The first blog by thinking machines lab on defeating nondeterminism in LLM inference makes a good attempt to answer what i have always been thinking about. If you havent read it yet, here is the link 🔗 to the blog.

I am writing this exceprt to add my own thoughts and subtle explanations to the blog. Since I had to go to and fro from chatgpt and back to understand this blog, I thought I would convert the conversation into a blog post so its easier for others to understand. Please note that this is in no way a replacement for the original blog, but rather a complementary piece to it.

That being said, If you have read the blog and are looking for more insights or if you have been confused by some terms, please read on.

The blog starts with the problem almost all of us have faced at some point in time.
You take a large language model (LLM) and run inference on it multiple times with the same input prompt. You expect the output to be the same every time, but to your surprise, you get different outputs on different runs.(Even when keeping tempreature same)

> Note - Tempreature is a parameter that controls the randomness of predictions by scaling the logits before applying softmax. A higher temperature results in more random outputs, while a lower temperature makes the model's predictions more deterministic. Setting it to 0 is supposed to make the output deterministic.

For Reference, here are two experiments I did with Gemini 2.5 Flash lite side by side.

{% include figure.liquid loading="eager" path="assets/img/experiment1.png" class="img-fluid rounded z-depth-1" %}

Experiment 1 - Gemini 2.5 Flash Lite

{% include figure.liquid loading="eager" path="assets/img/experiment2.png" class="img-fluid rounded z-depth-1" %}

Experiment 2 - Gemini 2.5 Flash Lite

PROF

Even though I kept the prompt and temperature same, the outputs are different. (With varying tokens, type of response as well as wordings)
This is true even when use a Inference server like VLLM on a local LLM.

There can be several reasons for this behavior.
One simple but common reason is that combination of floating-point non-associativity and concurrent execution leads to nondeterminism based on which concurrent core finishes first. We will call this the “concurrency + floating point” hypothesis for LLM inference nondeterminism.

Mathematically it can be expressed as

$$(a + b) + c \neq a + (b + c)$$

An example of this can be produced using the following python code

```
x= 0.1 + (0.2 + 0.3)
y = (0.1 + 0.2) + 0.3

print(f"x value:{x}")
print(f"y value:{y}")
print(f"Difference:{x-y}")
```

```
x value:0.6
y value:0.6000000000000001
Difference:-1.1102230246251565e-16
```

This is a very small difference, but when you have millions of such operations happening in parallel, the differences can accumulate and lead to significant variations in the final output.
However this does not explain the Non determinism part fully. Say we run the following code on our computer, we do get deterministic outputs in this case