

# Even GPT-5.2 Can't Count to Five: The Case for Zero-Error Horizons in Trustworthy LLMs

Ryoma Sato

*National Institute of Informatics*

*rsato@nii.ac.jp*

## Abstract

We propose Zero-Error Horizon (ZEH) for trustworthy LLMs, which represents the maximum range that a model can solve without any errors. While ZEH itself is simple, we demonstrate that evaluating the ZEH of state-of-the-art LLMs yields abundant insights. For example, by evaluating the ZEH of GPT-5.2, we found that GPT-5.2 cannot even compute the parity of a short string like 11000, and GPT-5.2 cannot determine whether the parentheses in ((((( ))))) are balanced. This is surprising given the excellent capabilities of GPT-5.2. The fact that LLMs make mistakes on such simple problems serves as an important lesson when applying LLMs to safety-critical domains. By applying ZEH to Qwen2.5 and conducting detailed analysis, we found that while ZEH correlates with accuracy, the detailed behaviors differ, and ZEH provides clues about the emergence of algorithmic capabilities. Finally, while computing ZEH incurs significant computational cost, we discuss how to mitigate this cost by achieving up to one order of magnitude speedup using tree structures and online softmax.

## 1 Introduction

Large Language Models (LLMs) are being applied to an increasingly wide range of domains. In particular, their use in safety-critical domains such as healthcare, law, finance, and science is being actively explored [1, 11, 34, 54, 64, 66].

While LLMs appear extremely intelligent and capable of reasoning, they sometimes make mistakes that seem inconceivably foolish from a human perspective. For example, GPT-5.2 can implement complex fluid dynamics simulation code, yet it cannot even compute the parity of the short string 11000, cannot determine whether the parentheses in ((((( ))))) are balanced, and makes calculation errors on  $127 \times 82$  (Figure 1). This inconsistency poses a major obstacle when applying LLMs to safety-critical domains. What if an AI conducting large-scale financial transactions employs sophisticated financial theories but then makes a calculation error on  $127 \times 82$  and incurs massive losses? What if an AI managing a nuclear reactor system mistakenly believes that the status flag 11000 contains an odd number of 1s and opens the door of an operating reactor?

In addition to the API commands in Figure 1, GPT-5.2 appears to be particularly poor at handling the parenthesis string ((((( ))))). As of January 22, 2026, when queried through the web interface, even with chain-of-thought reasoning (GPT-5.2-Thinking), it often fails to correctly determine whether ((((( ))))) is balanced (Figure 2). While the web interface behavior is stochastic, after trying many patterns, it frequently answered that ((((( ))))) was balanced. We encourage readers to try this.

In this paper, we propose Zero-Error Horizon (ZEH) as a tool for clarifying such “holes” in AI capabilities. A model having  $ZEH = n$  for a given problem means that it can solve all instances up to size  $n$  without error, but makes at least one error on instances of size  $n + 1$ . For example, in computing the parity of binary strings, if we define problem size as string length, GPT-5.2 can accurately compute parity for strings up to 4 characters, but makes an error on 11000, so  $ZEH = 4$ . For multiplication  $a \times b$ , if we define problem size as the larger operand value  $\max(a, b)$ , GPT-5.2 correctly answers all multiplications where both operands

```

$ curl -s https://api.openai.com/v1/responses \
-H "Authorization: Bearer $OPENAI_API_KEY" \
-H "Content-Type: application/json" \
-d '{
    "model": "gpt-5.2-2025-12-11",
    "instructions": "Compute the parity (XOR) of the binary string. Answer with only 0 or 1.",
    "input": "11000",
    "temperature": 0
}' \
| jq -r '.output[0].content[0].text'
1

$ curl -s https://api.openai.com/v1/responses \
-H "Authorization: Bearer $OPENAI_API_KEY" \
-H "Content-Type: application/json" \
-d '{
    "model": "gpt-5.2-2025-12-11",
    "instructions": "Is the parentheses string balanced? Answer with only Yes or No.",
    "input": "((((()))))",
    "temperature": 0
}' \
| jq -r '.output[0].content[0].text'
Yes

$ curl -s https://api.openai.com/v1/responses \
-H "Authorization: Bearer $OPENAI_API_KEY" \
-H "Content-Type: application/json" \
-d '{
    "model": "gpt-5.2-2025-12-11",
    "instructions": "Answer with only the integer.",
    "input": "127*82",
    "temperature": 0
}' \
| jq -r '.output[0].content[0].text'
10314

```

Figure 1: GPT-5.2 cannot compute the parity of the short string 11000, cannot determine whether the parentheses in ((((( ))))) are balanced, and makes a calculation error on  $127 \times 82$ . These commands can be copied and pasted, and if you set your API key `$OPENAI_API_KEY`, you can try them right now. Please give it a try.

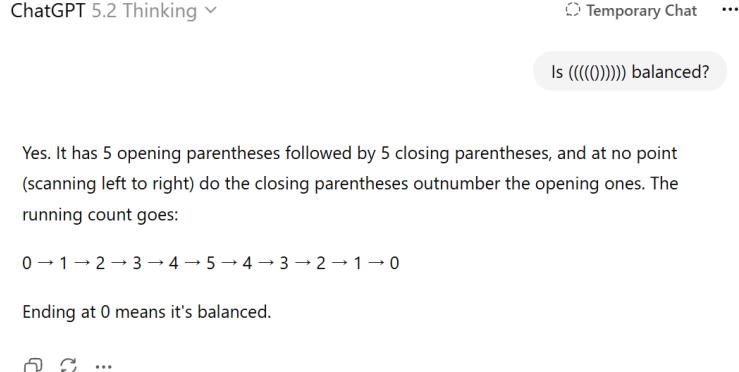


Figure 2: GPT-5.2-Thinking also fails to determine whether the parentheses in ((((( ))))) are balanced.

are at most 126, but makes an error on  $127 \times 82$ , so ZEH = 126. The definition of ZEH itself is not novel. However, we discovered that applying ZEH to LLMs automatically yields many surprising results.

There are numerous ways to use ZEH. First, ZEH provides a guideline for the problem sizes that can be safely entrusted to an LLM. Since ZEH = 126, we can say that GPT-5.2 can be trusted to some extent (though not with certainty, as we explain later) for mental arithmetic of two-digit multiplications. Conversely, one should be cautious about entrusting GPT-5.2 with tasks involving three-digit multiplications. While this might be acceptable for everyday shopping, in mission-critical tasks, the appearance of three-digit multiplication could serve as a warning signal worth considering. Similarly, with ZEH = 10 for balanced parentheses, GPT-5.2 might be trusted with nested structures of about 5 pairs, but one should be more cautious beyond that.

---

Additionally, we confirm that analyzing ZEH provides insights into the reasoning modes of LLMs.

Our main claim in this paper is that ZEH has numerous properties that accuracy lacks, and measuring ZEH is useful when investigating LLM performance. In particular, measuring ZEH is important when applying LLMs to safety-critical domains.

Code is available at <https://github.com/joisino/zeh>.

## 2 Zero-Error Horizon

### 2.1 Definition

We fix a model, problem, and prompt. For example, the model is GPT-5.2-2025-12-11, the problem is multiplication, and the prompt is `{"instructions": "Answer with only the integer.", "input": "{a}*{b}=?}`. Let  $B_n$  denote the set of problems of size  $n$ , and let  $T_n = \bigcup_{i=1}^n B_i$  denote the set of problems of size at most  $n$ .

**Example 1 (Multiplication).**

- $B_n = \{(a, b) \mid \max(a, b) = n\}$
- $T_n = \{(a, b) \mid a \leq n, b \leq n\}$

**Example 2 (Parity of binary strings).**

- $B_n = \{0, 1\}^n$
- $T_n = \{\text{all binary strings of length } \leq n\}$

**Example 3 (Balanced parentheses).**

- $B_n = \{"(\cdot, \cdot)"\}^n$
- $T_n = \{\text{all strings of } "(" \text{ and } ")" \text{ of length } \leq n\}$

**Example 4 (Chromatic number).**

- $B_n = \{\text{graphs with } n \text{ nodes}\}$
- $T_n = \{\text{graphs with at most } n \text{ nodes}\}$

Let  $C$  denote the set of problems that the fixed model, problem, and prompt answer correctly under deterministic greedy decoding. We define the **Zero-Error Horizon (ZEH)** as follows:

$$\text{ZEH} = \max\{n \mid T_n \subseteq C\} \tag{1}$$

That is, the model can solve all instances up to size ZEH without error, but makes at least one error on instances of size ZEH + 1. We call an instance of size ZEH + 1 on which the model makes an error a **ZEH limiter**.

### 2.2 Qualitative Characteristics of ZEH

#### 2.2.1 Safety Guarantees and Warning Signals

ZEH can be used for both safety arguments—“this size can be safely entrusted”—and danger arguments—“beyond this point is dangerous.” Unlike metrics such as accuracy that evaluate performance within a

---

Table 1: Prompt stability. Each column shows ZEH measured with different prompts. Low horizontal variation indicates robustness to prompts, and increasing trends in the vertical direction indicate that ZEH reflects model performance improvements.

Model	baseline	compute	product	eval	answer	Mean	Std
Qwen2.5-0.5B-Instruct	0	10	6	0	1	3.4	4.4
Qwen2.5-1.5B-Instruct	20	16	10	16	14	15.2	3.6
Qwen2.5-3B-Instruct	15	12	12	12	9	12.0	2.1
Qwen2.5-7B-Instruct	22	22	22	21	22	21.8	0.4
Qwen2.5-14B-Instruct	26	41	41	31	43	36.4	7.5
Qwen2.5-32B-Instruct	33	46	33	33	33	35.6	5.8
Qwen2.5-72B-Instruct	42	40	40	45	43	42.0	2.1

predetermined range, ZEH’s characteristic of evaluating the range itself works effectively. By determining the boundary, we can argue that inside the boundary is safe and outside is dangerous.

In safety-critical operations, even if average performance is high, holes in capability like careless mistakes can be fatal. Sampling problems may miss such holes, which is problematic in safety-critical domains. ZEH ensures no oversights through exhaustive verification.

It is important to note that problems within ZEH are not necessarily absolutely safe for the model. The definition of ZEH fixes the prompt (and context). For example, Qwen2.5-3B-Instruct has a ZEH of 15 on the multiplication task when using the prompt

- **Baseline:** `{"instructions": "Answer with only the integer.", "input": "{a}*{b}=?"}`

but may make errors on size 15 problems with other prompts or contexts. In fact, with the prompt

- **compute:** `{"instructions": "You are a calculator. Output only the result.", "input": "Compute a x b"}`

it makes errors on size 13 problems. However, as a rule of thumb, we can expect fairly high confidence of correct answers at or below ZEH. Table 1 shows the results of measuring ZEH for various Qwen2.5 models using different prompts. The prompts used are as follows:

- **product:** `{"instructions": "Give only the numerical answer.", "input": "What is the product of {a} and {b}?"}`
- **eval:** `{"instructions": "Evaluate and respond with just the number.", "input": "Evaluate: {a} * {b}"}`
- **answer:** `{"instructions": "Provide only the numerical answer.", "input": "{a} x {b} = ?"}`

The results in Table 1 show that while there is variation across prompts, the standard deviation of variation is approximately 3 on average, and the basic trends are consistent. For example, the 0.5B model cannot be trusted with multiplication. The 1.5B and 3B models can be trusted with single-digit multiplication. The 7B and 14B models are safe for multiplications up to 20, the 32B model up to 30, and the 72B model up to 40. For more safety-critical domains, it would be advisable to measure ZEH with multiple prompts and adopt the lowest value.

Conversely, we can say with certainty that caution should be exercised when applying to problems larger than ZEH—a guideline about danger. Since there actually exist examples where the model makes errors at  $ZEH + 1$  with a specific prompt, that domain is already not a completely safe zone. Qwen2.5-72B-Instruct is no longer absolutely safe for multiplications involving numbers greater than 42. One should be especially cautious in safety-critical domains.

---

ZEH on simple problems also provides implications for other tasks. In this paper, we focus on evaluating ZEH for simple problems such as multiplication, parity, and balanced parentheses. LLMs are rarely used for such simple problems directly. However, such tasks can serve as building blocks for complex problems. For example, when solving complex mathematical problems, multiplication may appear in intermediate calculations during chain-of-thought reasoning. For Qwen2.5-72B-Instruct, if this multiplication involves numbers greater than 42, the model may make an error in this step of reasoning, which can propagate and lead to errors in the final conclusion. Moreover, autoregressive LLMs are known to exhibit self-delusion, where once they output something incorrect, that error often adversely affects subsequent reasoning [38, 44]. In this way, even when multiplication is not the main objective, as long as multiplication appears in the process, the ZEH for multiplication serves as an effective warning signal.

## 2.2.2 Evidence via ZEH Limiters

One characteristic of ZEH is that ZEH limiters, which serve as concrete evidence for ZEH, are obtained. As shown in Figure 1, by presenting the results of ZEH limiters, anyone can be convincingly persuaded that GPT-5.2’s ZEH is at most 126. This is desirable mathematically, scientifically, and communicatively.

Additionally, ZEH limiters facilitate debugging. Validators may sometimes make misjudgments due to parsing errors, for example. With metrics like accuracy, one may not notice that some examples have such issues, causing the metric to deviate. With ZEH, by examining the ZEH limiter and the model’s output for that problem, one can verify that it is indeed a genuine error rather than a parsing mistake or misjudgment.

## 2.2.3 Analyzing ZEH Limiters is Beneficial

Analyzing how a model makes errors on ZEH limiters yields many insights. For example, the ZEH for multiplication of **Qwen2.5-0.5B-Instruct** is 0, and the ZEH limiter is  $1 \times 1$ . **Qwen2.5-0.5B-Instruct** answers 2 for this problem. It appears to be confusing multiplication with addition or not understanding the problem at all. The ZEH of **Qwen2.5-1.5B-Instruct** is 20, and the ZEH limiter is  $1 \times 21$ . **Qwen2.5-1.5B-Instruct** answers 42 for this problem. It may have confused this with  $2 \times 21$ . It seems unable to strictly distinguish between numbers. It also appears to be memorizing the multiplication table in this range rather than understanding the rules of calculation. The ZEH for multiplication of **Qwen2.5-32B-Instruct** is 33, and the ZEH limiter is  $34 \times 29$ . The correct answer to this problem is 986, but the model answers 1006. Compared to the errors made by the 0.5B and 1.5B models, this model’s error looks more reasonable. This multiplication is difficult for humans to compute mentally as well. Also, the ones digit is correct, and the difference from the correct answer is 20. This is the kind of error one might make when making a carry mistake in long multiplication, suggesting that the model understands the rules of multiplication but made an execution error. We discuss records showing that such patterns are widely observed in Section 3.2. Regarding GPT-5.2 as well, mistaking the parity of 11000 as 1, or perceiving the parenthesis string ((((( ))))) as balanced, is in some sense reasonable, serving as evidence that GPT-5.2 is performing rational reasoning. As Figure 2 shows, a ZEH limiter sometimes transfers to different prompts and contexts, indicating that the model has a fundamental weakness in that area.

## 2.2.4 Automatically Obtaining Surprising Results

Another benefit of evaluating ZEH is that surprising results are obtained automatically. The surprising results that GPT-5.2 cannot determine the parity of 11000 or whether ((((( ))))) is balanced are obtained as automatic byproducts when computing ZEH. In other words, ZEH alerts us that even extremely accurate models can fail on surprisingly small problem instances and are not completely safe. Since ZEH limiters are such smallest, most extreme examples, they provide maximum insight and surprise.

This characteristic is similar to adversarial examples, but they are not practically the same. Adversarial examples are unnatural, out-of-distribution (OOD) examples, so it is in some sense expected that models fail on them. ZEH limiters derive their practical significance and surprise from being natural, in-distribution (ID), small examples that could actually occur in reality, yet high-performance models still make mistakes on them.

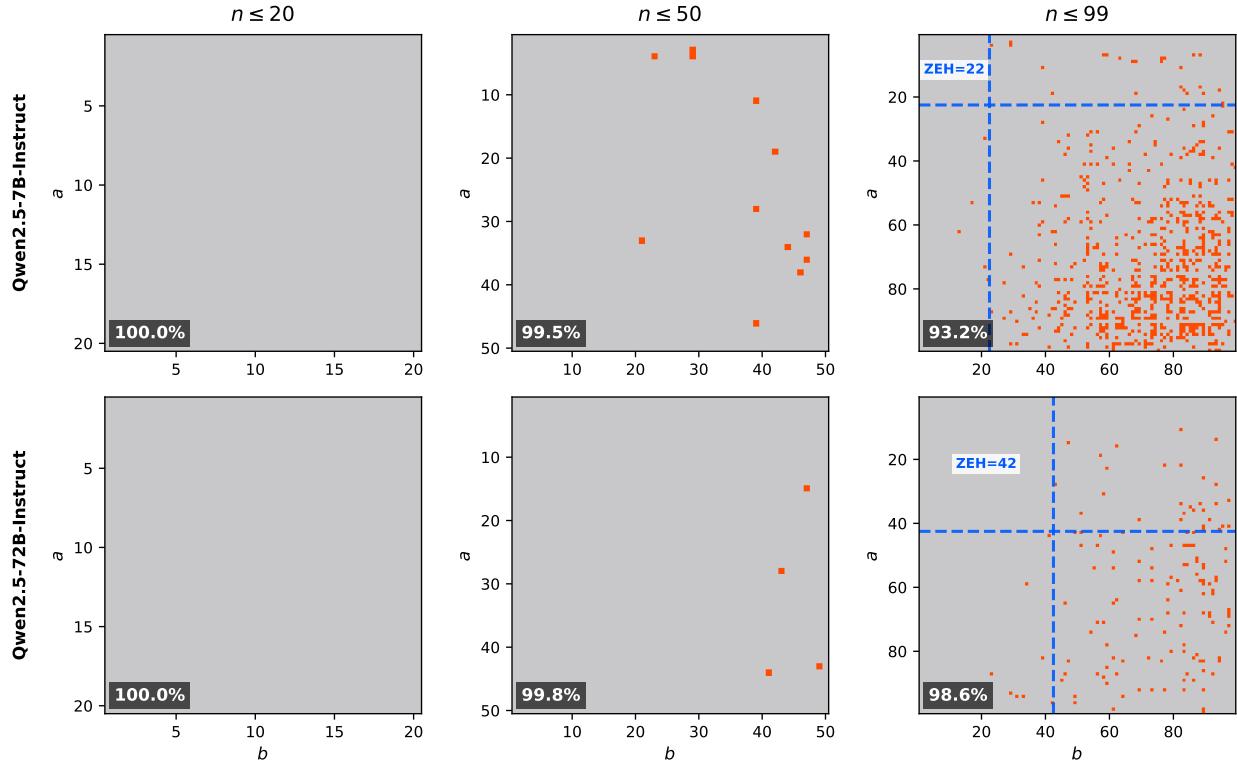


Figure 3: Comparison of ZEH and accuracy on the multiplication task for Qwen2.5. Red dots indicate errors. From left to right, accuracy is shown for ranges  $n \leq 20$ ,  $n \leq 50$ , and  $n \leq 99$ . The rightmost panel shows ZEH. Accuracy varies greatly depending on the choice of range, but ZEH does not require range selection, making it an objective metric.

### 2.2.5 Accuracy Has Scale Arbitrariness, but ZEH Does Not

To calculate accuracy, one must first determine the range of problems, but this range determination can be subject to the evaluator’s arbitrariness. On the other hand, ZEH has no scale arbitrariness. Figure 3 shows the multiplication performance of **Qwen2.5-7B-Instruct** and **Qwen2.5-72B-Instruct**. For example, a researcher proposing to compress the 72B model to the 7B model might set the benchmark range to  $n \leq 20$  or  $n \leq 50$ , show the left or middle panel of Figure 3, and claim: “Despite compressing from 72B to 7B by more than 10x, accuracy has hardly degraded.” Some readers might be deceived by this. However, when we expand the benchmark range to  $n \leq 99$ , we can see that the 7B model makes far more errors than the 72B model. In contrast, ZEH does not allow such range cherry-picking and can provide objective values of 22 vs 42.

This can become problematic even without intentionally selecting ranges to make a proposed method look good. Rather, the problem is often that people are unaware that accuracy has built-in biases regarding range. To measure accuracy, the evaluator must decide the range—for example, measuring accuracy for multiplications up to  $99 \times 99$ . At this point, the value 99 already incorporates human preconception. When the accuracy for  $99 \times 99$  is 99.8%, one might consider the accuracy sufficient for the multiplication task and not think deeply about it. But in reality, the ZEH might be 5, and the model might be making errors on single-digit multiplications. The range for ZEH is determined by the model, not by humans. By having the model determine the range, we can eliminate preconceptions and make objective judgments.

---

Table 2: Zero-Error Horizon (ZEH) results and first-failure examples for GPT-5.2 across tasks.

Task	ZEH	ZEH Limiter	Expected	GPT-5.2's Answer
Multiplication	126	$127 \times 82 =$	10414	10314
Parity	4	11000	0	1
Balanced Parentheses	10	((((()))))	No	Yes
Graph Coloring	4	$\{(1, 2), (1, 4), (1, 5), (2, 3)\}$	2	3

### 2.2.6 ZEH Does Not Become Obsolete as a Metric

Fixed-size benchmarks are destined to eventually become saturated as most problems get solved, approaching their limits. In the example of Figure 3, multiplications up to  $50 \times 50$  have saturated, making it impossible to clearly differentiate the performance of larger models. The same will eventually happen for  $99 \times 99$  as well. MNIST, CIFAR-10, and GLUE [48, 57] have all followed similar trajectories. They are no longer useful as benchmarks for state-of-the-art models. In contrast, ZEH is open-ended and continues to serve as a metric for model evolution. This is a benefit of ZEH not fixing the range in advance.

### 2.2.7 Instability

ZEH is more unstable compared to accuracy. Even if a base model has  $\text{ZEH} = 100$ , if a slight variant accidentally makes an error on just one problem at  $n = 10$ , that variant’s ZEH plummets to 9. In other words, this means it has high sensitivity, and when using it as an alarm to capture safety issues, higher sensitivity to failures is better. On the other hand, continuous metrics like accuracy are relatively stable but have low sensitivity, making them unsuitable as alarms.

### 2.2.8 Notes on Using External Tools

For the multiplication task, note that agents that make external calls to a calculator tool may have  $\text{ZEH} = \infty$ . While  $\text{ZEH} = \infty$  does have meaning, in this paper we primarily evaluate the LLM itself without external tool calls. The main reason is that we want to understand the capabilities of the LLM itself. Also, even when external calls are allowed, errors can occur in the judgment of whether to call or in integrating the results of the call, and error propagation can occur from there. Even when calls are permitted, for small number multiplications, the LLM may compute it itself without bothering to call the tool, and make errors there. For problems requiring long reasoning paths, there can be cases where the model deliberately chooses not to call the tool to save on the number of tool calls and context length. Moreover, Tool call results tend to be OOD with respect to the pretraining distribution, and sharp increases in LLM uncertainty (i.e., entropy) immediately after tool use have been observed [12]. Additionally, systems without tool calls are simpler and easier to maintain. In short, it is preferable to solve problems without tool calls when possible. For these reasons, there are sufficient use cases where LLMs solve problems without calling tools, even for formal tasks. In summary, this paper considers the case without tool calls for two purposes: understanding the characteristics of the LLM itself, and evaluating performance in such simple workflows.

## 3 Analysis

### 3.1 ZEH of GPT-5.2

Table 2 shows the ZEH of GPT-5.2 for various tasks. The tasks used are as follows:

- **Multiplication:** Multiplication of integers.
- **Parity:** Parity (even/odd count of 1s) of binary strings.
- **Balanced Parentheses:** Determining whether parentheses are balanced.
- **Graph Coloring:** Chromatic number of simple undirected graphs.

---

Table 3: Zero-Error Horizon (ZEH) and multiplication accuracy for Qwen2.5-Instruct models.

Model	ZEH	Accuracy ( $99 \times 99$ )
Qwen2.5-0.5B-Instruct	0	55.0%
Qwen2.5-1.5B-Instruct	20	75.9%
Qwen2.5-3B-Instruct	15	79.3%
Qwen2.5-7B-Instruct	22	93.2%
Qwen2.5-14B-Instruct	26	97.1%
Qwen2.5-32B-Instruct	33	98.6%
Qwen2.5-72B-Instruct	42	98.6%

Table 4: Spearman correlation between C4 corpus frequency statistics and accuracy. Small models have high correlation, indicating that they memorize training data. As model size increases, the correlation decreases, indicating a shift from memorization to algorithmic reasoning.

Model	Spearman's $\rho$
Qwen2.5-0.5B-Instruct	0.2651
Qwen2.5-1.5B-Instruct	0.2000
Qwen2.5-3B-Instruct	0.1872
Qwen2.5-7B-Instruct	0.1147
Qwen2.5-14B-Instruct	0.0683
Qwen2.5-32B-Instruct	0.0353
Qwen2.5-72B-Instruct	0.0374

As mentioned in Section 2.2.2, a characteristic of ZEH is that evidence showing ZEH is at most these values can be demonstrated. In fact, by running the commands in Figure 1, you can verify that GPT-5.2 makes errors on these examples, confirming that ZEH is at most these values. Please give it a try. Despite GPT-5.2 having the capability to write complex fluid dynamics simulation code, these examples show that GPT-5.2 makes errors on the 5-character parity of 11000, cannot determine whether the 11-character parenthesis string ((((( ))))) is balanced, and makes a mistake on the 3-digit multiplication  $127 \times 82$ . This vividly demonstrates holes in LLM capabilities.

### 3.2 Detailed Analysis of Qwen2.5

We analyze in detail the multiplication capability of Qwen2.5. This analysis suggests that Qwen2.5 acquires arithmetic rules as model size increases. ZEH is well-suited for capturing such emergence of capabilities.

As shown in Table 3, both accuracy and ZEH tend to increase as model size grows. However, the changes do not correspond one-to-one. Setting aside the 0.5B model with extremely low accuracy, ZEH stagnates from 1.5B to 7B, rises slightly at 14B, and then grows steadily at 32B and 72B. One reason for this is the transition in reasoning method from memorization to algorithms.

When a model relies on memorization, ZEH is difficult to increase. Whether each example is included in the training data is stochastic, and the locations of memorization are random. When errors occur randomly like holes, ZEH is difficult to increase. For example, even if accuracy is close to 99%, if the number of problems up to size 10,  $|T_{10}|$ , is 100, then on expectation there will be one mistake in  $T_{10}$ , so ZEH will be less than 10. Even if by luck all of  $T_{10}$  is answered correctly, it is still difficult for ZEH to exceed 10. Qwen is not quite so extreme—it has mostly mastered single-digit multiplication and almost mastered up to around 15. However, multiplications from around 15 to 25 largely rely on incomplete memorization, so even if accuracy increases substantially, ZEH does not increase much. On the other hand, when arithmetic is mastered, structure appears in the error patterns. Problems like  $34 \times 29$  that have many carries and are difficult for humans are still prone to errors even after mastering arithmetic, but such arithmetically difficult problems are concentrated at larger  $n$ . In other words, even if overall accuracy remains the same, when arithmetic is learned, errors concentrate on arithmetically difficult problem instances, and random holes become fewer.

Table 5: Total errors and structured errors. As model size increases, errors become increasingly structured.

Model	Total Errors	Structured Errors	Structured Rate
Qwen2.5-0.5B-Instruct	4412	2569	58%
Qwen2.5-1.5B-Instruct	2360	1838	78%
Qwen2.5-3B-Instruct	2032	1544	76%
Qwen2.5-7B-Instruct	668	562	84%
Qwen2.5-14B-Instruct	287	250	87%
Qwen2.5-32B-Instruct	140	121	86%
Qwen2.5-72B-Instruct	141	127	90%

Table 6: Logistic regression coefficients for predicting correctness based on carry presence and model size. The negative interaction coefficient indicates that larger models struggle more with carry problems.

	Carry Presence	Model Size ( $\log_{10}$ )	Interaction
Coefficient	-0.5456	1.7902	-0.3483

This is when ZEH can grow. And this ability to execute arithmetic without error grows with increasing parameters, enabling steady ZEH growth.

Below we refine this argument using data.

Table 4 shows Spearman’s correlation coefficient  $\rho$  between the number of times each instance  $a \times b$  appears in the C4 corpus and whether the model answers correctly. Small models have high correlation—that is, problems that appear in the corpus clearly have higher accuracy. This suggests memorization of training data. As model size increases, this dependence decreases. This is one piece of evidence for the transition from memorization to algorithms.

Table 5 shows the proportion of each model’s errors that are structured. Here, an error being structured means that

$$|\text{pred} - \text{gold}| \in \{10, 20, \dots, 100\}. \quad (2)$$

That is, the difference between the predicted value and the correct value is a multiple of 10 up to 100. For example, for the correct answer  $34 \times 29 = 986$ , if the model answers 1006, this error is structured because the difference between the predicted and correct values is 20. In this case, the model at least got the ones digit correct, serving as evidence that it understands the rule that the ones digit is computed by multiplying the ones digits of the operands. Making errors of exactly 10, 20, …, 100 is a near-miss that could also occur when humans do long multiplication, and a higher rate of this indicates indirect evidence of reasoning through a method similar to long multiplication internally. Table 5 shows that as models get larger, errors become remarkably more structured. This too serves as evidence that reasoning becomes more algorithmic as model size increases.

Next, we show that larger models struggle more with carries. An instance  $a \times b$  having a carry here means that for some digit  $d_a$  and  $d_b$ ,  $d_a \times d_b \geq 10$ . Using all correct/incorrect data from all models, we fit a logistic regression model to predict whether the model answers the instance correctly, using the following 3-dimensional feature vector  $\mathbf{x} \in \mathbb{R}^3$  (Table 6):

- $x_1 \in \{0, 1\}$ : Whether the instance has a carry
- $x_2 \in \mathbb{R}$ :  $\log_{10}$  (model size)
- $x_3 \in \mathbb{R}$ :  $x_1 \times x_2$

In this model, the coefficient of the interaction term  $x_3$  is  $-0.3483$ , and we reject at significance level 0.05 that the coefficient is 0. This means that the larger the model, the more pronounced the decrease in accuracy when there is a carry. What does this mean? The coefficient for model size  $x_2$  is  $1.7902 > 0$ . That is, the

---

**Algorithm 1:** Naive ZEH Evaluation Algorithm

---

**Input** : Model  $M$   
**Output:** Zero-Error Horizon (ZEH) value  
**for**  $n \leftarrow 1$  **to**  $\infty$  **do**  
     $B_n \leftarrow$  Generate all instances of size  $n$ ;  
    **foreach** instance  $t$  in  $B_n$  **do**  
         $pred \leftarrow M(t)$ ;  
        **if**  $pred$  is incorrect **then**  
            **return**  $n - 1$ ;

---

larger the model, the higher the base accuracy. However, within the model, there is a relative tendency to struggle more with carry problems. Small models are indifferent to whether there is a carry or not. After all, since small models rely on memorization, they answer correctly or incorrectly regardless of the arithmetic difficulty of the problem. On the other hand, the larger the model, the more it solves instances arithmetically, so the probability of calculation errors increases for arithmetically difficult tasks with many carries, and arithmetic difficulty comes to affect correctness. This too serves as evidence that as models get larger, they shift from memorization to rules.

Integrating the analysis so far, the Qwen2.5 family appears to be transitioning from memorization to algorithmic reasoning as model size increases. Since ZEH is difficult to increase through memorization, the fact that ZEH is increasing can serve as evidence that reasoning is being done through methods with algorithmic structure. In particular, the continuous growth of this value can serve as evidence that the precision of algorithm execution and operational ability is improving—that is, not only understanding the rules but being able to execute them reliably.

## 4 Speedup

Evaluating ZEH incurs computational cost because problems must be exhaustively evaluated. Algorithm 1 shows the naive pseudocode.

In this section, we propose methods to mitigate this problem through four speedups. For clarity, we focus on the multiplication task and assume each digit is one token.

### 4.1 From Autoregressive Decoding to Parallel Verification via Teacher Forcing

The most naive method decodes autoregressively exactly as in the definition. However,

$$\begin{array}{ll} 28 \times 43 = & \rightarrow 1 \\ 28 \times 43 = 1 & \rightarrow 2 \\ 28 \times 43 = 12 & \rightarrow 0 \\ 28 \times 43 = 120 & \rightarrow 4 \end{array}$$

as shown above, multiple inference passes are required for a single problem, making it inefficient. To solve this problem, parallel decoding via teacher forcing is effective. That is, we apply a causal mask and input the entire sequence  $28 \times 43 = 1204$  to the model at once, verifying in a single pass that the argmax of the next-token prediction after “=” is “1”, the argmax after “1” is “2”, the argmax after “2” is “0”, and the argmax after “0” is “4”.

However, teacher forcing cannot perform flexible decoding, which can be problematic. For example, even if the argmax of the next-token prediction after “1” is “,”, this cannot be called an error. With greedy decoding, the output might be “1,204”, which should be judged as correct. Also, the next token after “1” might be “20”. There are multiple token sequences representing the same number, but teacher forcing can only consider one of them. In other words, teacher forcing may make misjudgments. However, when teacher

Table 7: Verification runtime (ms) on the 1–99 multiplication suite (9801 tasks). Parentheses show speedup relative to TF. TF: Teacher Forcing. Trie (SDPA) uses the trie structure but uses standard attention and dense attention masks. Both Trie (SDPA) and FlashTree use teacher forcing and prompt prefilling as well.

Model	TF	TF + Prefill	Trie (SDPA)	FlashTree
Qwen2.5-0.5B-Instruct	2812 (1.00x)	1532 (1.84x)	1419 (1.98x)	<b>1037 (2.71x)</b>
Qwen2.5-1.5B-Instruct	7542 (1.00x)	3961 (1.90x)	3810 (1.98x)	<b>2845 (2.65x)</b>
Qwen2.5-3B-Instruct	14581 (1.00x)	7311 (1.99x)	6638 (2.20x)	<b>5001 (2.92x)</b>
Qwen2.5-7B-Instruct	29852 (1.00x)	15567 (1.92x)	13323 (2.24x)	<b>11165 (2.67x)</b>

Table 8: End-to-end runtime (ms) on computing ZEH. Parentheses show speedup relative to Naive Autoregression. Naive: Autoregressive Decoding. LA: Look Ahead. TF: Teacher Forcing. Both Trie (SDPA) and FlashTree use teacher forcing, look ahead, and prompt prefilling as well. TF methods include fallback to autoregressive decoding to avoid misjudgments as described in Section 4.1, and this fallback time is included in the measurements. Qwen2.5-3B-Instruct runs faster than Qwen2.5-1.5B-Instruct because its ZEH is smaller (Table 3), requiring fewer instances to evaluate.

Model	Naive	Naive + LA	TF	TF + LA	Trie (SDPA)	FlashTree
Qwen2.5-0.5B-Instruct	46 (1.00x)	330 (0.14x)	<b>29 (1.59x)</b>	214 (0.21x)	124 (0.37x)	126 (0.37x)
Qwen2.5-1.5B-Instruct	3135 (1.00x)	799 (3.92x)	1746 (1.80x)	647 (4.85x)	386 (8.12x)	<b>339 (9.25x)</b>
Qwen2.5-3B-Instruct	2657 (1.00x)	625 (4.25x)	1446 (1.84x)	467 (5.69x)	256 (10.38x)	<b>234 (11.35x)</b>
Qwen2.5-7B-Instruct	4515 (1.00x)	1825 (2.47x)	2984 (1.51x)	1661 (2.72x)	792 (5.70x)	<b>733 (6.16x)</b>

forcing judges an answer as correct, correctness can be guaranteed. Therefore, careful handling is necessary: judge with teacher forcing, examine the token where teacher forcing judged an error, determine it as an error if it is clearly wrong such as a different digit, and fall back to autoregressive decoding if it cannot be definitively called wrong at that point, such as with spaces or commas. As we will see later, even considering this, teacher forcing is often faster than autoregressive decoding.

## 4.2 Batching Across Sizes

The most naive method, as in Algorithm 1, divides instances by size and evaluates each size separately. However, when the number of instances of size  $n$  is small, GPU compute units may be underutilized. Therefore, it is effective to sort instances by size, batch them from smallest regardless of size for parallel verification, and if there is an error, take the smallest size as the ZEH. This maximizes utilization of GPU compute units. We call this look-ahead processing.

## 4.3 Prompt Cache Sharing

The beginning portions of strings that need to be evaluated in ZEH are shared. For example, the portion `{"instructions": "Answer with only the integer.", "input":` is common to all instances. By exploiting this, we can speed up by evaluating the common prompt portion only once and sharing its KV cache across all instances [21]. We call this prompt prefilling.

## 4.4 Tree Structure Sharing via FlashTree

Beyond the prompt, there are also tokens shared among some groups. For example, the instances

- $1 \times 1 =$
- $1 \times 2 =$
- $1 \times 3 =$

share the portion  $1\times$ . These KV caches can also be shared. In general, by managing the strings to be evaluated with a trie, we can share the KV cache for shared prefixes. This is the same idea as Tree Attention

---

in speculative decoding [5, 14, 39, 45, 53, 68]. However, in a naive implementation, an attention mask of size (#nodes)  $\times$  (#nodes) is required to control attention from children to parents, which can become a bottleneck. Based on the same idea as FlashAttention [8, 9, 49], we implemented a Triton kernel that does not explicitly create an attention mask, reads only the pairs needed for attention, and computes using online softmax. This maximizes the benefit of tree structures. We call this FlashTree.

Table 7 shows the verification time for each method, and Table 8 shows the end-to-end time to compute ZEH. Runtimes were measured on an RTX 4090 GPU machine. FlashTree is up to 3x faster than teacher forcing and up to 10x faster than naive autoregression. Note that the end-to-end time is longer for Qwen2.5-0.5B-Instruct because Qwen2.5-0.5B-Instruct has ZEH of 0, terminating after solving just one instance of size 1, making look-ahead counterproductive. A slight caveat is that while FlashTree is exact, the order of multiplication operations differs from naive methods, so due to floating-point properties, results may differ slightly. This characteristic is the same as FlashAttention. However, such instances are less than 1%, and the ZEH values themselves match the naive method. In safety-critical domains, care must be taken to ensure consistency of calculation between production and verification environments, including numerical errors.

These techniques are important for mitigating computational cost issues. However, these techniques are not a panacea, and when model performance improves, the explosion in the number of instances  $|T_n|$  that must be evaluated is unavoidable. Fundamentally solving this problem will likely require mathematical and formal methods. This direction is left for future work.

## 5 Related Work

### 5.1 Reliability Evaluation Metrics

The reliability of large language models is evaluated from multiple perspectives including factuality, groundedness, and handling of uncertainty. There are surveys and overviews providing recent systematization, organizing hallucinations and discussing their causes [20, 24, 25, 61]. Multiple benchmarks have been proposed [2, 4, 6, 35, 36, 41, 52, 60, 62, 63], including TruthfulQA [30] which induces plausible incorrect answers, and HalluEval [29] which focuses on hallucinations. To reduce the cost of human evaluation, automatic evaluators have also been developed [40, 63]. Methods for evaluating the reliability of each output include model self-confidence [22, 27, 37], output consistency [26, 58, 59], and uncertainty metrics [27]. Additionally, the accuracy of citations produced by LLMs is also an important issue [15]. While these focus on factuality, groundedness, and uncertainty for open-ended generation, Zero-Error Horizon in this paper provides a “size boundary where zero error is guaranteed” and concrete failure examples (ZEH limiters) for verifiable tasks, serving as a complementary evaluation axis for safety-critical operations.

### 5.2 LLMs and Arithmetic

There is extensive research on methods to train LLMs on arithmetic and methods to evaluate arithmetic capabilities. It is not trivial for LLMs to perform addition and multiplication, and various approaches have been proposed [3, 23, 28, 33, 43, 46, 51, 55]. For more advanced mathematical tasks, numerous benchmarks have been proposed [7, 13, 16, 17, 18, 56, 69], and improvements in LLM mathematical capabilities have been reported [10, 19, 31, 32, 47, 50, 59, 65, 67]. Our research is similar to Nikankin et al. [42] in that we investigate the capabilities of pre-trained LLMs on simple arithmetic tasks like multiplication, but while Nikankin et al. [42] focuses on heuristics within models, we focus on scaling within model families and the transition from memorization to algorithms.

## 6 Conclusion

We proposed ZEH, a metric representing the maximum range that a model can reliably solve. ZEH provides guarantees about LLM reliability and conversely serves as a signal for danger. ZEH limiters, which serve as evidence for ZEH, clarify the capability of high-performance LLMs to make mistakes on surprisingly simple examples. We also confirmed that ZEH is effective for capturing the qualitative evolution from memorization to algorithms. ZEH has many benefits that accuracy lacks and is effective for evaluating trustworthy LLMs.

---

## References

- [1] M. Aljohani, J. Hou, S. Kommu, and X. Wang. A comprehensive survey on the trustworthiness of large language models in healthcare. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 6720–6748. Association for Computational Linguistics, 2025.
- [2] J. Aula-Blasco, J. Falcão, S. Sotelo, S. Paniagua, A. Gonzalez-Agirre, and M. Villegas. Veritasqa: A truthfulness benchmark aimed at multilingual transferability. In *Proceedings of the 31st International Conference on Computational Linguistics, COLING*, pages 5463–5474. Association for Computational Linguistics, 2025.
- [3] T. Baeumel, J. van Genabith, and S. Ostermann. The lookahead limitation: Why multi-operand addition is hard for llms. In *Proceedings of the 8th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 250–262. Association for Computational Linguistics, 2025.
- [4] F. F. Bayat, L. Zhang, S. Munir, and L. Wang. Factbench: A dynamic benchmark for in-the-wild language model factuality evaluation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL*, pages 33090–33110. Association for Computational Linguistics, 2025.
- [5] T. Cai, Y. Li, Z. Geng, H. Peng, J. D. Lee, D. Chen, and T. Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Proceedings of the 41st International Conference on Machine Learning, ICML*. OpenReview.net, 2024.
- [6] S. Chen, Y. Zhao, J. Zhang, I. Chern, S. Gao, P. Liu, and J. He. FELM: benchmarking factuality evaluation of large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS*, 2023.
- [7] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems. *arXiv*, abs/2110.14168, 2021. URL <https://arxiv.org/abs/2110.14168>.
- [8] T. Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *Proceedings of the 12th International Conference on Learning Representations, ICLR*, 2024.
- [9] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS*, 2022.
- [10] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv*, abs/2501.12948, 2025. doi: 10.48550/ARXIV.2501.12948. URL <https://doi.org/10.48550/arXiv.2501.12948>.
- [11] F. Dehghani, R. Dehghani, Y. Naderzadeh Ardebili, and S. Rahnamayan. Large language models in legal systems: A survey. *Humanities and Social Sciences Communications*, 12(1):1977, 2025.
- [12] G. Dong, H. Mao, K. Ma, L. Bao, Y. Chen, Z. Wang, Z. Chen, J. Du, H. Wang, F. Zhang, G. Zhou, Y. Zhu, J. Wen, and Z. Dou. Agentic reinforced policy optimization. *arXiv*, abs/2507.19849, 2025. URL <https://doi.org/10.48550/arXiv.2507.19849>.
- [13] J. Fan, S. Martinson, E. Y. Wang, K. Hausknecht, J. Brenner, D. Liu, N. Peng, C. Wang, and M. P. Brenner. Hardmath: A benchmark dataset for challenging problems in applied mathematics. In *Proceedings of the 13th International Conference on Learning Representations, ICLR*, 2025.
- [14] Y. Fu, P. Bailis, I. Stoica, and H. Zhang. Break the sequential dependency of LLM inference using lookahead decoding. In *Proceedings of the 41st International Conference on Machine Learning, ICML*. OpenReview.net, 2024.
- [15] T. Gao, H. Yen, J. Yu, and D. Chen. Enabling large language models to generate text with citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 6465–6488. Association for Computational Linguistics, 2023.
- [16] A. Gulati, B. Miranda, E. Chen, E. Xia, K. Fronsdal, B. de Moraes Dumont, and S. Koyejo. Putnam-axiom: A functional & static benchmark for measuring higher level mathematical reasoning in llms. In *Proceedings of the 42nd International Conference on Machine Learning, ICML*, 2025.

- 
- [17] C. He, R. Luo, Y. Bai, S. Hu, Z. L. Thai, J. Shen, J. Hu, X. Han, Y. Huang, Y. Zhang, J. Liu, L. Qi, Z. Liu, and M. Sun. Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL*, pages 3828–3850, 2024.
- [18] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.
- [19] T. Hubert, R. Mehta, L. Sartran, M. Z. Horváth, G. Žužić, E. Wieser, A. Huang, J. Schrittwieser, Y. Schroecker, H. Masoom, et al. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, pages 1–3, 2025.
- [20] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. Bang, A. Madotto, and P. Fung. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12):248:1–248:38, 2023.
- [21] J. Juravsky, B. C. A. Brown, R. Ehrlich, D. Y. Fu, C. Ré, and A. Mirhoseini. Hydragen: High-throughput LLM inference with shared prefixes. *arXiv*, abs/2402.05099, 2024. URL <https://doi.org/10.48550/arXiv.2402.05099>.
- [22] S. Kadavath, T. Conerly, A. Askell, T. Henighan, D. Drain, E. Perez, N. Schiefer, Z. Hatfield-Dodds, N. DasSarma, E. Tran-Johnson, S. Johnston, S. E. Showk, A. Jones, N. Elhage, T. Hume, A. Chen, Y. Bai, S. Bowman, S. Fort, D. Ganguli, D. Hernandez, J. Jacobson, J. Kernion, S. Kravec, L. Lovitt, K. Ndousse, C. Olsson, S. Ringer, D. Amodei, T. Brown, J. Clark, N. Joseph, B. Mann, S. McCandlish, C. Olah, and J. Kaplan. Language models (mostly) know what they know. *arXiv*, abs/2207.05221, 2022. URL <https://doi.org/10.48550/arXiv.2207.05221>.
- [23] L. Kaiser and I. Sutskever. Neural gpus learn algorithms. In *Proceedings of the 4th International Conference on Learning Representations, ICLR*, 2016.
- [24] A. T. Kalai and S. S. Vempala. Calibrated language models must hallucinate. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC*, pages 160–171, 2024.
- [25] A. T. Kalai, O. Nachum, S. S. Vempala, and E. Zhang. Why language models hallucinate. *arXiv*, abs/2509.04664, 2025. URL <https://doi.org/10.48550/arXiv.2509.04664>.
- [26] A. Korikov, P. Du, S. Sanner, and N. Rekabsaz. Batched self-consistency improves LLM relevance assessment and ranking. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 32687–32703. Association for Computational Linguistics, 2025.
- [27] L. Kuhn, Y. Gal, and S. Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *Proceedings of the 11th International Conference on Learning Representations, ICLR 2023*, 2023.
- [28] N. Lee, K. Sreenivasan, J. D. Lee, K. Lee, and D. Papailiopoulos. Teaching arithmetic to small transformers. In *Proceedings of the 12th International Conference on Learning Representations, ICLR*, 2024.
- [29] J. Li, X. Cheng, X. Zhao, J. Nie, and J. Wen. Halueval: A large-scale hallucination evaluation benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 6449–6464. Association for Computational Linguistics, 2023.
- [30] S. Lin, J. Hilton, and O. Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL*, pages 3214–3252. Association for Computational Linguistics, 2022.
- [31] Y. Lin, S. Tang, B. Lyu, J. Wu, H. Lin, K. Yang, J. Li, M. Xia, D. Chen, S. Arora, and C. Jin. Goedel-prover: A frontier model for open-source automated theorem proving. *arXiv*, abs/2502.07640, 2025. URL <https://doi.org/10.48550/arXiv.2502.07640>.
- [32] Y. Lin, S. Tang, B. Lyu, Z. Yang, J. Chung, H. Zhao, L. Jiang, Y. Geng, J. Ge, J. Sun, J. Wu, J. Gesi, X. Lu, D. Acuna, K. Yang, H. Lin, Y. Choi, D. Chen, S. Arora, and C. Jin. Goedel-prover-v2: Scaling formal theorem proving with scaffolded data synthesis and self-correction. *arXiv*, abs/2508.03613, 2025. URL <https://doi.org/10.48550/arXiv.2508.03613>.

- 
- [33] A. Madsen and A. R. Johansen. Neural arithmetic units. In *Proceedings of the 8th International Conference on Learning Representations, ICLR*. OpenReview.net, 2020.
- [34] S. Mahdavi, J. Chen, P. K. Joshi, L. H. Guativa, and U. Singh. Integrating large language models in financial investments and market analysis: A survey. *arXiv*, abs/2507.01990, 2025. URL <https://doi.org/10.48550/arXiv.2507.01990>.
- [35] C. Malaviya, S. Lee, S. Chen, E. Sieber, M. Yatskar, and D. Roth. Expertqa: Expert-curated questions and attributed answers. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, NAACL, pages 3025–3045. Association for Computational Linguistics, 2024.
- [36] A. Mallen, A. Asai, V. Zhong, R. Das, D. Khashabi, and H. Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL, pages 9802–9822. Association for Computational Linguistics, 2023.
- [37] P. Manakul, A. Liusie, and M. J. F. Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 9004–9017. Association for Computational Linguistics, 2023.
- [38] R. T. McCoy, S. Yao, D. Friedman, M. Hardy, and T. L. Griffiths. Embers of autoregression: Understanding large language models through the problem they are trained to solve. *arXiv*, abs/2309.13638, 2023. URL <https://doi.org/10.48550/arXiv.2309.13638>.
- [39] X. Miao, G. Oliaro, Z. Zhang, X. Cheng, Z. Wang, Z. Zhang, R. Y. Y. Wong, A. Zhu, L. Yang, X. Shi, C. Shi, Z. Chen, D. Arfeen, R. Abhyankar, and Z. Jia. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS*, pages 932–949. ACM, 2024.
- [40] S. Min, K. Krishna, X. Lyu, M. Lewis, W. Yih, P. W. Koh, M. Iyyer, L. Zettlemoyer, and H. Hajishirzi. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 12076–12100. Association for Computational Linguistics, 2023.
- [41] D. Muhlgay, O. Ram, I. Magar, Y. Levine, N. Ratner, Y. Belinkov, O. Abend, K. Leyton-Brown, A. Shashua, and Y. Shoham. Generating benchmarks for factuality evaluation of language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics, EACL*, pages 49–66. Association for Computational Linguistics, 2024.
- [42] Y. Nikankin, A. Reusch, A. Mueller, and Y. Belinkov. Arithmetic without algorithms: Language models solve math with a bag of heuristics. In *Proceedings of the 13th International Conference on Learning Representations, ICLR*, 2025.
- [43] R. Nogueira, Z. Jiang, and J. Lin. Investigating the limitations of the transformers with simple arithmetic tasks. *arXiv*, abs/2102.13019, 2021. URL <https://arxiv.org/abs/2102.13019>.
- [44] P. A. Ortega, M. Kunesch, G. Delétang, T. Genewein, J. Grau-Moya, J. Veness, J. Buchli, J. Degrave, B. Piot, J. Pérolat, T. Everitt, C. Tallec, E. Parisotto, T. Erez, Y. Chen, S. E. Reed, M. Hutter, N. de Freitas, and S. Legg. Shaking the foundations: delusions in sequence models for interaction and control. *arXiv*, abs/2110.10819, 2021. URL <https://arxiv.org/abs/2110.10819>.
- [45] Z. Pan, Y. Ding, Y. Guan, Z. Wang, Z. Yu, X. Tang, Y. Wang, and Y. Ding. Fasttree: Optimizing attention kernel and runtime for tree-structured LLM inference. In *Proceedings of the 8th Conference on Machine Learning and Systems, MLSys*, 2025. URL <https://openreview.net/forum?id=BwvHcHZ3kJ>.
- [46] A. Patel, S. Bhattacharya, and N. Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 2080–2094. Association for Computational Linguistics, 2021.

- 
- [47] Z. Z. Ren, Z. Shao, J. Song, H. Xin, H. Wang, W. Zhao, L. Zhang, Z. Fu, Q. Zhu, D. Yang, Z. F. Wu, Z. Gou, S. Ma, H. Tang, Y. Liu, W. Gao, D. Guo, and C. Ruan. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv*, abs/2504.21801, 2025. URL <https://doi.org/10.48550/arXiv.2504.21801>.
- [48] P. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 4937–4946. Computer Vision Foundation / IEEE, 2020.
- [49] J. Shah, G. Bikshandi, Y. Zhang, V. Thakkar, P. Ramani, and T. Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS*, 2024.
- [50] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, M. Zhang, Y. K. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv*, abs/2402.03300, 2024. URL <https://doi.org/10.48550/arXiv.2402.03300>.
- [51] J. A. Sivakumar and N. S. Moosavi. How to leverage digit embeddings to represent numbers? In *Proceedings of the 31st International Conference on Computational Linguistics, COLING*, pages 7685–7697, 2025.
- [52] Y. Song, Y. Kim, and M. Iyyer. Veriscore: Evaluating the factuality of verifiable claims in long-form text generation. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 9447–9474. Association for Computational Linguistics, 2024.
- [53] B. Spector and C. Ré. Accelerating LLM inference with staged speculative decoding. *arXiv*, abs/2308.04623, 2023. URL <https://doi.org/10.48550/arXiv.2308.04623>.
- [54] Z. Tang, H. E, Z. Ma, H. He, J. Liu, Z. Yang, Z. Rong, R. Li, K. Ji, Q. Huang, X. Hu, Y. Liu, and Q. Zheng. Financereasoning: Benchmarking financial numerical reasoning more credible, comprehensive and challenging. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL*, pages 15721–15749. Association for Computational Linguistics, 2025.
- [55] A. Trask, F. Hill, S. E. Reed, J. W. Rae, C. Dyer, and P. Blunsom. Neural arithmetic logic units. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS*, pages 8046–8055, 2018.
- [56] G. Tsoukalas, J. Lee, J. Jennings, J. Xin, M. Ding, M. Jennings, A. Thakur, and S. Chaudhuri. Putnambench: Evaluating neural theorem-provers on the putnam mathematical competition. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS*, 2024.
- [57] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 7th International Conference on Learning Representations, ICLR*, 2019.
- [58] W. Wang, Y. Wang, and H. Huang. Ranked voting based self-consistency of large language models. In *Findings of the Association for Computational Linguistics, ACL*, pages 14410–14426. Association for Computational Linguistics, 2025.
- [59] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of the 11th International Conference on Learning Representations, ICLR*, 2023.
- [60] Y. Wang, R. G. Reddy, Z. M. Mujahid, A. Arora, A. Rubashevskii, J. Geng, O. M. Afzal, L. Pan, N. Borenstein, A. Pillai, I. Augenstein, I. Gurevych, and P. Nakov. Factcheck-bench: Fine-grained evaluation benchmark for automatic fact-checkers. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 14199–14230. Association for Computational Linguistics, 2024.
- [61] Y. Wang, M. Wang, M. A. Manzoor, F. Liu, G. N. Georgiev, R. J. Das, and P. Nakov. Factuality of large language models: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 19519–19529. Association for Computational Linguistics, 2024.

- 
- [62] J. Wei, N. Karina, H. W. Chung, Y. J. Jiao, S. Papay, A. Glaese, J. Schulman, and W. Fedus. Measuring short-form factuality in large language models. *arXiv*, abs/2411.04368, 2024. URL <https://doi.org/10.48550/arXiv.2411.04368>.
  - [63] J. Wei, C. Yang, X. Song, Y. Lu, N. Hu, J. Huang, D. Tran, D. Peng, R. Liu, D. Huang, C. Du, and Q. V. Le. Long-form factuality in large language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS*, 2024.
  - [64] J. Wei, Y. Yang, X. Zhang, Y. Chen, X. Zhuang, Z. Gao, D. Zhou, G. Wang, Z. Gao, J. Cao, Z. Qiu, X. He, Q. Zhang, C. You, S. Zheng, N. Ding, W. Ouyang, N. Dong, Y. Cheng, S. Sun, L. Bai, and B. Zhou. From AI for science to agentic science: A survey on autonomous scientific discovery. *arXiv*, abs/2508.14111, 2025. URL <https://doi.org/10.48550/arXiv.2508.14111>.
  - [65] H. Xin, D. Guo, Z. Shao, Z. Ren, Q. Zhu, B. Liu, C. Ruan, W. Li, and X. Liang. Deepseek-prover: Advancing theorem proving in llms through large-scale synthetic data. *arXiv*, abs/2405.14333, 2024. URL <https://doi.org/10.48550/arXiv.2405.14333>.
  - [66] Y. Yamada, R. T. Lange, C. Lu, S. Hu, C. Lu, J. N. Foerster, J. Clune, and D. Ha. The AI scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv*, abs/2504.08066, 2025. URL <https://doi.org/10.48550/arXiv.2504.08066>.
  - [67] A. Yang, B. Zhang, B. Hui, B. Gao, B. Yu, C. Li, D. Liu, J. Tu, J. Zhou, J. Lin, K. Lu, M. Xue, R. Lin, T. Liu, X. Ren, and Z. Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv*, abs/2409.12122, 2024. URL <https://doi.org/10.48550/arXiv.2409.12122>.
  - [68] J. Yao, K. Chen, K. Zhang, J. You, B. Yuan, Z. Wang, and T. Lin. Deft: Decoding with flash tree-attention for efficient tree-structured LLM inference. In *Proceedings of the 13th International Conference on Learning Representations, ICLR*, 2025.
  - [69] Y. Zhao, Y. Long, H. Liu, R. Kamoi, L. Nan, L. Chen, Y. Liu, X. Tang, R. Zhang, and A. Cohan. Docmath-eval: Evaluating math reasoning capabilities of llms in understanding financial documents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL*, pages 16103–16120, 2024.