

Evaluating LLMs’ Ability to Understand Numerical Time Series for Text Generation

Mizuki Arai^{1,2}, Tatsuya Ishigaki², Masayuki Kawarada^{2,4},
Yusuke Miyao^{3,2}, Hiroya Takamura², Ichiro Kobayashi^{1,2}

¹Ochanomizu University, ²National Institute of Advanced Industrial Science and Technology (AIST),

³The University of Tokyo, ⁴CyberAgent, Inc., Japan

{g2120503, koba}@is.ocha.ac.jp {ishigaki.tatsuya, takamura.hiroya}@aist.go.jp
kawarada_masayuki@cyberagent.co.jp yusuke@is.s.u-tokyo.ac.jp

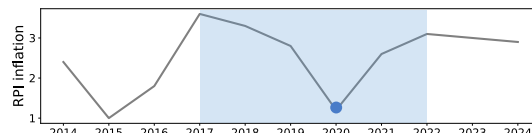
Abstract

Data-to-text generation tasks often involve processing numerical time-series as input such as financial statistics or meteorological data. Although large language models (LLMs) are a powerful approach to data-to-text, we still lack a comprehensive understanding of how well they actually understand time-series data. We therefore introduce a benchmark with 18 evaluation tasks to assess LLMs’ abilities of interpreting numerical time-series, which are categorized into: 1) event detection—identifying maxima and minima; 2) computation—averaging and summation; 3) pairwise comparison—comparing values over time; and 4) inference—imputation and forecasting. Our experiments reveal five key findings: 1) even state-of-the-art LLMs struggle with complex multi-step reasoning; 2) tasks that require extracting values or performing computations within a specified range of the time-series significantly reduce accuracy; 3) instruction tuning offers inconsistent improvements for numerical interpretation; 4) reasoning-based models outperform standard LLMs in complex numerical tasks; and 5) LLMs perform interpolation better than forecasting. These results establish a clear baseline and serve as a wake-up call for anyone aiming to blend fluent language with trustworthy numeric precision in time-series scenarios.

1 Introduction

The application of large language models (LLMs) has expanded from tasks with natural language inputs, such as question answering and summarization, to tasks involving structured data inputs, referred to as data-to-text tasks. Within data-to-text, numerical time series generation, such as various sensor data or stock market prices, hold significant potential for practical applications like financial reporting (Murakami et al., 2017; Hamazono et al., 2020; Kantharaj et al., 2022) and automated com-

Chat-to-Text



Inflation is an important measure of any country’s economy, and the Retail Price Index (RPI) is one of the most widely used indicators in the United Kingdom, with the rate expected to fall to 1.2 percent in 2020. The forecasted inflation rate for this index is estimated to increase in upcoming years, to 3.1 percent by 2022.

Benchmarks for Understanding Numerical Time Series

What is the minimum value from 2017 to 2022?
Time series:
{'2014': 2.4, '2015': 1.0, '2016': 1.8, '2017': 3.6, '2018': 3.3, '2019': 2.8, '2020': 1.2, '2021': 2.6, '2022': 3.1, '2023': 3.0, '2024': 2.9}



LLM

Minimum value is 1.2



Figure 1: The figure above shows an example of a text generation task using numerical time series data from the chart-to-text dataset. The bottom part displays an example from our constructed “Benchmarks for Understanding Numerical Time Series.” This benchmark is designed to assess the numerical comprehension capabilities required for downstream tasks.

mentary generation for games. Accurate text generation over time-series demands two key capabilities: 1) numerical interpretation—understanding numerical sequence; and 2) generation—producing a fluent text based on the understood numerical sequence.

However, most prior research focused on evaluating only the final generated text, indirectly assessing numerical understanding without directly verifying if LLMs accurately comprehend numerical sequences. Additionally, existing benchmarks for assessing numerical understanding in LLMs are primarily centered around numerical computations (Li et al., 2024a; Liu et al., 2024a), leaving a gap in research specifically focused on evaluating the ability of LLMs to comprehend numerical sequences for text generation tasks.

To systematically evaluate these capabilities, we introduce a benchmark with 18 evaluation tasks designed to assess LLMs’ ability to interpret numerical time-series data. These tasks are categorized into: 1) event detection, such as detecting maxima and minima; 2) computation, such as averaging and summation; 3) pairwise comparison, such as evaluating numerical relationships across time; and 4) inference, such as imputation and forecasting. Because each answer is a plain number, every slip is instantly visible, offering a direct read on numeric fidelity in generated text.

Through experiments, we reveal five key findings: 1) even state-of-the-art LLMs struggle with multi-step numerical reasoning; 2) performance significantly declines even with simple range constraints, highlighting the inherent difficulty of multi-step reasoning in time-series tasks; 3) instruction tuning does not consistently improve numerical interpretation; 4) certain reasoning-based models outperform standard LLMs in complex numerical tasks; and 5) applying a common linear methodology reveals a notable divergence in accuracy, particularly between imputation and forecasting tasks, indicating that inherent task differences significantly influence LLM performance despite employing identical analytical approaches.

2 Related Work

Interpreting sequences of numbers is a basic step for any model that needs to generate reliable text about data. A recent survey highlights the growing use of time-series inputs in large language models (Zhang et al., 2024). Real applications already depend on this skill, for example automated financial reports (Murakami et al., 2017; Hamazono et al., 2020), chart-to-text captioning (Kantharaj et al., 2022), and systems that produce natural-language forecasts (Jin et al., 2024).

The numerical understanding ability of LLMs has attracted significant attention from researchers, leading to the release of benchmark datasets for mathematical problems (Liu et al., 2024b; Li et al., 2024b; Ahn et al., 2024; Collins et al., 2023; Lu et al., 2023), reasoning with tabular data (Akhtar et al., 2023), and handling numerical information in free text (Chen et al., 2024). However, these datasets do not examine whether models can transform raw numbers into coherent natural-language prose, especially for time-series inputs—an ability our benchmark is designed to measure.

While existing time-series forecasting tasks, such as those introduced by Jin et al. (2024), could be used as one of benchmarks for time-series understanding to some extent, they fundamentally differ from our approach because of three reasons: 1) the time-series forecasting tasks lack of a definitive answer – forecasting tasks require predicting future values, which inherently allows for multiple plausible outcomes, and even among humans, predictions can vary, 2) dependence on inference – many existing tasks require extrapolation or assumptions beyond the given data, and 3) requirement for domain expertise – some tasks, such as predicting market prices in finance (Jin et al., 2024), rely heavily on domain knowledge.

A very recent work by Fons et al. (2024) also proposes a benchmark for evaluating LLMs on numerical time-series understanding. They primarily use synthetic data for tasks that often integrate numerical series with textual information or analyze multiple series, rather than focusing solely on the interpretation of individual numerical sequences. In contrast, our benchmark provides a more foundational and comprehensive assessment focused on single numerical sequences, quantitatively measuring LLMs’ core numerical comprehension across diverse task types.

3 Benchmark

3.1 Evaluation Tasks

Our benchmark aims to evaluate a model’s ability to understand numerical properties directly from time-series data without complex inference or domain-specific knowledge. We define this ability as the capability to obtain well-defined numerical properties—such as maximum values, means, and the time points at which extrema occur—directly from time-series data, without relying on domain-specific knowledge or external inference.

Specifically, we propose a set of 18 evaluation tasks, which are detailed along with their prompts in Table 1. These tasks are categorized into four categories: 1) event detection, 2) computation, 3) pairwise comparison, and 4) inference. Numerical values obtained by solving these tasks are often mentioned in text of data-to-text generation. Therefore, these tasks would measure LLMs’ ability to interpret numerical time-series that is required for data-to-text generation. Each instance of the tasks has a uniquely determined correct answer, ensuring that interpretation remains objective and aligned

Category	Task	Task Instructions	Exp. Answer
Event Detection	Max	Which is the value that is the maximum?	8.9
	Min	Which is the value that is the minimum?	0.014
	Max w/ range	Which is the value that is the maximum from 2008 to 2010?	213
	Min w/ range	Which is the value that is the minimum from 2008 to 2010 ?	208
	Maxtime	Which is the year corresponding to the maximum value?	2003
	Mintime	Which is the year corresponding to the minimum value?	1990
	Peak point	Which is the peak value in the following time series data? Multiple peak values may exist. If there are no peaks, please respond with None.	2756.67
	Dip point	Which is the dip value in the following time series data? Multiple peak values may exist. If there are no peaks, please respond with None.	1005.42, 1002.25
Computation	Exceed thresh.	Which year does the value becomes bigger than 0.9?	2003, 2006
	Below thresh.	Which year does the value becomes smaller than 0.9?	2010
	Average	Calculate the average value of the following time series data.	2006.45
	Average w/ range	Calculate the average value of the range from 2001 to 2005.	205.45
Pairwise Comparison	Sum	Calculate the sum of the following time series data.	20006.456
	Sum w/ range	Calculate the sum of the values in the range from 2004 to 2009.	10800.594
Inference	Difference	Calculate the absolute difference between the values for 2004 and 2009.	60.1
	Magnitude Comparison	Compare the values of the years 2008 and 2010 in the given time series. Provide the appropriate symbol to fill in the parentheses: '>' or '<' or '='.	>
Inference	Imputation	Linearly interpolate the 'NaN' value using the data points immediately before and after it.	101
	Forecasting	Predict the value for the next chronological period using linear regression based on the provided data.	12000

Table 1: The task instructions in prompts and examples of expected answers.

with human judgment.

3.1.1 Event Detection (10 tasks)

This category assesses LLMs' ability to identify significant numerical events. In downstream applications like language generation from numerical time-series data (Kantharaj et al., 2022), such numerical values in the time series need to be recognized and mentioned in the textual output. This category contains the following 10 tasks.

Maximum/minimum value: This is a task to find the maximum or minimum values in the given time series. This task would be essential for understanding the scale of the data.

Maximum/minimum value within a specified range: This is a task to find the maximum or minimum value within a given range of a given time series.

Time with maximum/minimum value: This is a task to identify the time points at which the maximum (or minimum) value occurs in a given time series. This task measures the model's ability to track temporal variations and recognize when significant events take place.

Peak and dip points: This is a task to detect peak and dip points in numerical time series data. A peak point refers to a local maximum compared to surrounding values, while a dip point refers to a local minimum.

Points exceeding or below a threshold: This task is to identify points that exceed or fall below a predefined threshold in a given time series.

3.1.2 Computation (4 tasks)

This category evaluates the ability to perform fundamental arithmetic operations on numerical time series data. While the arithmetic ability of LLMs for solving math word problems have been studied in previous research, we specifically design basic arithmetic tasks for understanding numerical time series data.

Average: This is a task to compute the mean of a given numerical time series.

Average within a specified range: This is a task to compute the mean within a specific segment of a given numerical time series. By calculating the mean over a limited period, we assess the model's ability to capture local trends accurately.

Sum: This is a task to compute the sum of all values in a given numerical time series.

Sum within a specified range: This is a task to compute the sum within a specific period in a given numerical time series. This task assesses the ability to correctly identify and process local values for summation.

3.1.3 Pairwise Comparison (2 tasks)

This category examines the ability to compare values at different time points. In datasets used for language generation from numerical time series (Kantharaj et al., 2022), reference texts often contain comparisons explicitly or implicitly, such as in “*The population of Europe as of 2020 was estimated to be 743 million, an increase of three million when compared with 2015*”. To choose a word *decrease*, LLMs need to have the ability to recognize the larger of the two given values.

Difference: This is a task to compute the difference between two values at different time points in a given numerical time series.

Magnitude comparison: This is a task to compare the magnitudes of values at two different time points in a given numerical time series, i.e., to recognize the larger of the two values.

3.1.4 Inference tasks (2 tasks)

This category evaluates the model’s ability to infer missing and future values from time-series data. For clarity and simplicity, we adopt two tasks—imputation to fill missing data by linear interpolation, and forecasting to predict future values by linear regression. By specifying a method for imputation or forecasting as linear regression, we make our metric immune to the choice of a method and attempt to focus on the core ability of time-series reasoning.

Imputation: This task assesses the ability to estimate and fill in missing data points within a given numerical time-series using linear interpolation.

Forecasting: This task requires the prediction of subsequent values in a given numerical time-series using linear regression on historical data.

3.2 Benchmark Dataset

We construct our benchmark dataset based on Chart-to-Text dataset (Kantharaj et al., 2022). Although each instance in Chart-to-Text is a pair of a chart and a text, we use only charts, because we are not going to work on a text generation task. We obtain the preprocessed version of this data pro-

vided by Kawarada et al. (2024b), which contains only line graphs. This dataset consists of 2,360 numerical time series on topics such as crime rates, mortality rates, and national debt, enabling evaluation across diverse real-world contexts.

For each instance in the dataset, we calculated the gold-standard answers to the tasks described in Section 3.1 using Python scripts.¹ Although those are basic Python scripts such as calculating the average given a list of numbers, we sampled 180 instances (10 per task) and confirmed that their answers are correct.

For tasks requiring specified ranges, such as “maximum values within a specified range”, we randomly select two time points (e.g., 2010 and 2017) and generate the gold answer accordingly. A similar automatic annotation process is applied to “points exceeding or below a threshold” tasks, where a threshold value is randomly selected within the given numerical range. For “difference” and “magnitude comparison” tasks, two time points for comparison are randomly selected. Similarly, for imputation, missing values are introduced at random positions for the model to predict. We also manually validated randomly selected 100 instances and found no outliers, noise, or irregularities in the underlying series. This check was performed independently of the 180-instance validation used to confirm the accuracy of our gold-answer scripts.

4 Experiments

4.1 Prompts

Prompts used to compare LLMs typically begin with an instruction² (e.g., “*Which is the maximum value?*”). All task instructions are listed in Table 1.

Our preliminary experiments revealed significant variations in the output format, making consistent comparisons challenging. Specifically, some LLMs provided lengthy explanatory responses, while others output mathematical formulas instead of a direct numerical answer. To address this issue, we employed few-shot learning and additional instructions to enforce a standardized output format.

For few-shot learning, we used five-shot prompting, as performance gains plateaued beyond five

¹The python script and the gold annotations will be made publicly available at <https://github.com/chiwacco/numerical-time-series-understanding.git>.

²We also evaluate Japanese translations. In our experiments, accuracy is higher when the prompt language matches each model’s pretraining (Fig. 11, Appendix F).

examples, consistent with findings from Kawarada et al. (2024a). Despite this, some models still failed to adhere to the required format, particularly in computation tasks, where they either generated formulas or long descriptive sentences instead of directly outputting numerical values. To mitigate this, we explicitly instructed models to provide only numerical answers, preventing the generation of intermediate steps or explanations. In practice, we experimented with several different prompt formulations and selected the one that yielded the highest accuracy and the most stable performance across all tasks and models (Appendix B). The proportion of outputs with format errors is reported in Appendix C.

Numerical time-series are represented as a JSON (Kawarada et al., 2024b) format and written in the prompt. In this structure, the keys in the JSON format represent time, i.e., year in our dataset, and values are the corresponding data. Appendix A provides details, including JSON structure examples, specific instruction phrasing for numerical-only outputs, and the integration of few-shot examples.

4.2 Evaluation Metrics

Our primary evaluation metrics are **Accuracy** and **F1 score**, chosen to suit the characteristics of each task. **Accuracy** is used for tasks where a single numerical value is expected. These tasks include: 1) maximum value, minimum value, maximum value within a specified range, minimum value within a specified range, 2) time with the maximum or minimum value³, 3) average, average within a specified range, sum, sum within a specified range, 4) magnitude comparison, difference, and 5) imputation value and forecasting value. **F1 score** is used for the tasks involving the detection of multiple values, such as peak and dip points, points exceeding or below a threshold. This metric assesses the ability to accurately identify events while minimizing false positives and false negatives.

For the computation tasks evaluated with Accuracy (i.e., mean, mean within a specified range, sum, sum within a specified range, difference, imputation value, and forecasting value), we implement a tolerance-based scoring because LLMs may

³Strictly speaking, multiple timestamps may exist for maximum and minimum values, but such cases were rare in our dataset. Thus, we treated these tasks as single-value predictions, accepting only the first timestamp output as correct if it matched any valid timestamp.

produce outputs that slightly deviate from the precise result due to numerical rounding and truncation, making exact matching challenging. In fact, our preliminary experiments show that all methods exhibited very low performance under an exact match evaluation. Therefore, we consider an output to be correct if it falls within $\pm 5\%$ of the ground truth value. A more detailed justification for this approach, including accuracy results across various tolerance levels (0% to 5%) and an assessment of error magnitudes using Median Absolute Error (MedAE), can be found in Appendix D.

4.3 Large Language Models

We evaluate four categories of LLMs: 1) API-based GPT models, 2) open-source models with approximately 7 billion parameters, 3) open-source models with approximately 70 billion parameters, and 4) reasoning-based models.

In the GPT category, we use GPT-3.5-turbo, GPT-4-turbo, GPT-4o, and GPT-4o-mini (Brown et al., 2020; OpenAI, 2024a,b). For open-source models with approximately 7 billion parameters LLMs: Llama-2-7B, Llama-2-7B-Instruct, Llama-3.1-8B, Llama-3.1-8B-Instruct (Meta, 2023, 2024), Llama-3.1-Swallow-8B and Llama-3.1-Swallow-8B-Instruct (Fujii et al., 2024), and Gemma-2 (Gemma Team, 2024).

For open-source models with approximately 70 billion parameters, we evaluate Llama-3.1-70B (Meta, 2024), Llama-3.1-Swallow-70B-v0.1 (Fujii et al., 2024), and Llama-3.1-Swallow-70B-Instruct (Fujii et al., 2024). Finally, for the reasoning-based category, we include OpenAI o1 (OpenAI, 2024c) and DeepSeek-R1-Distill-Llama-8B (Distill-Llama) (DeepSeek-AI, 2025).

4.4 Implementation Details

In all experiments, we set the temperature to 0.2 and limited the generation to 128 tokens. For experiments with local models, we used Hugging Face’s transformers library⁴ without applying 4-bit or 8-bit quantization, and inference was performed using a batch size of 1. We conducted inference on A100 80GB GPUs: models with approximately 7 billion parameters ran on a single GPU, whereas models with approximately 70 billion parameters were distributed across four GPUs. During evaluation, any output that did not adhere to the specified format was deemed incorrect. To account for vari-

⁴<https://huggingface.co/docs/transformers>

	Ins.	Rea.	Full		Range		Time		Extrema		Thresh		Full		Range		Pair		Inf	
			max	min	max	min	max	min	peak	dip	exc	bel	avg	sum	avg	sum	diff	comp	imp	fcst
GPT-3.5-turbo	✓		.98	.95	.35	.32	.39	.38	.34	.28	.82	.74	.49	.13	.25	.26	.26	.57	.50	.27
GPT-4-turbo	✓		1.	.99	.77	.76	.86	.87	.54	.46	.95	.84	.83	.56	.25	.53	.92	.77	.91	.27
GPT-4o	✓		1.	.99	.76	.70	.94	.91	.64	.52	.98	.98	.74	.50	.30	.48	.79	.83	.90	.26
GPT-4o-mini	✓		.98	.95	.35	.49	.56	.68	.35	.39	.90	.93	.60	.17	.26	.25	.40	.81	.61	.29
Llama2-7B	✓		.47	.41	.20	.15	.08	.07	.14	.09	.35	.29	.41	.05	.30	.03	.07	.32	.25	.18
Llama3.1-8B	✓		.72	.68	.20	.17	.10	.10	.24	.17	.35	.27	.40	.03	.26	.03	.13	.52	.01	.15
Swallow-8B	✓		.89	.85	.27	.30	.30	.24	.24	.24	.52	.48	.52	.17	.27	.13	.56	.48	.32	.09
Gemma2-9B	✓		.89	.84	.29	.25	.32	.27	.25	.24	.52	.50	.46	.08	.25	.12	.47	.46	.28	.10
Llama3.1 70B	✓		.97	.95	.43	.55	.63	.55	.19	.23	.59	.56	.61	.47	.31	.32	.87	.79	.68	.27
Swallow-70B	✓		.98	.97	.60	.56	.80	.74	.14	.44	.89	.84	.25	.14	.11	.21	.84	.83	.77	.27
OpenAI o1	✓	✓	1.	1.	.97	.96	1.	.98	.82	.79	1.	1.	1.	1.	.23	.90	.97	.99	1.	.56
Distill-Llama	✓	✓	.74	.76	.73	.69	.70	.68	.36	.40	.48	.31	.20	.48	.15	.48	.10	.35	.74	.23

Table 2: Evaluation results for event detection and computation tasks. A ✓ in the 'Ins.' column indicates instruction-tuned models. A ✓ in the 'Rea.' column indicates reasoning models. Thresh: Threshold-based detection (exc: exceed, bel: below). Pair: Pairwise comparison tasks (diff: difference, comp: magnitude comparison). Inf: Inference tasks (imp: imputation, fcst: forecasting). The numbers in **bold** indicate the highest values within each model category, and the numbers highlighted in **blue** represent the highest values among all models.

ability in generation, we ran each experiment five times using the same prompt and reported the average score across the runs.

5 Main Results

Which Tasks Are Difficult for LLMs?: Table 2 presents the results of the event detection tasks, computation tasks, and pairwise comparison tasks. Among non-reasoning-based models (i.e., excluding OpenAI o1 and Distill-Llama), GPT-4-turbo and GPT-4o generally achieve the best performance. In relatively simple tasks, such as detecting the maximum and minimum values in a time series, both models perform almost perfectly (1.0 for maximum detection and 0.99 for minimum detection). They also show high accuracy in detecting points exceeding (0.98 for GPT-4o, 0.95 for GPT-4-turbo) or falling below (0.98 for GPT-4o, 0.84 for GPT-4-turbo) a given threshold.

However, certain tasks remain challenging. In event detection, detecting peak and dip points proves particularly difficult, with GPT-4o achieving only 0.64 for peak detection and 0.52 for dip detection. Additionally, computation tasks also pose challenges—for example, GPT-4o scores only 0.50 on the summation task. These results indicate that while LLMs perform well on simpler numerical time-series interpretation tasks, more complex

computations remain difficult.

Does Model Size Improve Performance?: Increasing the model size in both Llama and Swallow leads to significant performance improvements across all tasks. For example, we observe a substantial accuracy increase from 0.34 (Llama3.1-8B) to 0.60 (Llama3.1-70B). This size-driven gain was statistically significant (Appendix G).

This trend is consistent across different task categories, including computation and pairwise comparison tasks, as shown in Table 2. These results suggest that larger models generally achieve better performance, reinforcing the importance of model scaling in numerical reasoning tasks.

Does Instruction Tuning Improve Numerical Reasoning?: Instruction-tuning is known to improve performances of various language processing tasks (Zhou and Zhao, 2024). However, it does not necessarily improve numerical reasoning abilities for most cases. In fact, for both the Llama series (Llama2-7B and Llama3.1-8B) and Swallow-8B, we observed performance degradation. For example, in Table 2, Llama3.1-8B without instruction tuning achieved 0.52 on the average calculation for the whole time series, whereas adding instruction tuning decreased performance to 0.32. This suggests that instruction tuning, while beneficial for general NLP tasks, does not consistently enhance

numerical comprehension.

The Impact of “Range” Conditions on Performance: When event detection or computation is restricted to a specific range, accuracy drops significantly. For example, in maximum and minimum value detection, GPT-4o’s performance drops from 1.0 (whole time-series) to 0.76 and 0.70, respectively, when range conditions are introduced. This suggests that tasks requiring multi-step reasoning are particularly challenging. In the maximum value detection task within a specific range, the model needs to first extract the values within the specified range and then identify the maximum value from that subset. This additional step introduces complexity, potentially leading to lower accuracy. Reasoning-based models, such as OpenAI o1 and Distill-Llama, appear to have an advantage in such multi-step reasoning tasks. Their performance under range conditions will be discussed next.

Performance of Recent Reasoning-Based Models: OpenAI o1 and Distill-Llama: OpenAI o1 demonstrates the highest scores across all tasks, except for average computation within a specific range. For example, in the maximum value detection task within a range, GPT-4o achieved only 0.76, whereas OpenAI o1 attained 0.97, showing its superiority to multi-step reasoning, i.e., extracting values within the specified range and finding the maximum. Additionally, OpenAI o1 exhibits low variance across different tasks, suggesting greater robustness in numerical time-series interpretation. The differences between OpenAI o1’s performance and others were statistically significant (Appendix G).

However, OpenAI o1 does not always show a clear advantage. In the computation task of averaging within a specific range, it achieved only 0.23, a score comparable to other GPT models. While reasoning-based models excel in many NLP tasks, they fail to outperform others in certain numerical reasoning tasks such as constrained averaging.

Distill-Llama is an open-source reasoning-based model built on Llama3.1-8B. Due to its smaller model size, its performance is consistently lower than OpenAI o1. Distill-Llama demonstrates significant performance improvements over its base model, Llama3.1-8B. For example, we observed the performance gains in: 1) the event detection tasks with a specific range and 2) simple computation tasks, such as summation. In the maximum value detection task within a range, Distill-Llama achieved 0.73, a substantial improvement over

Llama3.1-8B’s 0.34. Similarly, in the summation task, Distill-Llama often outperformed Llama3.1-8B, improving from 0.24 to 0.48. For event detection tasks without a range, Llama3.1-8B already achieved high accuracy (0.80), while Distill-Llama obtained a comparable score (0.74), offering no clear advantage. Distill-Llama also underperformed the base model on the averaging task with a specific range, where the base model achieved 0.20 while Distill-Llama achieved only 0.15.

Divergent Performance on Linear Imputation and Forecasting Tasks: Beyond event detection and direct computation, we investigated models’ abilities on time-series imputation and forecasting, detailed in Table 2.

Although both tasks involve linear prediction, their performances differed greatly. Most models, particularly high-performing ones like OpenAI o1 (1.00), GPT-4o (0.90), and GPT-4-turbo (0.91), showed strong proficiency in linear imputation. This suggests LLMs can generally execute linear interpolation, filling missing values using the linear relationship between immediately adjacent known points. Conversely, all models performed significantly worse on linear forecasting. Even OpenAI o1, excelling at imputation, scored only 0.56 on forecasting. GPT-4o (0.26) and GPT-4-turbo (0.27) scores were markedly lower.

6 Analysis

6.1 Examples of Generated Reasoning Tokens

One key advantage of reasoning-based models is their explainability.⁵ Unlike standard LLMs, Distill-Llama generates reasoning tokens, allowing us to examine its thought process when solving numerical tasks. To understand how the model reasons through numerical time-series problems, we manually analyzed its reasoning tokens.

Table 3 presents two representative examples from the maximum value detection task within a specified range, a task where Distill-Llama outperforms its base model, Llama3.1-8B. In the first example, the model successfully identifies the values within the specified range (2010–2008) and correctly determines the maximum value. Such successful instances demonstrate that Distill-Llama can effectively extract relevant numerical information and apply the correct reasoning process.

⁵Since OpenAI’s API does not provide access to reasoning tokens, we use Distill-Llama for our analysis.

Evaluation	Reasoning Process
Correct (GT: 8.3)	First, I observe that the time series provided starts in 2008 and ends in 2010, so I need to extract the relevant data points that fall within the specified range of 2010 to 2008. Looking at each data point: [2010, 7.37], [2009, 8.3], [2008, 6.21]. These are the data points within the range from 2010 to 2008. Then, I identify the maximum value among these data points. The values are: 7.37 (from 2010), 8.3 (from 2009), 6.21 (from 2008). Among these, the highest value is 8.3. Finally, I conclude that the maximum value within the specified range is 8.3 .
Incorrect (GT: 1.70)	First, I need to identify the time series provided in the question. The series starts with the year 2020 and goes down to the year 1999. Next, I'll locate the range specified in the problem, which is from the year 2008 to the year 1999. This means I should consider only the data points from 2008 backward to 1999. I'll extract the corresponding values for these years: - 2008: 1.70- 2007: 0.40- 2006: 0.58- 2005: 0.67- 2004: 0.72- 2003: 0.75- 2002: 0.75- 2001: 0.72- 1999: 0.76. Now, I'll compare all these values to determine the maximum. Among the listed values, the highest number is 0.76. Therefore, the maximum value within the specified range is 0.76. **Final Answer:**0.76

Table 3: Reasoning process for the range-based tasks. GT stands for ground truth.

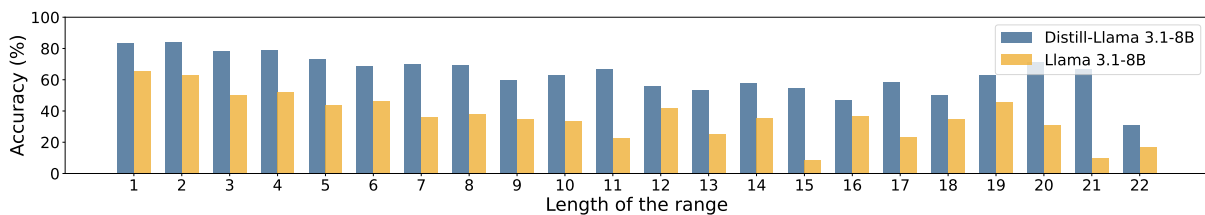


Figure 2: The performances of Distill-Llama and the base model, Llama3.1-8B, depending on the lengths of range.

However, the second example illustrates a failure case: while the model correctly extracts the values within the specified range (1999–2020), it fails to identify the maximum value accurately. This suggests that although the model is capable of range-based numerical selection, it does not always execute the final reasoning step correctly. Understanding such failure cases provides insights into how reasoning-based models can be further improved for numerical time-series tasks.

6.2 Why Distill-Llama Fails on Averaging?

Our analysis reveals that Distill-Llama struggles primarily with extracting values within a specified range when performing average computation. In all 20 incorrect cases we examined, the model failed to correctly extract the subset of values required for averaging, leading to cascading errors in the subsequent summation and division steps. Notably, while Distill-Llama successfully extracts values in the summation task, it fails to do so in the average computation task. This discrepancy suggests that the additional step of counting values before division increases the likelihood of errors.

Solving an average computation task within a specified range involves at least six steps: 1) understanding the task (recognizing that the goal is to compute an average), 2) identifying the specified range in the prompt, 3) extracting the correct val-

ues that fall within the specified range, 4) summing the extracted values, 5) counting the number of extracted values, finally 6) dividing the sum by the count to compute the final average.

Interestingly, Distill-Llama successfully extracts values in the summation task for most cases but fails to do so in the average computation task. Since value extraction accuracy varies depending on the task, an effective improvement could be to handle this step as a preprocessing stage rather than relying on the LLM, from the viewpoint of application. By extracting relevant values before passing them to the model, we could reduce errors introduced during multi-step reasoning and improve overall numerical computation performance.

6.3 Effect of range length

Figure 2 presents a comparison of accuracy based on the length of the specified range in the maximum value detection within a specified range. Distill-Llama outperforms Llama3.1 across all range lengths, indicating that incorporating a reasoning process leads to enhanced performance. While Llama3.1 tends to exhibit decreased accuracy as the range length increases, Distill-Llama maintains its performance even with longer ranges.

7 Conclusion

This paper introduced a set of 18 evaluation tasks for assessing the interpretation abilities of LLMs on numerical time-series. Our benchmark focuses on four aspects: 1) *Event Detection*—identifying specific values such as maxima and minima; 2) *Computation*—averaging and summation; 3) *Pairwise Comparison*—comparing values across time; and 4) *Inference*—imputation and forecasting. Each task expects a numeric answer, making evaluation fully automatic and comparable across models and prompts. Our experiments show that even state-of-the-art LLMs still struggle with multi-step reasoning, lose accuracy when the calculation must stay within a specific range, gain inconsistent benefits from instruction turning, and interpolation is more easily than forecasting; models with explicit reasoning steps handle the hardest tasks best. These findings establish a clear baseline and highlighting the remaining gap between fluent text generation and dependable numeric precision.

Limitations

While our benchmark provides a comprehensive evaluation of LLMs’ ability to interpret numerical time-series data, several limitations remain as follows.

Task Scope

Our evaluation focuses on fundamental numerical interpretation tasks, such as event detection, basic arithmetic computations, and numerical comparisons. However, real-world applications often require more complex reasoning, such as trend analysis, anomaly detection, and forecasting. Moreover, many practical applications involve downstream tasks such as data-to-text generation (e.g., generating textual summaries from financial or climate data). While our benchmark assesses isolated numerical reasoning abilities, it does not directly evaluate how these abilities translate into improvements in downstream tasks. Future work should integrate evaluations on data-to-text generation and other practical applications to better understand the real-world impact of numerical reasoning capabilities.

Challenges in Zero-shot Evaluation

Ideally, LLMs should be evaluated in a zero-shot setting, as this better reflects their generalization capabilities. However, our experiments revealed

that models often fail to follow output format constraints when using zero-shot prompts, making direct performance comparison difficult. To ensure fair comparisons, we had to introduce few-shot learning and additional formatting instructions. Future work should explore better strategies for enforcing output constraints while maintaining zero-shot evaluation.

Dataset

Our benchmark is constructed using a single dataset derived from the public and re-distributable Chart-to-Text dataset (Kantharaj et al., 2022). While this dataset covers various numerical time-series domains (e.g., finance, crime statistics, and mortality rates), it would remain limited in scope. If licensing issues allow, expanding our benchmark with additional datasets from diverse domains—such as medical time-series data, financial reports, or sensor-based data—would provide a more comprehensive assessment of LLMs’ numerical reasoning abilities.

Modality

Our evaluation relies exclusively on text-based numerical representations, whereas real-world numerical data is often stored and processed in tabular, spreadsheet, or graphical formats (e.g., line charts). This discrepancy may limit the benchmark’s applicability to multimodal LLMs designed to process structured data directly. Future research should explore evaluation frameworks that assess models’ ability to interpret structured numerical data from diverse modalities (e.g., tables and visual charts).

Computational Costs and Reproducibility

Large-scale open-source models, such as Llama-3.1-70B, require substantial computational resources for inference. While GPT models (e.g., GPT-4o) offer higher performance, they introduce reproducibility concerns due to potential model updates over time. Future work should explore techniques to efficiently fine-tune smaller models for numerical reasoning while maintaining accuracy.

Acknowledgments

This paper is based on results obtained from a project JPNP20006, commissioned by the New Energy and Industrial Technology Development Organization (NEDO), and AIST policy-based budget project “R&D on Generative AI Foundation Models for the Physical Domain”.

References

- Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. [Large language models for mathematical reasoning: Progresses and challenges](#). In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 225–237, St. Julian’s, Malta. Association for Computational Linguistics.
- Mubashara Akhtar, Abhilash Shankarampeta, Vivek Gupta, Arpit Patil, Oana Cocarascu, and Elena Simperl. 2023. [Exploring the numerical reasoning capabilities of language models: A comprehensive analysis on tabular data](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15391–15405, Singapore. Association for Computational Linguistics.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. [Language models are few-shot learners](#). *Preprint*, arXiv:2005.14165.
- Chung-Chi Chen, Hiroya Takamura, Ichiro Kobayashi, and Yusuke Miyao. 2024. [The impact of language on arithmetic proficiency: A multilingual investigation with cross-agent checking computation](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 631–637, Mexico City, Mexico. Association for Computational Linguistics.
- Katherine M. Collins, Albert Qiaochu Jiang, Simon Frieder, Li Siang Wong, Miri Zilka, Umang Bhatt, Thomas Lukasiewicz, Yuhuai Wu, Joshua B. Tenenbaum, William Hart, Timothy Gowers, Wen-Ding Li, Adrian Weller, and Mateja Jamnik. 2023. [Evaluating language models for mathematics through interactions](#). *ArXiv*, abs/2306.01694.
- DeepSeek-AI. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#). *Preprint*, arXiv:2501.12948.
- Elizabeth Fons, Rachneet Kaur, Soham Palande, Zhen Zeng, Tucker Balch, Manuela Veloso, and Svitlana Vyetrenko. 2024. [Evaluating large language models on time series feature understanding: A comprehensive taxonomy and benchmark](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21598–21634, Miami, Florida, USA. Association for Computational Linguistics.
- Kazuki Fujii, Taishi Nakamura, Mengsay Loem, Hiroki Iida, Masanari Ohi, Kakeru Hattori, Hirai Shota, Sakae Mizuki, Rio Yokota, and Naoaki Okazaki. 2024. [Continual pre-training for cross-lingual llm adaptation: Enhancing japanese language capabilities](#). In *Proceedings of the First Conference on Language Modeling*, COLM, page (to appear), University of Pennsylvania, USA.
- Google DeepMind Gemma Team. 2024. [Gemma 2: Improving open language models at a practical size](#). *Preprint*, arXiv:2408.00118.
- Yumi Hamazono, Yui Uehara, Hiroshi Noji, Yusuke Miyao, Hiroya Takamura, and Ichiro Kobayashi. 2020. [Market comment generation from data with noisy alignments](#). In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 148–157, Dublin, Ireland. Association for Computational Linguistics.
- Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. 2024. [Time-LLM: Time series forecasting by reprogramming large language models](#). In *International Conference on Learning Representations (ICLR)*.
- Shankar Kantharaj, Rixie Tiffany Leong, Xiang Lin, Ahmed Masry, Megh Thakkar, Enamul Hoque, and Shafiq Joty. 2022. [Chart-to-text: A large-scale benchmark for chart summarization](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4005–4023, Dublin, Ireland. Association for Computational Linguistics.
- Masayuki Kawarada, Tatsuya Ishigaki, and Hiroya Takamura. 2024a. [Prompting for numerical sequences: A case study on market comment generation](#). In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 13190–13200, Torino, Italia. ELRA and ICCL.
- Masayuki Kawarada, Tatsuya Ishigaki, Goran Topić, and Hiroya Takamura. 2024b. [Demonstration selection strategies for numerical time series data-to-text](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7378–7392, Miami, Florida, USA. Association for Computational Linguistics.
- Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024a. [Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers](#). *Preprint*, arXiv:2402.19255.
- Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024b. [GSM-plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers](#). In *Proceedings of the 62nd Annual Meeting of the Association*

for *Computational Linguistics (Volume 1: Long Papers)*, pages 2961–2984, Bangkok, Thailand. Association for Computational Linguistics.

Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, Wenwei Zhang, Songyang Zhang, Dahua Lin, and Kai Chen. 2024a. [Mathbench: Evaluating the theory and application proficiency of llms with a hierarchical mathematics benchmark](#). *Preprint*, arXiv:2405.12209.

Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, Wenwei Zhang, Songyang Zhang, Dahua Lin, and Kai Chen. 2024b. [MathBench: Evaluating the theory and application proficiency of LLMs with a hierarchical mathematics benchmark](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6884–6915, Bangkok, Thailand. Association for Computational Linguistics.

Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. 2023. [A survey of deep learning for mathematical reasoning](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14605–14631, Toronto, Canada. Association for Computational Linguistics.

Meta. 2023. [Llama 2: Open foundation and fine-tuned chat models](#). *Preprint*, arXiv:2307.09288.

Meta. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.

Soichiro Murakami, Akihiko Watanabe, Akira Miyazawa, Keiichi Goshima, Toshihiko Yanase, Hiroya Takamura, and Yusuke Miyao. 2017. [Learning to generate market comments from stock prices](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1374–1384, Vancouver, Canada. Association for Computational Linguistics.

OpenAI. 2024a. [Gpt-4 technical report](#). *Preprint*, arXiv:2303.08774.

OpenAI. 2024b. [Gpt-4o system card](#). *Preprint*, arXiv:2410.21276.

OpenAI. 2024c. [Openai o1 system card](#). *Preprint*, arXiv:2412.16720.

Xiyuan Zhang, Ranak Roy Chowdhury, Rajesh K. Gupta, and Jingbo Shang. 2024. [Large language models for time series: A survey](#). *Preprint*, arXiv:2402.01801.

Yongwei Zhou and Tiejun Zhao. 2024. [Dual instruction tuning with large language models for mathematical reasoning](#). *Preprint*, arXiv:2403.18295.

A Prompt Format

Figure 3 shows the 5-shot prompt structure used to guide the Large Language Model in accurately identifying the maximum value within a time series; the model is explicitly instructed to respond with only a number, continuing directly after the Maximum value: cue.

```
Which is the value that is the maximum?
Please respond with the maximum value only as a
number.

###
Examples (5-shot):
Time series:
{"2000":12, "2001":45, "2002":23, "2003":67,
"2004":34}
Maximum value: 67
###

Time series: [INPUT].
Maximum value:
```

Figure 3: 5-shot prompt used to identify the maximum value in a time series.

B Prompt Ablation

We compared five paraphrased prompts (p_1 – p_5) per task on the development set; Figure 12 shows accuracy relative to p_1 , and Table 4 lists the variants. We use p_1 for all main results.

C Error Rate for Output Format

Figure 13 shows the heatmap of format error rates. A format error is defined as any output containing non-numeric text. The values were computed automatically by a script that checked all model outputs. This visualization highlights failure cases caused not by incorrect reasoning but by outputs in an invalid format.

D Accuracy for Computation Category with Tolerance

Tables 5, 6, 7, and 8 show the values for the average, average within a specified range, sum, and sum within a specified range, where answers are now considered correct if they fall within a $\pm 5\%$ error margin of the true value. This revised evaluation criterion was adopted because initial evaluations using strict exact match—under which the non-reasoning model, for example, scored nearly 0.0 on almost all tasks—indicated that exact match was too stringent. The results presented in these tables

are further visualized in Figures 4, 5, 6, 7, 8, 9, and 10. These figures depict the trend of accuracy for each task as a function of varying error tolerance levels, illustrating how performance changes as the tolerance for numerical discrepancies is adjusted.

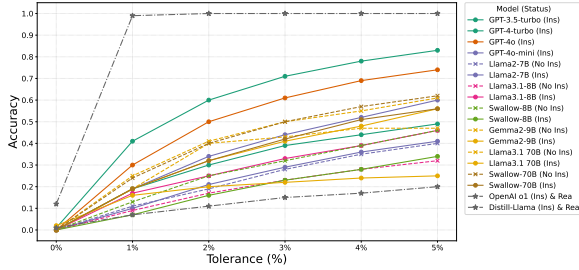


Figure 4: Trend in Accuracy of Computation Average task with Varying Tolerance from 0% to 5%

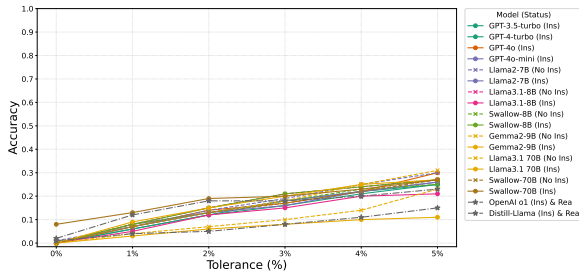


Figure 5: Trend in Accuracy of Computation Average w/range task with Varying Tolerance from 0% to 5%

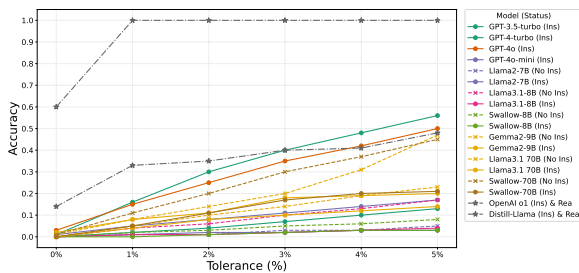


Figure 6: Trend in Accuracy of Computation Sum task with Varying Tolerance from 0% to 5%

E Computation Task Performance: Median Absolute Error (MedAE)

Table 9 shows the median absolute errors for different LLMs.

F Comparisons on Languages of Prompt

We use Japanese LLMs, i.e., Llama-3.1-Swallow-8B and Llama-3.1-Swallow-8B-Instruct (Fuji

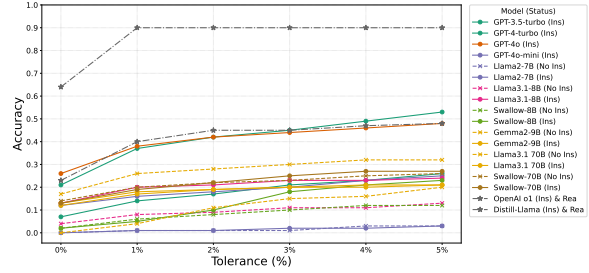


Figure 7: Trend in Accuracy of Computation Sum w/range task with Varying Tolerance from 0% to 5%

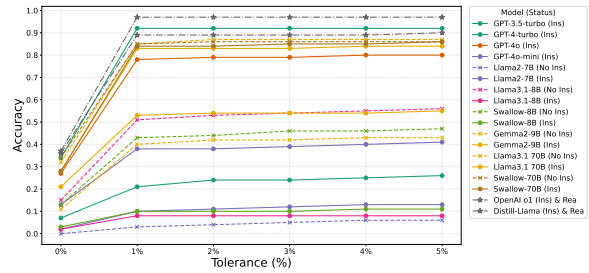


Figure 8: Trend in Accuracy of Computation Difference task with Varying Tolerance from 0% to 5%

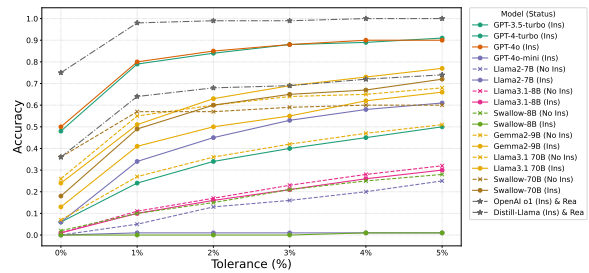


Figure 9: Trend in Accuracy of imputation task with Varying Tolerance from 0% to 5%

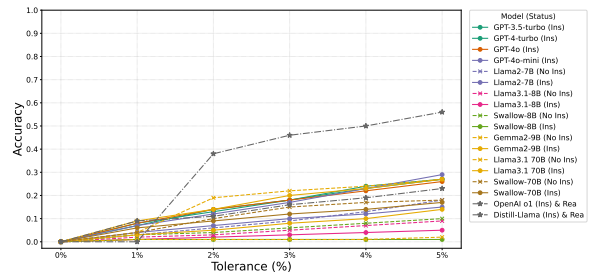


Figure 10: Trend in Accuracy of forecasting task with Varying Tolerance from 0% to 5%

et al., 2024) for analyzing the effects of prompts' languages. Figure 11 presents the performance differences observed when using Japanese and English prompts. We find that the multilingual GPTs i.e., GPT-4 and Llama, achieve higher accuracy

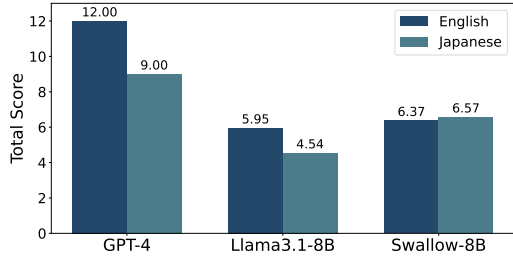


Figure 11: Comparison by Prompt Language.

that scaling improves accuracy.

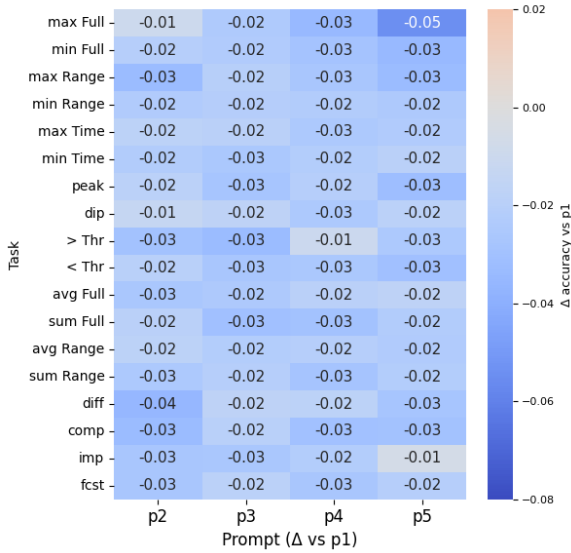


Figure 12: Prompt sensitivity on the development set. Each cell shows $\Delta = Acc(p_k) - Acc(p_1)$ (model-averaged accuracy) for paraphrased prompt p_k relative to the reference prompt p_1 (blue = lower, red = higher). Task labels are abbreviated; see Table 4 for full prompts. Most differences are small ($|\Delta| \leq 0.03$), supporting our use of p_1 in the main experiments.

with English prompts. In contrast, Swallow, an LLM adapted to Japanese through continual training, performs better with Japanese prompts. It is reasonable to conclude that prompts should be written in a language that was more extensively used during pretraining.

G Statistical Significance Analysis

We assessed statistical significance on per-task accuracies across $T=18$ tasks using paired Wilcoxon signed-rank test. First, OpenAI o1 versus each other model: all pairwise comparisons yielded $p < 0.001$, confirming that o1’s superiority is consistent across tasks and unlikely due to chance. Second, model size (within Llama3.1): the larger variant (70B) significantly outperformed smaller one (8B) across the same tasks ($p < 0.001$), indicating

Family	Prompt text (p1–p5 variants)
Value (Max/Min; Full/Range)	<p>p1: Which is the value that is the <OP> [SCOPE]?</p> <p>p2: What is the <OP> value [SCOPE]?</p> <p>p3: Within [SCOPE], which number is <OP>?</p> <p>p4: Find the <OP> value [SCOPE].</p> <p>p5: Among values [SCOPE], which is <OP>?</p>
Time (Max/Min)	<p>p1: Which time point corresponds to the <OP> value?</p> <p>p2: Return the time index of the <OP> value.</p> <p>p3: At what time does the <OP> occur?</p> <p>p4: Find the time step with the <OP> value.</p> <p>p5: Which time has the <OP> value?</p>
Extrema (Peak/Dip)	<p>p1: Which is the <OP> value in the following time series data?</p> <p>p2: Return the highest/lowest local <OP> in the following time series data.</p> <p>p3: Which value is a local <OP> in the following time series data?</p> <p>p4: Find the <OP> value. in the following time series data</p> <p>p5: What is the biggest/smallest spike?</p>
Threshold (Exceed/Below)	<p>p1: Which year does the value become [REL] [THR]?</p> <p>p2: When does the series first cross [REL] [THR]?</p> <p>p3: List all years where value [REL] [THR].</p> <p>p4: Give the indices where the threshold [THR] is crossed [DIR].</p> <p>p5: Report the years that the value is [REL] [THR].</p>
Computation (Avg/Sum; Full/Range)	<p>p1: Calculate the <AGG> value of the range [SCOPE].</p> <p>p2: What is the <AGG> value of the range [SCOPE]?</p> <p>p3: Compute the <AGG> in the specified range [SCOPE].</p> <p>p4: Find the <AGG> for the given data [SCOPE].</p> <p>p5: Return the <AGG> value restricted to [SCOPE].</p>
Pairwise Diff	<p>p1: Calculate the absolute difference between the values for [T1] and [T2].</p> <p>p2: What is $\text{value}([T1]) - \text{value}([T2])$?</p> <p>p3: Compute the absolute gap between [T1] and [T2].</p> <p>p4: Find the absolute difference between [T1] and [T2].</p> <p>p5: Return the absolute difference between the values for [T1] and [T2].</p>
Pairwise Compare	<p>p1: Compare the values of [T1] and [T2].</p> <p>p2: Which is larger: [T1] or [T2]?</p> <p>p3: Check if [T1] and [T2] are the same.</p> <p>p4: Compare [T1] and [T2].</p> <p>p5: Return the relation (>, <, =) between [T1] and [T2].</p>
Imputation	<p>p1: Linearly interpolate the NaN value using the data points immediately before and after it.</p> <p>p2: Fill the NaN by performing a linear interpolation between the points immediately preceding and succeeding it.</p> <p>p3: Replace the missing value (NaN) by drawing a straight line between its two adjacent data points.</p> <p>p4: Use the values directly before and after the NaN to estimate its value via linear interpolation.</p> <p>p5: Calculate a replacement for the NaN by linearly interpolating from its nearest neighboring data points.</p>
Forecast	<p>p1: Predict the value for the next chronological period using linear regression based on the provided series.</p> <p>p2: Using the provided time series, create a linear regression model to forecast the value for the upcoming period.</p> <p>p3: Fit a straight-line trend to the given data series and use it to predict the value for the subsequent period.</p> <p>p4: Apply a linear regression to the existing data to determine the expected value for the next time step.</p> <p>p5: Extrapolate the next value in the series by fitting the historical data with a linear regression model.</p>

Table 4: Prompt templates used in the prompt ablation study (Appendix B). Templates are grouped by **prompt family**. Task-specific prompts are instantiated by substituting the bracketed placeholders: [OP] (operation; e.g., maximum, minimum, peak, dip), [AGG] (aggregate; average or sum), [REL] (relational phrase used for threshold crossing; “bigger than” / “smaller than”), [SCOPE] (specified range; “from 2001 to 2005”), [THR] (threshold), [T1] and [T2] (specific years).



Figure 13: Heatmap of format error rates (% of outputs containing non-numeric text).

Model	Ins.	Rea.	Full					Range						
			0%	1%	2%	3%	4%	5%	0%	1%	2%	3%	4%	5%
GPT-3.5-turbo	✓		.00	.19	.30	.39	.44	.49	.00	.06	.12	.18	.22	.25
GPT-4-turbo	✓		.01	.41	.60	.71	.78	.83	.00	.06	.12	.16	.21	.25
GPT-4o	✓		.00	.30	.50	.61	.69	.74	.00	.07	.14	.17	.22	.30
GPT-4o-mini	✓		.00	.19	.34	.44	.52	.60	.00	.08	.13	.18	.22	.26
Llama2-7B			.00	.11	.19	.28	.35	.40	.01	.07	.14	.19	.25	.30
Llama2-7B	✓		.00	.10	.21	.29	.36	.41	.00	.07	.13	.16	.22	.26
Llama3.1-8B			.00	.09	.17	.23	.28	.32	.00	.08	.15	.21	.24	.27
Llama3.1-8B	✓		.00	.17	.25	.33	.39	.46	.00	.05	.12	.15	.20	.21
Swallow-8B			.00	.13	.25	.32	.39	.46	.00	.08	.13	.18	.23	.25
Swallow-8B	✓		.00	.07	.16	.23	.28	.34	.00	.08	.15	.21	.24	.27
Gemma2-9B			.00	.19	.40	.43	.47	.47	.00	.04	.07	.10	.14	.23
Gemma2-9B	✓		.00	.19	.32	.41	.48	.56	.00	.09	.15	.20	.25	.27
Llama3.1 70B			.00	.25	.41	.50	.55	.61	.00	.07	.13	.18	.25	.31
Llama3.1 70B	✓		.02	.16	.20	.22	.24	.25	.00	.03	.06	.08	.10	.11
Swallow-70B			.00	.24	.40	.50	.57	.62	.00	.07	.13	.17	.22	.27
Swallow-70B	✓		.00	.19	.32	.42	.51	.56	.08	.13	.19	.20	.23	.27
OpenAI o1	✓	✓	.12	.99	1.	1.	1.	1.	.02	.12	.18	.18	.20	.23
Distill-Llama	✓	✓	.01	.07	.11	.15	.17	.20	.01	.04	.05	.08	.11	.15

Table 5: Accuracy of Computation Average tasks (Average, Average w/range) with Varying Tolerance Levels.

Model	Ins.	Rea.	Full					Range						
			0%	1%	2%	3%	4%	5%	0%	1%	2%	3%	4%	5%
GPT-3.5-turbo	✓		.00	.02	.04	.07	.10	.13	.07	.14	.17	.21	.23	.26
GPT-4-turbo	✓		.01	.16	.30	.40	.48	.56	.21	.37	.42	.45	.49	.53
GPT-4o	✓		.03	.15	.25	.35	.42	.50	.26	.38	.42	.44	.46	.48
GPT-4o-mini	✓		.01	.05	.08	.11	.14	.17	.12	.16	.18	.20	.23	.25
Llama2-7B			.00	.01	.01	.03	.03	.05	.00	.01	.01	.01	.03	.03
	✓		.00	.01	.02	.02	.03	.03	.00	.01	.01	.02	.02	.03
Llama3.1-8B			.00	.04	.06	.10	.13	.17	.04	.08	.09	.11	.11	.13
	✓		.00	.01	.01	.02	.03	.04	.13	.20	.21	.23	.23	.24
Swallow-8B			.00	.02	.03	.05	.06	.08	.02	.06	.08	.10	.12	.12
	✓		.00	.00	.01	.02	.03	.03	.02	.05	.10	.18	.21	.23
Gemma2-9B			.00	.05	.10	.14	.19	.23	.00	.04	.11	.15	.16	.20
	✓		.02	.08	.11	.18	.19	.20	.13	.18	.19	.20	.20	.21
Llama3.1 70B			.01	.08	.14	.20	.31	.47	.17	.26	.28	.30	.32	.32
	✓		.00	.04	.08	.10	.12	.14	.12	.17	.19	.20	.21	.21
Swallow-70B			.01	.11	.20	.30	.37	.45	.14	.20	.22	.23	.25	.26
	✓		.00	.05	.11	.17	.20	.21	.13	.19	.22	.25	.27	.27
OpenAI o1	✓	✓	.60	1.	1.	1.	1.	1.	.64	.90	.90	.90	.90	.90
Distill-Llama	✓	✓	.14	.33	.35	.40	.41	.48	.23	.40	.45	.45	.47	.48

Table 6: Accuracy of Computation Sum tasks (Sum, Sum w/range) with Varying Tolerance Levels.

Model	Ins.	Rea.	imp					fcst						
			0%	1%	2%	3%	4%	5%	0%	1%	2%	3%	4%	5%
GPT-3.5-turbo	✓		.06	.24	.34	.40	.45	.50	.00	.07	.14	.18	.23	.27
GPT-4-turbo	✓		.48	.79	.84	.88	.89	.91	.00	.08	.13	.18	.24	.27
GPT-4o	✓		.50	.80	.85	.88	.90	.90	.00	.08	.14	.18	.22	.26
GPT-4o-mini	✓		.06	.34	.45	.53	.58	.61	.00	.07	.12	.17	.23	.29
Llama2-7B			.00	.05	.13	.16	.20	.25	.00	.03	.06	.09	.13	.18
	✓		.00	.01	.01	.01	.01	.01	.00	.04	.07	.10	.12	.15
Llama3.1-8B			.01	.11	.17	.23	.28	.32	.00	.02	.03	.05	.07	.09
	✓		.01	.10	.16	.21	.26	.30	.00	.01	.02	.03	.04	.05
Swallow-8B			.02	.10	.15	.21	.25	.28	.00	.03	.04	.06	.08	.10
	✓		.00	.00	.00	.00	.01	.01	.00	.01	.01	.01	.01	.01
Gemma2-9B			.07	.27	.36	.42	.47	.51	.00	.01	.01	.01	.01	.02
	✓		.13	.41	.50	.55	.62	.66	.00	.03	.05	.08	.10	.14
Llama3.1 70B			.26	.55	.60	.64	.65	.68	.00	.04	.19	.22	.24	.27
	✓		.24	.51	.63	.69	.73	.77	.00	.09	.14	.20	.23	.27
Swallow-70B			.36	.57	.57	.59	.60	.60	.00	.04	.10	.15	.17	.18
	✓		.18	.49	.60	.65	.67	.72	.00	.06	.09	.12	.14	.17
OpenAI o1	✓	✓	.75	.98	.99	.99	1.	1.	.00	.00	.38	.46	.50	.56
Distill-Llama	✓	✓	.36	.64	.68	.69	.72	.74	.00	.09	.11	.16	.19	.23

Table 7: Accuracy of Inference tasks (imputation, forecasting) with Varying Tolerance Levels.

Model	Ins.	Rea.	diff					
			0%	1%	2%	3%	4%	5%
GPT-3.5-turbo	✓		.07	.21	.24	.24	.25	.26
GPT-4-turbo	✓		.34	.92	.92	.92	.92	.92
GPT-4o	✓		.27	.78	.79	.79	.80	.80
GPT-4o-mini	✓		.13	.38	.38	.39	.40	.41
Llama2-7B			.00	.03	.04	.05	.06	.06
	✓		.02	.10	.11	.12	.13	.13
Llama3.1-8B			.15	.51	.53	.54	.55	.56
	✓		.02	.08	.08	.08	.08	.08
Swallow-8B			.13	.43	.44	.46	.46	.47
	✓		.03	.10	.10	.10	.11	.11
Gemma2-9B			.11	.40	.42	.42	.43	.43
	✓		.21	.53	.54	.54	.54	.55
Llama3.1 70B			.32	.85	.87	.87	.87	.87
	✓		.28	.83	.83	.83	.84	.84
Swallow-70B			.34	.85	.86	.86	.86	.86
	✓		.28	.84	.84	.85	.85	.86
OpenAI o1	✓	✓	.37	.97	.97	.97	.97	.97
Distill-Llama	✓	✓	.36	.89	.89	.89	.89	.90

Table 8: Accuracy of Computation Difference task with Varying Tolerance Levels.

	Ins.	Rea.	Full		Range		Pair	Inf	
			avg	sum	avg	sum	diff	imp	fcst
GPT-3.5-turbo	✓		1.10	120.00	3.50	36.00	1.30	1.30	3.10
GPT-4-turbo	✓		0.28	22.00	13.00	5.70	0.00	0.00	3.10
GPT-4o	✓		0.38	25.00	3.40	7.20	0.00	0.00	3.70
GPT-4o-mini	✓		0.74	78.00	4.80	25.00	0.87	0.78	3.10
Llama2-7B			1.70	350.00	3.50	240.00	13.00	16.00	5.60
	✓		1.80	400.00	4.00	420.00	5.70	3.40	7.60
Llama3.1-8B			0.81	73.00	2.80	52.00	0.10	2.40	5.60
	✓		1.50	730.00	3.00	56.00	8.00	2.40	38.00
Swallow-8B			1.10	130.00	2.70	40.00	0.50	17.00	5.00
	✓		2.10	140.00	2.30	42.00	5.80	2.90	3.40
Gemma2-9B			1.00	86.00	4.70	210.00	0.70	1.50	6.30
	✓		39.00	102.00	3.60	28.00	0.00	0.50	5.10
Llama3.1 70B			0.84	200.00	20.00	3.50	0.00	13.0	4.10
	✓		46.00	2.10	42.00	16.00	0.00	3.10	3.40
Swallow-70B			0.58	60.00	30.00	33.00	0.00	17.00	4.12
	✓		0.96	100.00	48.00	22.00	0.00	0.47	3.80
OpenAI o1	✓	✓	0.00	0.00	4.00	0.00	0.00	0.00	1.10
Distill-Llama	✓	✓	9.10	59.00	15.00	8.80	0.00	8.00	12.00

Table 9: MedAE for Computation Tasks.