

John Tan Chong Min
5 Jun 2024

The “strange” hallucinations of LLMs (in this case GPT-4o)

Recently, on LinkedIn, there has been a post highlighting the weaknesses of Large Language Models (LLMs) in doing certain tasks, such as counting letters. Here is an example:

ChatGPT 4o ▾

How many R's are there in strawberry



There are two "R"s in the word "strawberry."



Fig 1: Counting Rs in strawberry with GPT-4o

LLMs like GPT-4-Turbo, and recently, GPT-4o, are supposed to be able to do many complex tasks like summarization, sentiment analysis, translation. How come they struggle so much with a seemingly simple task like counting letters?

The Way LLMs see the world

Unlike humans like us, when LLMs see text, they do not see them as letter by letter, or word by word. In fact, they see things by tokens. What exactly is a token?

The mathematical answer is that tokens are generated via **Byte Pair Encoding (BPE)**, which could be the topic of another article altogether.

The simple answer is that tokens reflect the groups of letters which happen most often in web data.

The image shows the sentence "How many R's are there in strawberry" where each word is contained within a single, multi-colored rectangular token box. The colors of the boxes are: "How" (purple), "many" (green), "R's" (orange), "are" (red), "there" (blue), "in" (purple), and "strawberry" (orange). This visualizes how LLMs like GPT-4o process text as discrete tokens rather than individual characters or words.

Fig 2: Tokens for GPT-4o of “How many R’s are there in strawberry” (Visualise the tokens yourself at: <https://gpt-tokenizer.dev/>)

As a single r happens very unfrequently compared to the entire word “strawberry” – hence, we map the word “strawberry” as a single token.

Well then – what does this mean?

It means that we have already fixated the abstraction space in which LLMs process data. It is now good at understanding words (and part of words – subwords), but it loses out on character level resolution.

BAM. Now give it a character counting task without giving it information about the characters. How do you think it would fare?

Of course, it doesn’t do the task very well 😊

How to do better for this task?

If we are interested in giving the LLM a character level task, we should give it the character level view.

The easiest way to go about this (without re-training the entire LLM or the tokenizer) is to simply put spaces in strawberry (s t r a w b e r r y). This makes the tokenizer tokenise single letters and the LLM can then “see” the individual characters!

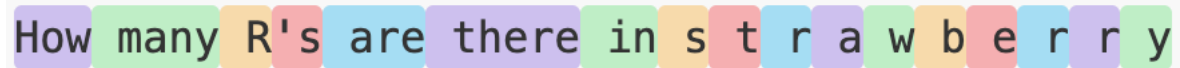
The image shows the sentence "How many R's are there in s t r a w b e r r y" where each character, including spaces, is enclosed in a separate colored box. The boxes are colored in a repeating pattern of purple, green, orange, red, blue, and light blue.

Fig 3: Tokens for GPT-4o of “How many R’s are there in s t r a w b e r r y” (Visualise the tokens yourself at: <https://gpt-tokenizer.dev/>)

Trying it out, we indeed get the right answer now!

ChatGPT 4o ▾

How many R's are there in s t r a w b e r r y



There are 3 R's in the word "strawberry."



Fig 4: Counting Rs in s t r a w b e r r y with GPT-4o

What else is difficult for this task?

It is well known that LLMs typically do not perform mathematical tasks well, as the way the characters are tokenized are not naturally reflect the mathematical vector space. In fact, if your training data contains a lot of “1+1=3”, there is a very high chance the LLM will output 3 when given 1+1.

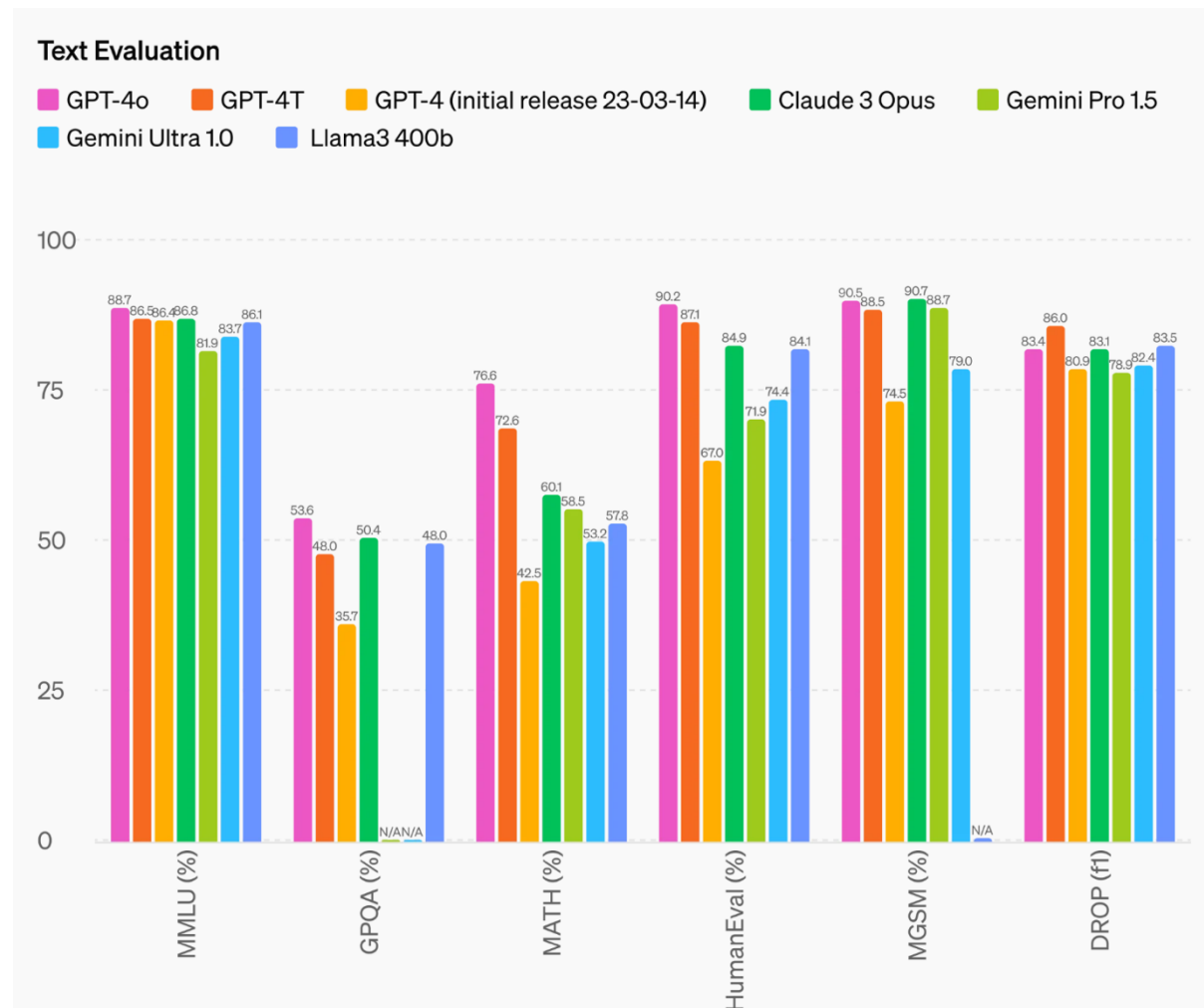


Fig 5: Various LLM performance on various benchmarks. As can be seen, the MATH benchmark is a tough one as the solve rate is still below 80% for competition mathematical problems. Image taken from <https://openai.com/index/hello-gpt-4o/>

Simply put, tokens are not exact points like how numbers are situated in the number line. Hence, counting numbers is a tough task for LLMs.

Moreover, the task of counting requires us to keep track of how many times a character has occurred, which is not innate to the LLM's abilities of pattern matching by vector similarity, and requires some form of memory.

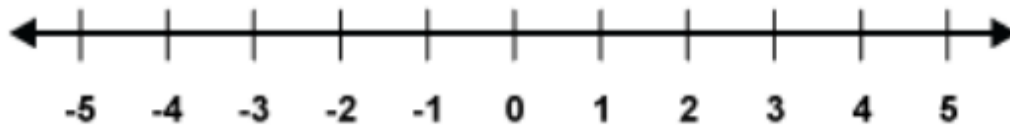


Fig 6: A number line. LLM tokens are not natively representative of the exact positioning in the number line

ChatGPT 4o ▾

What is 100^4 ?



100^4 is equal to $100 \times 100 \times 100 \times 100$.

To calculate it:

$$100^4 = 100 \times 100 \times 100 \times 100 = 10,000,000,000$$

So, $100^4 = 10,000,000,000$.

Fig 7: LLMs are not calculators. The answer of 100^4 is supposed to be 100,000,000

LLMs are not calculators – the pattern matching abilities they have do not translate well to exact mathematical calculations.

With in-built calculator tools, LLMs can approach exact calculation required for mathematics, but they should not be used to do mathematics directly.

The future of LLMs

LLMs are extremely powerful. They do have certain biases with their tokenization and the attention mechanism, but they have proven to do many arbitrary tasks well thanks to learning from web-scale data with next-token prediction as the objective.

LLMs should not be used for **ALL** tasks. For more logical and mathematical tasks, it is best off-loaded to exact systems to do it.

We should imbue LLMs with tools to do more tasks effectively, like how current LLM Agentic systems are doing it. Just a simple letter counting tool when taking a word as input would be able to solve the strawberry counting task almost perfectly every time, and with any word.

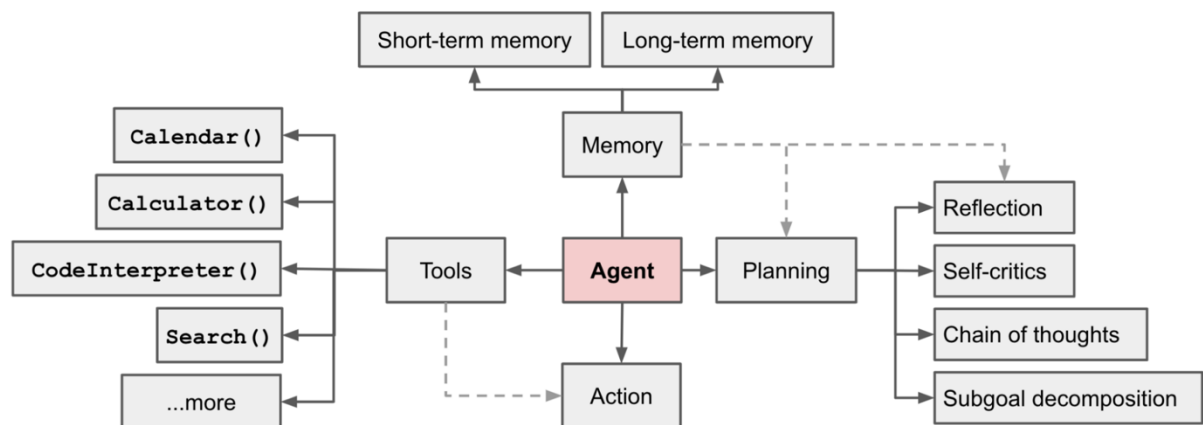


Fig 8: An overview of an LLM Agentic System. Taken from <https://lilianweng.github.io/posts/2023-06-23-agent/agent-overview.png>

We need more abstraction spaces to solve tasks. As can be seen from the letter counting task, a letter abstraction space would be helpful for that. There are also many different spaces that can be imbued, like sentence-level, summarization-level etc, which could improve performance for that specific task.

Storing information into memory when doing tasks, like storing the count when counting letters, could also be extremely helpful to boost the LLM's performance.

In between tasks, learning from the task and storing it into memory and using it again for future tasks could be very useful for continual learning and adaptation.

There are many more structures and tools that can be imbued with LLMs. The system as a whole will be more powerful than just what a single LLM can do.

So, when you hear people hyping up LLMs alone as being powerful, remember, an LLM is powerful on its own, but **even more powerful when integrated as a system**.



Fig 9: LLMs are stronger together in a system with tools, memory, various abstraction spaces etc. Image from <https://knowyourmeme.com/memes/apes-together-strong>