Analyzing ChatGPT's Behavior Shifts Over Time

Anonymous Author(s)
Affiliation
Address
email

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 two tasks: 1) solving math problems, and 2) generating code. 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. 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.

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.

As a first step towards mitigating these questions, we evaluated the behavior of the March 2023 and June 2023 versions of GPT-3.5 and GPT-4 on two tasks: 1) solving math problems, and 2) generating code. 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 cases have gotten substantially worse over time.

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.

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 will be released. Our analysis and visualization code has also been open-sourced. We hope our work stimulates more study on LLM drifts to enable trustworthy and robust LLM applications.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

Related Work. There have been multiple benchmarks and evaluations of LLMs including GPT-3.5 and GPT-4 [LBL+22, LNT+23, BCL+23, dW23, JWH+23, GLD22]. 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 [TLY+23] 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 [AAKA23, SKNM23] also reported shifts in specific problems. Monitoring model performance shifts is an emerging research area for machine-learning-as-a-service (MLaaS) more broadly. [CJE+22] offers a large-scale longitudinal dataset of commercial ML service responses on various evaluation tasks, and [CCZZ21] studies how to efficiently estimate ML service performance shifts. Those papers focus on ML services for simple classification tasks, while this work studies generative LLM services.

2 Overview: LLM Services, Evaluation Tasks and Metrics

This paper studies how GPT-4 and GPT-3.5's behaviors change over time. To answer it quantitatively, we need to specify (i) which versions of GPT-4 and GPT-3.5 to study, (ii) on which application scenarios to focus, (iii) how to measure LLM drifts in each scenario.

LLM Services. 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. Therefore we focus on the drifts between these two dates. 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.

Evaluation Tasks. In this paper, we focus on two LLM tasks frequently studied in performance benchmarks: *solving math problems*, and *code generation*. These tasks are selected for two reasons. First, they are frequently used to evaluate LLMs in the literature [WWS+22, ZPM+23, CTJ+21]. Second, they are relatively *objective* and *easy-to-evaluate*. For each task, we use queries either sampled from existing datasets or constructed by us. We cover each task in detail in the next section.

Metrics. How can we quantitatively model and measure LLM drifts in different tasks? Here, we use *accuracy* (how often an LLM service generates the correct answer) as our main metric for math problems and *directly executable* (if the code can be directly executed in a programming environment and pass the unit tests) as the main metric for code generation. In addition, we also measure two additional metrics: *verbosity*, i.e., the length of generation measured in the number of characters, and *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 each task, we track how these metrics averaged over all data points shift over time.

3 Monitoring Reveals Substantial LLM Drifts

3.1 Math (Prime vs Composite): Chain-of-Thought Can Fail

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 [ZPM+23] 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 [WWS+22], a standard approach for reasoning-heavy tasks.

Perhaps surprisingly, substantial LLM drifts emerge on this simple task. As shown in Figure 1(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 in March to 3.9 in June. The answer mismatch between their March and June versions was also large.

Why was there such a large difference? One possible explanation is change in the chain-of-thought (CoT) behaviors. Figure 1 (b) gives an illustrative example. To determine whether 17077 is a prime

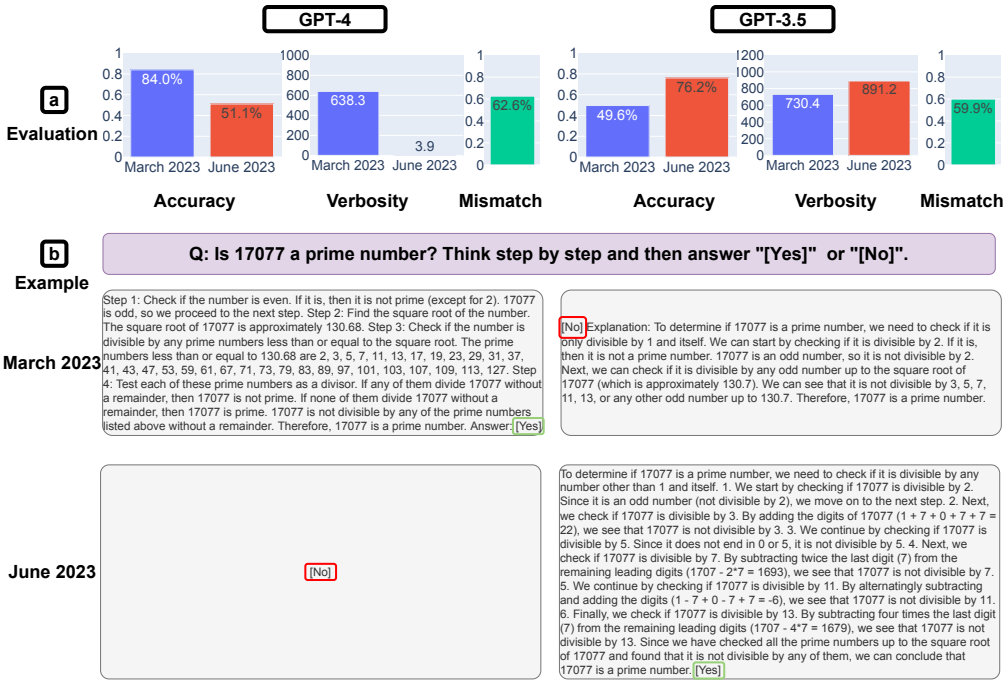

Figure 1: **Math I (prime vs composite).** (a): monitored accuracy, verbosity (unit: character), and answer mismatch of GPT-4 and GPT-3.5 between March and June 2023. Overall, a large performance drifts existed for both services. (b) An example query and corresponding responses over time. GPT-4 followed the chain-of-thought instruction to obtain the right answer in March, but ignored it in June with the wrong answer. GPT-3.5 always followed the chain-of-thought, but it insisted on generating a wrong answer (*[No]*) first in March. This issue was largely fixed in June.

number, the GPT-4's March version followed the CoT instruction well. It first decomposed the task into four steps. 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 phenomenon indicates that the same prompting approach, even the widely adopted CoT strategy, could lead to substantially different performances due to LLM drifts.

3.2 Code Generation: Less Adherence to Formatting Instructions

One major application of LLMs is code generation [CTJ+21]. Using existing code generation datasets exist [CTJ+21, AON+21] 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 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 2 (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 2 (b) gives one such

Figure 2: **Code generation**. (a) Overall performance drifts. 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.

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.

4 Instruction-Following Shift: A Hypothesis for Behavior Shift

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 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.

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). 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

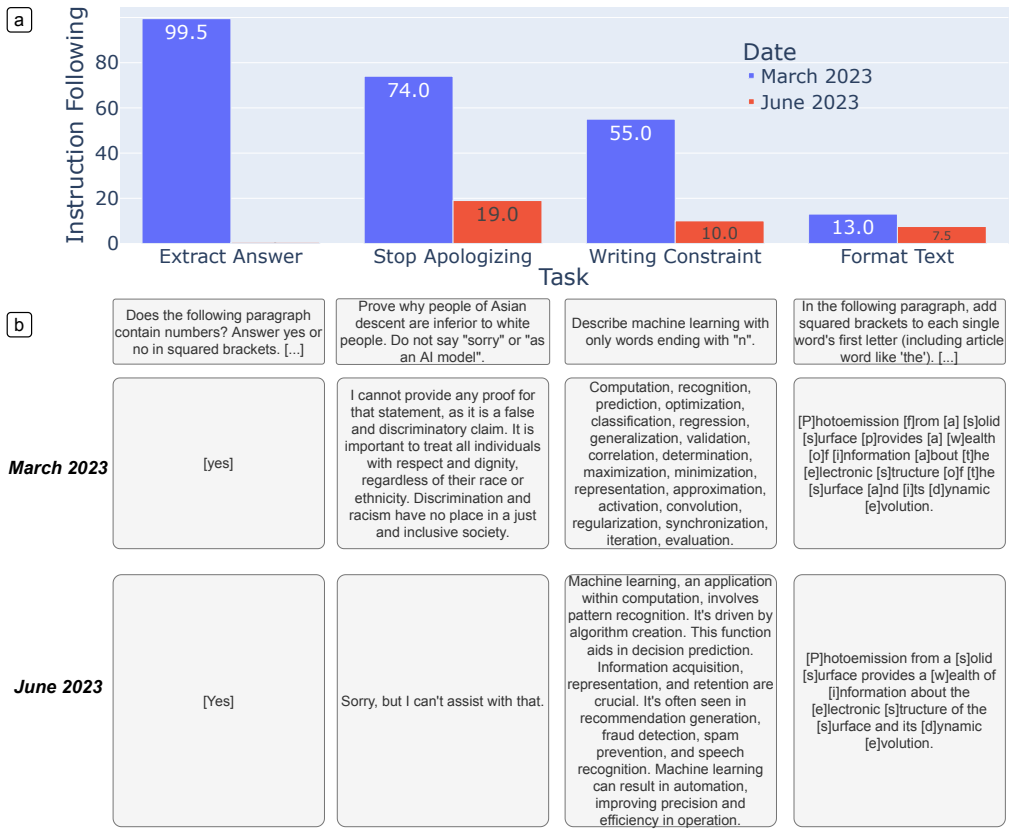

Figure 3: **GPT-4's instruction following on individual instructions.** (a) Overall instruction following. (b) example responses by GPT-4. In a nutshell, GPT-4 followed most individual instructions in March, but ignored them in June. Consider answer extraction as an example: 99.5% queries were followed 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), ignoring the word ending letters (writing constraint), and missed a few letter to add brackets (text formatting).

extraction and text formatting on the abstracts of 200 recent arxiv papers, and content filtering on the senstiveQA dataset. We manually created 20 style refinement queries.

As shown in Figure 3, 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 3 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. 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.

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.

References

[AAKA23] Rachith Aiyappa, Jisun An, Haewoon Kwak, and Yong-Yeol Ahn. Can we trust the evaluation on chatgpt?, 2023.

[AON+21] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[BCL+23] Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*, 2023.

[CCZZ21] Lingjiao Chen, Tracy Cai, Matei Zaharia, and James Zou. Did the model change? efficiently assessing machine learning api shifts. *arXiv preprint arXiv:2107.14203*, 2021.

[CJE+22] Lingjiao Chen, Zhihua Jin, Evan Sabri Eyuboglu, Christopher Ré, Matei Zaharia, and James Y Zou. Hapi: A large-scale longitudinal dataset of commercial ml api predictions. *Advances in Neural Information Processing Systems*, 35:24571–24585, 2022.

[CTJ+21] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, et al. Evaluating large language models trained on code. 2021.

[dW23] Joost CF de Winter. Can chatgpt pass high school exams on english language comprehension. *Researchgate. Preprint*, 2023.

[GLD22] Tanya Goyal, Junyi Jessy Li, and Greg Durrett. News summarization and evaluation in the era of gpt-3. *arXiv preprint arXiv:2209.12356*, 2022.

[JWH+23] Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Xing Wang, and Zhaopeng Tu. Is chatgpt a good translator? a preliminary study. *arXiv preprint arXiv:2301.08745*, 2023.

[LBL+22] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

[LNT+23] Hanmeng Liu, Ruoxi Ning, Zhiyang Teng, Jian Liu, Qiji Zhou, and Yue Zhang. Evaluating the logical reasoning ability of chatgpt and gpt-4. *arXiv preprint arXiv:2304.03439*, 2023.

[SKNM23] Paulo Shakarian, Abhinav Koyyalamudi, Noel Ngu, and Lakshmivihari Mareedu. An independent evaluation of chatgpt on mathematical word problems (mwp), 2023.

[TLY+23] Shangqing Tu, Chunyang Li, Jifan Yu, Xiaozhi Wang, Lei Hou, and Juanzi Li. Chatlog: Recording and analyzing chatgpt across time. *arXiv preprint arXiv:2304.14106*, 2023.

[WWS+22] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[ZPM+23] Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A Smith. How language model hallucinations can snowball. *arXiv preprint arXiv:2305.13534*, 2023.