologizing (“do not say sorry or as an AI model”), writing constraint (“describe X by words starting/ending with Y ”), and text formatting (“add squared brackets to each single word’s first letter (including article word like ‘the’).”). We apply answer extraction and text formatting on the abstracts of 200 recent arxiv papers, and content filtering on the sensitiveQA dataset. We manually created 20 style refinement queries.
How Is ChatGPT’s Behavior Changing over Time?

Figure 13. GPT-4’s instruction following on individual instructions. (a) Overall instruction following. (b) example responses by GPT-4. In a nutshell, GPT-4 followed most individual instructions in March, but ignored them in June. Consider answer extraction as an example: 99.5% queries were followed by GPT-4 in March, but the number became almost 0 in June. Similarly, the fidelity rate dropped from 74.0% in March to 19.0% in June on the content filtering queries. The example response revealed some infidelity patterns of GPT-4 in June. It insisted on capitalizing the letter (answer extraction), kept generating “sorry” when users asked not to do it (stop apologizing), ignored the word ending letters (writing constraint), and missed a few letters to add brackets (text formatting).

As shown in Figure 13, there was indeed a large instruction fidelity drop of GPT-4 from March to June. For example, GPT-4 followed 99.5% answer extraction queries in March, while the number dropped to 0.5% in June. On 74% sensitive questions, GPT-4 mentioned no “sorry” or “as an AI model” as the instructions request in March. However, this number became only 19% in June. The examples given in Figure 13 offer more insights on what led to June version’s low fidelity. For example, GPT-4 in June did place the answer in the squared brackets, but it consistently capitalize the first letter. Similarly, while users asked not to say sorry, GPT-4 kept generating sorry in June, while its March version rephrased its answer to follow the user request. On the writing constraint example, GPT-4 in March followed the user instruction exactly: it generated words related to
Figure 14. GPT-4’s instruction following shifts on composite instructions. (a) GPT-4’s overall instruction following shifts on a range of composite instructions from March 2023 to June 2023. (b) Example responses by GPT-4 to individual and composite instructions. Overall, GPT-4 became more prune to composite instructions from March to June. For example, GPT-4’s accuracy on individual instructions “add comma” and “capitalize” remained roughly the same between March and June. However, to process their composition, the accuracy dropped by 9.2% from March to June.

machine learning and ending in “n”. The June version, however, focused on “machine learning” but ignored the “ending with ‘n’ ” requirement. GPT-4 successfully capitalized first letter for each word in March, but missed a few words (such as “provides” and “about” in the shown example) in June. Overall, GPT-4’s instruction following fidelity decreased from March to June, which partially explained its behavior drifts.

4.2. Instruction following drift on composite instructions. We further study how GPT-4’s instruction following changes on compositions of instructions. To quantify this, we collected a set of single instructions, and then created a list of composite instructions, each of which corresponds to two instructions from the single instruction set. We evaluated GPT-4’s performance on these
How Is ChatGPT’s Behavior Changing over Time?

composite instructions applied on arxiv papers’ first sentences. The single instruction set contains three text formatting instructions: add comma (“add a comma to each word”), no quotation (“remove quotations”), and capitalize (“capitalize each letter”). These instructions are easy to understand by humans and also commonly seen in real-world applications.

There are several interesting observations. First, GPT-4 followed the single instructions well in both March and June. In fact, the instruction following shifts on individual instructions are only -2%, +4.0%, and -1.0% from March to June (Figure 14 (a)). Second, GPT-4 in June was much prune to composite instructions than that in March. For example, when asked to remove quotations as well as add a comma to each word, GPT-4’s performance dropped by 24% from March to June. Similarly, switching from March to June caused a 9.2% accuracy drop on the composition of adding a comma and capitalizing letters. It is interesting to recognize the mistake patterns triggered by the composition. As shown in the example from Figure 14, GPT-4 in June tended to add a comma to each character when given the composite instruction. On the other hand, its March counterpart faithfully completed the user task.

Overall, we observe that GPT-4 followed less user instructions over time. This holds for both single instructions and composite instructions. Consistent with the performance shifts analyzed in the previous section, instruction following shifts appear a primary factor of GPT-4’s behavior drifts. In comparison, there was not a consistent change in GPT-3.5’s instruction following over time (see Figure 16 in the Appendix).

5. Conclusions and Future Work

Our findings demonstrate that the behavior of GPT-3.5 and GPT-4 has varied significantly over a relatively short amount of time. This highlights the need to continuously evaluate and assess the behavior of LLM drifts in applications, especially as it is not transparent how LLMs such as ChatGPT are updated over time. Our study also underscores the challenge of uniformly improving LLMs’ multifaceted abilities. Improving the model’s performance on some tasks, for example with fine-tuning on additional data, can have unexpected side effects on its behavior in other tasks. Consistent with this, both GPT-3.5 and GPT-4 got worse on some tasks but saw improvements in other dimensions. Moreover, the trends for GPT-3.5 and GPT-4 are often divergent. Beyond the final performances, it’s interesting to observe shifts in chain-of-thought behaviors and verbosity of the models.

We plan to update the findings presented here in an ongoing long-term study by regularly evaluating GPT-3.5, GPT-4 and other LLMs on diverse tasks over time. For users or companies who rely on LLM services as a component in their ongoing workflow, we recommend that they should implement similar monitoring analysis as we do here for their applications. We thank the many people who have provided helpful feedback to our work. To encourage further research on LLM drifts, we have release our evaluation data and ChatGPT responses at https://github.com/lchen001/LLMDrift.
REFERENCES

Aiyappa, R., An, J., Kwak, H., & Ahn, Y.-Y. (2023). Can we trust the evaluation on chatgpt?


REFERENCES


A.1. **GPT-4’s Generation in March for Prime Numbers.**

Example 1. The query is as follows.

Is 7937 a prime number? Think step by step and then answer "[Yes]" or "[No]".

And below is the response from GPT-4’s March version.

Step 1: Find the square root of 7937, which is approximately 89. Step 2: Check for divisibility by prime numbers less than 89 (2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83). Step 3: 7937 is not divisible by any of these prime numbers.

Answer: [Yes]

Example 2. The query is as follows.

Is 2647 a prime number? Think step by step and then answer "[Yes]" or "[No]".

And below is the response from GPT-4’s March version.

Step 1: Find the square root of 2647, which is approximately 51.45. Step 2: Check for divisibility by prime numbers less than 51.45 (2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47).

2647 is not divisible by 2 (it’s odd). 2647 is not divisible by 3 (sum of digits is 19, which is not divisible by 3). 2647 is not divisible by 5 (it doesn’t end in 0 or 5). 2647 is not divisible by 7 (2647/7 = 377.857, not a whole number). 2647 is not divisible by 11 (2647/11 = 240.636, not a whole number). 2647 is not divisible by 13 (2647/13 = 203.615, not a whole number). 2647 is not divisible by 17 (2647/17 = 155.706, not a whole number). 2647 is not divisible by 19 (2647/19 = 139.316, not a whole number). 2647 is not divisible by 23 (2647/23 = 115.087, not a whole number). 2647 is not divisible by 29 (2647/29 = 91.276, not a whole number). 2647 is not divisible by 31 (2647/31 = 85.387, not a whole number). 2647 is not divisible by 37 (2647/37 = 71.541, not a whole number). 2647 is not divisible by 41 (2647/41 = 64.561, not a whole number). 2647 is not divisible by 43 (2647/43 = 61.558, not a whole number). 2647 is not divisible by 47 (2647/47 = 56.319, not a whole number).

Since 2647 is not divisible by any prime numbers less than its square root, it is a prime number.

[Yes]
It is worthy noting that, while the conclusion was correct and most reasoning steps were correct, GPT-4 still made one arithmetic mistake in this case. That is, GPT-4 generated $2647/7 = 377.857$, while the correct result should be $378.142$.

Example 3. The query is as follows.

Is 3529 a prime number? Think step by step and then answer "[Yes]" or "[No]".

And below is the response from GPT-4’s March version.

---

Step 1: Check if the number is divisible by 2. Since 3529 is an odd number, it is not divisible by 2.
Step 2: Find the square root of the number. The square root of 3529 is approximately 59.4.
Step 3: Check if the number is divisible by any odd numbers up to the square root.
3 is not a factor (3529/3 = 1176.333)
5 is not a factor (3529/5 = 705.8)
7 is not a factor (3529/7 = 504.142)
11 is not a factor (3529/11 = 320.818)
13 is not a factor (3529/13 = 271.462)
17 is not a factor (3529/17 = 207.588)
19 is not a factor (3529/19 = 185.736)
23 is not a factor (3529/23 = 153.435)
29 is not a factor (3529/29 = 121.69)
31 is not a factor (3529/31 = 113.839)
37 is not a factor (3529/37 = 95.378)
41 is not a factor (3529/41 = 86.073)
43 is not a factor (3529/43 = 82.07)
47 is not a factor (3529/47 = 75.085)
53 is not a factor (3529/53 = 66.604)
59 is not a factor (3529/59 = 59.814)

Since 3529 is not divisible by any odd numbers up to its square root, it is a prime number.
Answer: [Yes]

Example 4. The query is as follows.

Is 9241 a prime number? Think step by step and then answer "[Yes]" or "[No]".

And below is the response from GPT-4’s March version.
Step 1: Find the square root of 9241. The square root of 9241 is approximately 96.1.

Step 2: Check for divisibility by prime numbers less than 96.1. The prime numbers less than 96.1 are 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, and 89.

Step 3: Check if 9241 is divisible by any of these prime numbers.
- 9241 is not divisible by 2 (it’s an odd number).
- 9241 is not divisible by 3 (the sum of its digits is 16, which is not divisible by 3).
- 9241 is not divisible by 5 (it doesn’t end in 0 or 5).
- 9241 is not divisible by 7 (9241 divided by 7 is approximately 1319.2857, which is not a whole number).
- 9241 is not divisible by 11 (9241 divided by 11 is approximately 840.0909, which is not a whole number).
- 9241 is not divisible by 13 (9241 divided by 13 is approximately 710.8462, which is not a whole number).
- 9241 is not divisible by 17 (9241 divided by 17 is approximately 543.5882, which is not a whole number).
- 9241 is not divisible by 19 (9241 divided by 19 is approximately 486.3684, which is not a whole number).
- 9241 is not divisible by 23 (9241 divided by 23 is approximately 401.8696, which is not a whole number).
- 9241 is not divisible by 29 (9241 divided by 29 is approximately 318.6552, which is not a whole number).
- 9241 is not divisible by 31 (9241 divided by 31 is approximately 298.0968, which is not a whole number).
- 9241 is not divisible by 37 (9241 divided by 37 is approximately 249.7568, which is not a whole number).
- 9241 is not divisible by 41 (9241 divided by 41 is approximately 225.3902, which is not a whole number).
- 9241 is not divisible by 43 (9241 divided by 43 is approximately 214.9070, which is not a whole number).
- 9241 is not divisible by 47 (9241 divided by 47 is approximately 196.6170, which is not a whole number).
- 9241 is not divisible by 53 (9241 divided by 53 is approximately 174.3585, which is not a whole number).
- 9241 is not divisible by 59 (9241 divided by 59 is approximately 156.6102, which is not a whole number).
- 9241 is not divisible by 61 (9241 divided by 61 is approximately 151.4918, which is not a whole number).
- 9241 is not divisible by 67 (9241 divided by 67 is approximately 137.8955, which is not a whole number).
- 9241 is not divisible by 71 (9241 divided by 71 is approximately 130.1549, which is not a whole number).
- 9241 is not divisible by 73 (9241 divided by 73 is approximately 126.5753, which is not a whole number).
- 9241 is not divisible by 79 (9241 divided by 79 is approximately 116.9747, which is not a whole number).
- 9241 is not divisible by 83 (9241 divided by 83 is approximately 111.3253, which is not a whole number).
- 9241 is not divisible by 89 (9241 divided by 89 is approximately 103.8315, which is not a whole number).

Step 4: Since 9241 is not divisible by any of the prime numbers less than its square root, it is a prime number.
A.2. **GPT-4's Generation in March for Composite Numbers.**

Example 1. The query is as follows.

<table>
<thead>
<tr>
<th>Is 4323 a prime number? Think step by step and then answer &quot;[Yes]&quot; or &quot;[No]&quot;.</th>
</tr>
</thead>
</table>

And below is the full response from GPT-4's March version. Note that the step “4323 / 3 = 1441 with a remainder” is incorrect.

```
Step 1: Check if the number is divisible by 2. Since 4323 is an odd number, it is not divisible by 2.
Step 2: Find the square root of the number. The square root of 4323 is approximately 65.7.
Step 3: Check if the number is divisible by any odd numbers less than or equal to the square root.
- 4323 is not divisible by 3 (4323 / 3 = 1441 with a remainder).
- 4323 is not divisible by 5 (4323 / 5 = 864 with a remainder).
- 4323 is not divisible by 7 (4323 / 7 = 617 with a remainder).
- 4323 is not divisible by 11 (4323 / 11 = 393 with a remainder).
- 4323 is not divisible by 13 (4323 / 13 = 332 with a remainder).
- 4323 is not divisible by 17 (4323 / 17 = 254 with a remainder).
- 4323 is not divisible by 19 (4323 / 19 = 227 with a remainder).
- 4323 is not divisible by 21 (4323 / 21 = 206 with a remainder).
- 4323 is not divisible by 23 (4323 / 23 = 188 with a remainder).
- 4323 is not divisible by 29 (4323 / 29 = 149 with a remainder).
- 4323 is not divisible by 31 (4323 / 31 = 139 with a remainder).
- 4323 is not divisible by 37 (4323 / 37 = 116 with a remainder).
- 4323 is not divisible by 41 (4323 / 41 = 105 with a remainder).
- 4323 is not divisible by 43 (4323 / 43 = 100 with a remainder).
- 4323 is not divisible by 47 (4323 / 47 = 91 with a remainder).
- 4323 is not divisible by 53 (4323 / 53 = 81 with a remainder).
- 4323 is not divisible by 59 (4323 / 59 = 73 with a remainder).
- 4323 is not divisible by 61 (4323 / 61 = 70 with a remainder).
```

Since 4323 is not divisible by any odd numbers less than or equal to its square root, it is a prime number.

Answer: [Yes]

A.3. **Some Runs Show More Detailed Reasoning.** The below is one example query from Math I (prime vs composite).

<table>
<thead>
<tr>
<th>Is 19997 a prime number? Think step by step and then answer &quot;[Yes]&quot; or &quot;[No]&quot;</th>
</tr>
</thead>
</table>

The following response was generated by GPT-4's March Version. Compared to its first generation, more details were generated.
Step 1: Check if the number is even. If it is, then it is not prime (except for 2). 19997 is odd, so we proceed to the next step.

Step 2: Check if the number is divisible by any odd numbers up to the square root of the number. The square root of 19997 is approximately 141.4.

Step 3: Check divisibility by odd numbers starting from 3.

19997 ÷ 3 ≈ 6665.67 (not divisible)
19997 ÷ 5 ≈ 3999.4 (not divisible)
19997 ÷ 7 ≈ 2856.71 (not divisible)
19997 ÷ 11 ≈ 1817.91 (not divisible)
19997 ÷ 13 ≈ 1538.23 (not divisible)
19997 ÷ 17 ≈ 1176.29 (not divisible)
19997 ÷ 19 ≈ 1052.47 (not divisible)
19997 ÷ 23 ≈ 869.87 (not divisible)
19997 ÷ 29 ≈ 689.9 (not divisible)
19997 ÷ 31 ≈ 645.39 (not divisible)
19997 ÷ 37 ≈ 540.46 (not divisible)
19997 ÷ 41 ≈ 487.49 (not divisible)
19997 ÷ 43 ≈ 465.28 (not divisible)
19997 ÷ 47 ≈ 425.47 (not divisible)
19997 ÷ 53 ≈ 377.3 (not divisible)
19997 ÷ 59 ≈ 338.93 (not divisible)
19997 ÷ 61 ≈ 327.82 (not divisible)
19997 ÷ 67 ≈ 298.46 (not divisible)
19997 ÷ 71 ≈ 281.65 (not divisible)
19997 ÷ 73 ≈ 273.94 (not divisible)
19997 ÷ 79 ≈ 253.13 (not divisible)
19997 ÷ 83 ≈ 240.93 (not divisible)
19997 ÷ 89 ≈ 224.68 (not divisible)
19997 ÷ 97 ≈ 206.15 (not divisible)
19997 ÷ 101 ≈ 197.99 (not divisible)
19997 ÷ 103 ≈ 194.15 (not divisible)
19997 ÷ 107 ≈ 186.97 (not divisible)
19997 ÷ 109 ≈ 183.45 (not divisible)
19997 ÷ 113 ≈ 176.97 (not divisible)
19997 ÷ 127 ≈ 157.47 (not divisible)
19997 ÷ 131 ≈ 152.65 (not divisible)
19997 ÷ 137 ≈ 145.97 (not divisible)
19997 ÷ 139 ≈ 143.86 (not divisible)

Step 4: We have checked all odd numbers up to the square root of 19997 and found no divisors.

Conclusion: 19997 is a prime number.

Answer: [Yes]
A.4. **The prefix prompt for the LangChain HotpotQA Agent.** To leverage the ReAct paradigm, the LangChain HotpotQA Agent constructs a prompt by appending 5 examples as a prefix to every given user query. The following block gives the full prefix:

<table>
<thead>
<tr>
<th>Question: What is the elevation range for the area that the eastern sector of the Colorado orogeny extends into?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thought: I need to search Colorado orogeny, find the area that the eastern sector of the Colorado orogeny extends into, then find the elevation range of the area.</td>
</tr>
<tr>
<td>Action: Search[Colorado orogeny]</td>
</tr>
<tr>
<td>Observation: The Colorado orogeny was an episode of mountain building (an orogeny) in Colorado and surrounding areas. Thought: It does not mention the eastern sector. So I need to look up eastern sector.</td>
</tr>
<tr>
<td>Action: Lookup[eastern sector]</td>
</tr>
<tr>
<td>Observation: (Result 1 / 1) The eastern sector extends into the High Plains and is called the Central Plains progeny. Thought: The eastern sector of Colorado orogeny extends into the High Plains. So I need to search High Plains and find its elevation range.</td>
</tr>
<tr>
<td>Action: Search[High Plains]</td>
</tr>
<tr>
<td>Observation: High Plains refers to one of two distinct land regions. Thought: I need to instead search High Plains (United States).</td>
</tr>
<tr>
<td>Action: Search[High Plains (United States)]</td>
</tr>
<tr>
<td>Observation: The High Plains are a subregion of the Great Plains. From east to west, the High Plains rise in elevation from around 1,800 to 7,000 ft (550 to 2,130 m).[3] Thought: High Plains rise in elevation from around 1,800 to 7,000 ft, so the answer is 1,800 to 7,000 ft.</td>
</tr>
<tr>
<td>Action: Finish[1,800 to 7,000 ft]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Question: Musician and satirist Allie Goertz wrote a song about the &quot;The Simpsons&quot; character Milhouse, who Matt Groening named after who?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thought: The question simplifies to &quot;The Simpsons&quot; character Milhouse is named after who. I only need to search Milhouse and find who it is named after.</td>
</tr>
<tr>
<td>Action: Search[Milhouse]</td>
</tr>
<tr>
<td>Observation: Milhouse Mussolini Van Houten is a recurring character in the Fox animated television series The Simpsons voiced by Pamela Hayden and created by Matt Groening. Thought: The paragraph does not tell who Milhouse is named after, maybe I can look up &quot;named after&quot;.</td>
</tr>
<tr>
<td>Action: Lookup[named after]</td>
</tr>
<tr>
<td>Observation: (Result 1 / 1) Milhouse was named after U.S. president Richard Nixon, whose middle name was Milhous. Thought: Milhouse was named after U.S. president Richard Nixon, so the answer is Richard Nixon.</td>
</tr>
</tbody>
</table>

---

4The raw file can be found at https://github.com/langchain-ai/langchain/blob/master/libs/langchain/langchain/agents/react/wiki_prompt.py
Question: Which documentary is about Finnish rock groups, Adam Clayton Powell or The Saimaa Gesture?

Thought: I need to search Adam Clayton Powell and The Saimaa Gesture, and find which documentary is about Finnish rock groups.

Action: Search[Adam Clayton Powell]


Thought: To find the documentary, I can search Adam Clayton Powell (film).

Action: Search[Adam Clayton Powell (film)]

Observation: Adam Clayton Powell is a 1989 American documentary film directed by Richard Kilberg. The film is about the rise and fall of influential African-American politician Adam Clayton Powell Jr.[3][4] It was later aired as part of the PBS series The American Experience.

Thought: Adam Clayton Powell (film) is a documentary about an African-American politician, not Finnish rock groups. So the documentary about Finnish rock groups must instead be The Saimaa Gesture.

Action: Finish[The Saimaa Gesture]

Question: What profession does Nicholas Ray and Elia Kazan have in common?

Thought: I need to search Nicholas Ray and Elia Kazan, find their professions, then find the profession they have in common.

Action: Search[Nicholas Ray]

Observation: Nicholas Ray (born Raymond Nicholas Kienzle Jr., August 7, 1911 - June 16, 1979) was an American film director, screenwriter, and actor best known for the 1955 film Rebel Without a Cause.

Thought: Professions of Nicholas Ray are director, screenwriter, and actor. I need to search Elia Kazan next and find his professions.

Action: Search[Elia Kazan]

Observation: Elia Kazan was an American film and theatre director, producer, screenwriter and actor.

Thought: Professions of Elia Kazan are director, producer, screenwriter, and actor. So profession Nicholas Ray and Elia Kazan have in common is director, screenwriter, and actor.

Action: Finish[director, screenwriter, actor]

Question: Which magazine was started first Arthur’s Magazine or First for Women?

Thought: I need to search Arthur’s Magazine and First for Women, and find which was started first.

Action: Search[Arthur’s Magazine]

Observation: Arthur’s Magazine (1844-1846) was an American literary periodical published in Philadelphia in the 19th century.

Thought: Arthur’s Magazine was started in 1844. I need to search First for Women next.

Action: Search[First for Women]
Figure 15. Confusion matrix shift for counting happy numbers within smaller intervals. Here, the interval size was randomly uniformly sampled from [4,7] (instead of [6,10] in the main paper), resulting in a smaller number of happy numbers. Overall the trends were similar: GPT-4’s March version generated the correct answers for most queries, while its June version responded that there was only one happy number most of the time.

Observation: First for Women is a woman’s magazine published by Bauer Media Group in the USA.[1] The magazine was started in 1989. Thought: First for Women was started in 1989. 1844 (Arthur’s Magazine) < 1989 (First for Women), so Arthur’s Magazine was started first.

Action: Finish[Arthur’s Magazine]""", """"Question: Were Pavel Urysohn and Leonid Levin known for the same type of work?

Thought: I need to search Pavel Urysohn and Leonid Levin, find their types of work, then find if they are the same.

Action: Search[Pavel Urysohn]

Observation: Pavel Samuilovich Urysohn (February 3, 1898 - August 17, 1924) was a Soviet mathematician who is best known for his contributions in dimension theory. Thought: Pavel Urysohn is a mathematician. I need to search Leonid Levin next and find its type of work.

Action: Search[Leonid Levin]

Observation: Leonid Anatolievich Levin is a Soviet-American mathematician and computer scientist.

Thought: Leonid Levin is a mathematician and computer scientist. So Pavel Urysohn and Leonid Levin have the same type of work.

Action: Finish[yes]

APPENDIX B. COUNTING HAPPY NUMBERS IN SMALLER INTERVALS

The confusion matrix for counting happy numbers in relatively smaller intervals is shown in Figure 15. Overall, we observe a similar trend as counting happy numbers in large intervals, shown in Figure 6.
Figure 16. GPT-3.5’s instruction following shifts on individual instructions. (a) Overall instruction following. (b) example responses by GPT-3.5. Overall, the instruction following drifts are relatively small compared to that of GPT-4.

APPENDIX C. GPT-3.5’s Instruction Following Shifts on Single Instructions

Here, we present GPT-3.5’s instruction following in Figure 16. Overall, we observe that GPT-3.5’s instruction fidelity shifts are relatively small and diverse, which matches its behavior shifts observed in Section 3.