LREVAL: Long-Chain Reasoning Evaluation for Large Language Models

Anonymous ACL submission

Abstract

The evaluation of reasoning capabilities is crucial for the advancement of Artificial General Intelligence. While Large Language Models (LLMs) demonstrate proficiency in reasoning tasks, existing benchmarks such as GSM8K and LogiQA are limited, focusing mainly on individual problem-solving with linear logic and static conditions. To bridge this gap, we introduce an automated data construction pipeline that simulates real-world reasoning scenarios by combining existing reasoning problems into more complex, long-chain reasoning problems. Based on this pipeline, a new benchmark, LREval, is designed to assess comprehensive reasoning skills, such as multi-step logical deduction, integration of diverse information sources, and dynamic decision-making. The evaluations underscore huge reasoning challenges faced by LLMs. Closed-source models perform well in dynamic contexts but struggle with integrating information from multiple sources, while open-source models exhibit the opposite trend. Moreover, model performance is highly sensitive to perturbations in task conditions, revealing the fragility of reasoning capabilities in current LLMs and the necessity for robust evaluation frameworks. Additionally, models struggle with tasks requiring simultaneous comprehension of multiple languages, further emphasizing their limitations in multilingual understanding.

1 Introduction

In recent years, large language models (LLMs) (Brown et al., 2020; OpenAI, 2023; Touvron et al., 2023; Bai et al., 2023) have made remarkable progress in foundational natural language processing (NLP) tasks and capability-specific tasks, showcasing impressive abilities in generation (Dubey et al., 2024a), reasoning (Huang and Chang, 2023), and instruction following (Zhang et al., 2023a). Reasoning ability, an essential aspect of general intelligence, has attracted significant attention from researchers. To evaluate the reasoning capability of LLMs, a variety of datasets have been employed, including mathematical reasoning benchmarks GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), logical reasoning dataset LogiQA (Liu et al., 2020), and commonsense reasoning dataset HellaSwag (Zellers et al., 2019).

These evaluations provide valuable insights into basic reasoning abilities. However, they fall short in offering a comprehensive assessment due to several limitations (Yang et al., 2024b). Current reasoning benchmarks tend to: 1) focus on isolated problems, emphasizing linear reasoning where each step depends only on the previous; 2) involve a single source of information, resulting in simple information processing requirements; 3) remain largely static, lacking the flexibility necessary to evaluate the models' ability to dynamically adapt to changes in conditions and context. Thus, a new dataset is essential to evaluate comprehensive reasoning capabilities, e.g., multi-step logical deduction, integration of multiple information sources, dynamic decision-making, and chaining of intermediate conclusions. These abilities not only encompass deep logical thinking requirements but also play a vital role in complex real-world scenarios.

Addressing the challenges of manual data collection, which is both costly and inefficient (Wu et al., 2024), we introduce an automated data construction pipeline. This pipeline simulates real-world reasoning scenarios by combining existing reasoning problems into more complex, long-chain reasoning problems. It consists of two key components including the collection of existing reasoning problems to form a subproblem set, and the combination of these subproblems into complex, long-chain tasks using two innovative strategies.

Specifically, we employ two core combination strategies including answer combination and problem jumping, as illustrated in Fig. 1. The answer combination strategy targets the evaluation of infor-

Answer Combination

The following is a long-chain reasoning problem that requires solving several sub-problems and then combining their answers into a single formatted string.

Let's break down the problem:
Answer question Q1.
Answer question Q2.
Concatenate the answers of Q1, and Q2 into a single string, separating them with '####'. Explain your answer step by step and give the final answer on the last line by itself in the format of 'The answer of Q1 is [answer1].
The answer of Q2 is [answer2].
The final answer is \boxed{[answer1]####[answer2]}'.

Where [answer1] must be integer, [answer2] must be one of A, B, C and D, and the final answer must be put within \boxed{}.

Q1: {Q1}
Q2: {Q2}

Problem Jumping

The following is a multi-hop long-chain reasoning problem that requires jumping to the subsequent question based on the answer to the previous question.

Initial question: {Q0}
Square each digit in the answer to the initial question, sum these squares, and then take the result modulo 2 to obtain a key number X.
First jump: If X is equal to 0, proceed to Q1_0. If X is equal to 1, proceed to Q1_1.

Q1_0: {Q1}
Q1_1: {Q2}

Solve the multi-hop question. Explain your answer step by step and give the final answer on the last line by itself in the format of
'The answer of the initial question is [answer0].
The first-jump question is [question1].
The answer of the first-jump question is [answer1].
The final answer is \boxed{[answer0]====[answer1]}'.

Where [answer0] must be integer, [answer1] must be one of A, B, C and D, and the final answer must be put within \boxed{}.

Figure 1: This figure presents the templates for the answer combination and problem-jumping strategies. The text highlighted in red indicates the parameters that need to be filled in.

mation integration and structured reasoning, requiring sequential resolution of independent subproblems. Conversely, the problem-jumping strategy, inspired by mechanisms in (Krosnick et al., 2010), examines dynamic decision-making, where subsequent subproblems depend conditionally on the answers to prior questions. Various logical jumping mechanisms, such as sequential ($q_2 \rightarrow q_3$), backtracking ($q_1 \rightarrow q_3$), and composite jumping ($(q_1, q_2) \rightarrow q_3$), offer robust testing of model adaptability. Appendix Figs 6 and 7 present examples of backtracking and composite jumping, respectively.

Utilizing this pipeline, we construct two versions of datasets including Easy and Hard, to accommodate different complexity evaluation needs, shown in Table 1. The Easy version applies only the answer combination strategy. The Hard subset depends exclusively on the problem-jumping strategy. Additionally, a Chinese benchmark is designed to support multilingual evaluation, showcasing the pipeline's versatility. Evaluations expose notable reasoning limitations in LLMs, e.g., the advanced reasoning model o1-mini exhibits reasoning gaps of 14.21% and 18.16% in the English Easy and Hard subset, respectively. Overall, our conclusions are as follows:

- Closed-source LLMs excel in flexible adaptation and dynamic decision-making, whereas open-source LLMs have a slight advantage in information integration and structured reasoning, highlighting differences in training data and strategies.
- LLMs below 7B parameters demonstrate limited capabilities in instruction understanding and reasoning.
- Even minor changes in task conditions affect model performance, underscoring the fragility of the reasoning capabilities of LLMs.
- Mainstream LLMs struggle with tasks requiring simultaneous multilingual comprehension, revealing their multilingual reasoning limitations.

2 Related Works

With the rapid development of LLMs, evaluating their capabilities, particularly reasoning ability, has become increasingly critical (Valmeekam et al., 2022). Recognizing its pivotal role, researchers have developed a variety of datasets to assess the reasoning abilities of LLMs (Chang et al., 2024). Typical examples include mathematical reasoning datasets like GSM8K (Cobbe et al., 2021), MCGSM8K (Zhang et al., 2024b), GaokaoBench-Math (Zhang et al., 2023b), TheoremQA (Chen et al., 2023), and MATH (Hendrycks et al., 2021). Additionally, logical reasoning datasets like LogiQA (Liu et al., 2020) and commonsense reasoning datasets such as HellaSwag (Zellers et al., 2019) have garnered significant attention within the research community.

Complementing these efforts, some studies have sought to explore the compositional reasoning abilities of models through multi-hop problems, where the overall solution relies on accurately combining answers to sub-problems. For instance, Press

Language	Subproblem Source	Dataset	Combination Strategy	Dataset Size	Subproblem Size	Difficulty Level
English	GSM8K:MATH:LogiQA = 400:400:400	Easy	Answer Combination	1000	2	1
		Hard	Problem Jumping	1000	2	2
			Problem Jumping	1000	3	3
			Problem Jumping	1000	4	4
Chinese	MGSM:LogiQA = 250:250	Easy	Answer Combination	500	2	1
		Hard	Problem Jumping	500	2	2

Table 1: An overview of the LREval benchmark. Overall, LREval contains test data in both Chinese and English, with each portion comprising the Easy and Hard portions.

et al. (2023) introduces the multi-hop question-answer task, e.g., "Who won the Master's Tournament the year Justin Bieber was born", designed to assess the application and reasoning of factual knowledge. Bhuiya et al. (2024) incorporates distractor paragraphs into the reading comprehension task, thereby concentrating on the ability to discern and synthesize information from multiple textual sources. Another study (Hosseini et al., 2024) chains two test questions together so that the answer to the first question is used as a variable in the second question, testing how well LLMs can combine learned concepts to solve new problems.

Unlike existing methods, our constructed dataset employs innovative strategies including problem jumping and answer combination to create long-chain reasoning challenges. Our dataset is designed to simulate real-world complex reasoning scenarios by pushing models to tackle tasks requiring multi-step logical reasoning, the integration of information from diverse sources, dynamic decision-making, and the sequential linking of intermediate conclusions. This approach not only evaluates how well models handle complex cognitive tasks but also reflects more authentic reasoning requirements, providing a more comprehensive assessment of LLM capabilities.

3 Data Construction

This section details the data construction pipeline. We begin with the creation of the subproblem set, proceed with the combination strategies of answer combination and problem jumping, and conclude with evaluation metrics and the entire data construction process.

3.1 Construction of Subproblem Set

As illustrated in Table 1, the English subproblems are drawn from three representative reasoning datasets: GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), and LogiQA (Liu et al., 2020). The GSM8K dataset comprises 7,000 grade school math word problems. The MATH dataset consists of 5,000 mathematics competition samples. The LogiQA dataset offers 650 logical comprehension samples available in both English and Chinese. From each of these datasets, we randomly sample 400 English examples, resulting in a combined collection of 1,200 subproblems.

For Chinese subproblems, we utilize MGSM (Shi et al., 2023) and LogiQA (Liu et al., 2020), as there is no Chinese equivalent for the MATH dataset. MGSM provides translations of 250 GSM8K examples across ten languages. We randomly choose 250 Chinese examples from MGSM and LogiQA, creating a set of 500 subproblems.

3.2 Combination Strategies

The strategies for combining problems are categorized into two types: answer combination and problem jumping.

3.2.1 Answer Combination

In the answer combination strategy, subproblems remain independent and must be solved sequentially in a specific order, e.g., $q_3 \rightarrow q_1 \rightarrow q_2$. The answers to the subproblems are then concatenated into a single string, separated by a fixed format including "===", "####", or "****". Key guidelines include: 1) Each subproblem has a clear input and a unique output. 2) Subproblems must be answered in a specific order to form the final answer. 3) The final answer is derived by concatenating the answers of all subproblems using a predetermined format. An example template is shown in the left part of Fig. 1, where the red text highlights the parameters to be filled.

3.2.2 Problem Jumping

In the problem-jumping strategy, there are clear conditional dependencies among the subproblems, as illustrated in the right part of Fig. 1. Each jump provides multiple candidate subproblems, with the

Algorithm 1 Algorithm for Constructing the Easy set

Require: Subproblem set $\mathbf{Q}$, string concatenation format set $\mathbf{Comb} = [\#\#\#\#, ====, ****]$, dataset size $\mathbf{M_E}$, subproblem size of the long-chain reasoning problems $\mathbf{S} = 2$, the answer combination template $\mathbf{T_C}$
Ensure: An Easy subset $\mathbf{D_E}$ contains $\mathbf{M_E}$ long-chain reasoning problems, each consisting of $\mathbf{S}$ subproblems combined using the answer combination strategy
Initialize with an empty list $\mathbf{D_E}$
$\mathbf{i} = 1$
while $\mathbf{i} \leq \mathbf{M}$ **do**
    $\mathbf{S}$ subproblems $[\mathbf{q_1}, ...\mathbf{q_S}]$ with different answers are randomly sampled from $\mathbf{Q}$
    A string concatenation format $\mathbf{c}$ is sampled from **Comb**
    Generate a random order $\mathbf{O}$ = (random.shuffle $[\mathbf{1}, ..., \mathbf{S}]$) to solve all the subproblems
    Construct a sample $\mathbf{p_i}$ by filling the answer combination template $\mathbf{T_C}$ with parameters $[\mathbf{q1}, ...\mathbf{q_S}]$, $\mathbf{c}$, $\mathbf{O}$
    Appending the new sample $\mathbf{p_i}$ into $\mathbf{D_E}$
    $\mathbf{i} \leftarrow \mathbf{i} + 1$
Return $\mathbf{D_E}$

path determined by the answers to the preceding questions. The specific guidelines are as follows: 1) Each subproblem has a clear input and a unique output. 2) Within the same jump, each subproblem yields a unique answer. 3) Subproblems may connect to one or more non-sequential subproblems. 4) The final answer is derived by concatenating the answers of all subproblems using a predetermined format.

3.2.3 Diversified Jumping Rules

Jumping rules are crafted based on the format of the answers, which are categorized into option labels, integers, and strings. Our design follows these guiding principles: 1) Each jump action associated with an answer is distinct and unambiguous, ensuring that there are no overlapping or confusing options. 2) Jumping rules are designed based on straightforward logic or basic arithmetic (addition, subtraction, multiplication, division, and modulus), to maintain simplicity. In line with these principles, we have developed the following jumping rules:

Option Labels: Direct jumping (R_{o1}). Specifically, each option label (e.g., A, B, C, D) corresponds to a specific candidate subproblem. For example, option label A maps to the first candidate subproblem, while option label D maps to the fourth candidate subproblem.

Integers: Jumping based on arithmetic operations involves the following methods: summing all the numbers provided in the answers and then applying modulus N (R_{i1}), subtracting the smallest number from the largest number and then applying modulus N (R_{i2}), summing the squares of all the numbers and then applying modulus N (R_{i3}), converting the sum directly to base N and taking the last digit (R_{i4}). Finally, the results from these operations are mapped to the N candidates. For example, a result of 3 would map to the first candidate subproblem, while a result of 0 would map to the fourth candidate subproblem.

Strings: Jumping methods for answers that are strings involve arithmetic operations based on the numeric digits within the strings: summing all the individual numeric digits (0-9) and then applying modulus N (R_{s1}), subtracting the smallest individual numeric digit from the largest and then applying modulus N (R_{s2}), summing the squares of all individual numeric digits and then applying modulus N (R_{s3}). Finally, the results from these operations are mapped correspondingly to the N candidates.

Furthermore, three different logical jumping methods are established:

Sequential Jumping: The path to the $(i+1)$-th subproblem relies on the i-th subproblem. This creates a straightforward, linear sequence where each step logically follows from the previous one.

Backtracking Jumping: The path to the $(i+1)$-th subproblem depends on any one of the earlier subproblems from 1 to $i-1$. This involves exploring and revisiting previous subproblems to find a viable solution.

Composite Jumping: The path to the $(i+1)$-th subproblem based on any two subproblems from 1 to i. It allows for a more complex and nuanced decision-making process by considering multiple prior interactions simultaneously.

3.3 Evaluation Metrics

In addition to assessing the accuracy of long-chain reasoning problems by exact string match, we also utilize the reasoning gap, which reflects how often models can correctly answer subproblems individually but not generate the overall solution. The formula for calculating the reasoning gap in the long-chain reasoning problems composed of n sub-

problems is:

$$gap = 1 - (s^*/s^n), \quad (1)$$

where s denotes accuracy on the subproblem set, and s^* denotes the actual accuracy of the long-chain reasoning problems.

3.4 Construction of Dataset

Based on the proposed pipeline, two datasets in Chinese and English are constructed, introduced in Table 1. The English dataset consists of Easy and Hard versions. The Easy subset employs an answer combination approach, with a subproblem size of 2 and a problem complexity of 1. Algorithm 1 details the construction algorithm for the Easy subset, which requires generating the string concatenation format, the order for solving all subproblems, and the collection of subproblems. These parameters are then input into the answer combination template illustrated in the left part of Fig. 1 to create a complete sample.

The Hard subset employs a problem-jumping strategy, with subproblem sizes of 2, 3, and 4, corresponding to problem complexities of 2, 3, and 4, respectively. The data construction algorithm for the Hard subset is detailed in Appendix Algorithm 2. Similar to the answer combination algorithm, a series of parameters must be generated and then input into the problem-jumping template shown in the right part of Fig. 1. Appendix Figs 6 and 7 provide detailed examples of long-chain reasoning problems with a subproblem size of 4, utilizing the problem-jumping strategy.

4 Experiments

This section focuses on two main aspects: evaluating the reasoning capabilities of various models on LREval and analyzing the potential factors influencing model performance.

4.1 Experimental Setup

We evaluate the performance of several representative instruction-tuned models, including: (i) closed-source models GPT-4o[1] (OpenAI, 2023), o1-mini[2], and Claude-3.5-sonnet[3], (ii) open-source models including LLaMA3.1, LLaMA3.2 (Dubey et al., 2024b), Qwen2.5 (Yang et al., 2024a), Mistral-Nemo, Mistral-Large, and Gemma2 series (Rivière et al., 2024). Due to the long-chain reasoning process required by the task, all models except o1-mini are evaluated using the Chain of Thought (CoT) approach (Wei et al., 2022), which involves prompting step-by-step thinking.

[1] gpt-4o-2024-11-20
[2] o1-mini-2024-09-12
[3] claude-3-5-sonnet-20241022

4.2 Evaluation on the English portion of LREval

Table 2 presents an overview of the evaluation results on the English portion of LREval. Accuracy indicates the correctness of a model on long-chain reasoning problems, with higher values being better. Reasoning gap reflects how often models can correctly answer subproblems individually but fail to generate the overall solution for the long-chain problem, with lower values being preferable. First, smaller models under 7B exhibit nearly 100% reasoning gap on the Hard subset, underscoring their limited reasoning capabilities. As model size increases, the reasoning gap narrows, suggesting that scaling up models can enhance their long-chain reasoning abilities. However, noticeable reasoning gaps still exist, e.g., the reasoning gaps of Qwen2.5-72B reach 12.40%, 36.34%, and 54.24% on the 2-hop (problems with a subproblem size of 2), 3-hop, and 4-hop hard subsets, respectively. Even the most advanced closed-source models show substantial reasoning gaps, particularly in 4-hop problems, where the smallest gap with o1-mini is as high as 27.73%. These results underscore the challenges posed by the proposed long-chain reasoning task.

Closed-source models generally outperform open-source models on the Hard subset. However, on the Easy subset, closed-source models perform even worse than open-source models with 70B parameters or more. This indicates that closed-source models excel in reasoning tasks that require flexible adaptation and dynamic decision-making, whereas open-source models have a slight advantage in information integration and structured reasoning. This performance disparity suggests underlying differences in the training methodologies and architecture designs between closed-source and open-source models. Closed-source models likely benefit from proprietary enhancements and access to diverse, high-quality datasets, which equip them with robust capabilities for handling complex, high-stakes reasoning tasks that necessitate flexibility and adaptive problem-solving strategies. On the other hand, open-source models might be leveraging broader community-driven advancements and collaborative fine-tuning techniques, which enhance their ability to integrate information and

Model	Reasoning Gap ↓ (%)				Accuracy ↑ (%)			
	Easy	Hard			Easy	Hard		
	2-hop	2-hop	3-hop	4-hop	2-hop	2-hop	3-hop	4-hop
Open-source models (<7B)								
LLaMA3.2-1B	88.48	100.00	100.00	100.00	0.60	0.00	0.00	0.00
LLaMA3.2-3B	77.60	100.00	100.00	100.00	1.10	0.00	0.00	0.00
Qwen2.5-0.5B	98.16	100.00	100.00	100.00	0.10	0.00	0.00	0.00
Qwen2.5-1.5B	61.05	97.30	100.00	100.00	10.10	0.70	0.00	0.00
Qwen2.5-3B	**36.26**	**92.68**	**95.65**	**98.89**	**27.00**	**3.10**	**1.20**	**0.20**
Gemma-2-2B	48.47	98.39	100.00	100.00	9.60	0.30	0.00	0.00
Open-source models (7B)								
LLaMA3.1-8B	21.41	56.08	76.42	89.15	27.20	15.20	4.80	1.30
Gemma-2-9B	**18.35**	82.22	**61.52**	**84.02**	30.30	6.60	8.70	2.20
Qwen2.5-7B	21.35	**54.26**	76.42	88.93	**42.30**	**24.60**	**9.30**	**3.20**
Open-source models (14B)								
Mistral-nemo-2407	14.79	37.66	61.20	83.32	27.20	19.90	7.00	1.70
Qwen2.5-14B	**10.58**	**36.71**	**48.94**	**59.44**	**55.10**	**39.00**	**24.70**	**15.40**
Open-source models (34B)								
Gemma-2-27B	12.80	61.50	53.74	75.26	37.60	16.60	13.10	4.60
Qwen2.5-32B	**10.03**	**39.81**	**46.33**	**71.66**	**58.30**	**39.00**	**28.00**	**11.90**
Open-source models (≥70B)								
LLaMA3.1-70B	9.79	57.15	57.02	76.44	45.90	21.80	15.60	6.10
Qwen2.5-72B	**5.87**	**12.40**	**36.34**	54.24	**60.50**	**56.30**	**32.80**	**18.90**
Mistral-large-2407	6.17	21.45	38.30	**49.58**	52.20	43.70	25.60	15.60
Open-source reasoning models								
QwQ-32B-Preview	-4.44	39.82	28.28	40.36	63.00	36.30	33.60	21.70
Closed-source models								
GPT-4o	**8.36**	11.62	30.53	44.04	53.40	51.50	30.90	19.00
o1-mini	14.21	**7.67**	**19.09**	**27.73**	**62.71**	**67.49**	**50.57**	**38.62**
Claude-3.5	15.94	19.12	34.70	53.61	52.80	50.80	32.50	18.30

Table 2: Evaluation results of different models on the English portion of LREval. 4-hop indicates that the subproblem size is equal to 4. Higher accuracy is preferable, while a smaller reasoning gap is desirable.

perform structured, systematic reasoning.

4.3 Analyses

In this section, we first investigate model performance on the jumping rules designed in this paper. Then, we explore various factors that may influence model performance.

4.3.1 Analysis of the Designed Jumping Rules

We directly evaluate the accuracy of all tested models on each jump rule, shown in Appendix Table 4. Detailed prompts can be found in Appendix Fig. 8. The input texts are derived from the answers to the subproblems within the subproblem set. For cases where the preceding answer is an option, there is only one rule, i.e., direct jumping (R_{o1}). For cases where the preceding answer is an integer, there are four rules including R_{i1}, R_{i2}, R_{i3}, and R_{i4}. For cases where the preceding answer is a string, there are three rules: R_{s1}, R_{s2}, and R_{s3}. Detailed jumping rules can be found in subsection 3.2.3.

Additionally, three sixth-grade student annotators are engaged to establish a human baseline for each jumping rule. Detailed guidelines are introduced to ensure consistency and clarity before the evaluation begins. To manage costs, 10% of samples are sampled for each jumping rule. Remarkably, all annotators achieve 100% accuracy across all rules, demonstrating the designed jumping rules are trivial for humans to follow. However, small models smaller than 7B exhibit low accuracy on these jump rules, with results as low as 11.25%. This indicates a severe deficiency in the context understanding and reasoning capabilities of small models. In contrast, closed-source models and open-source models ($\geq 70B$) achieve over 96.50% accuracy on these jump rules, indicating that these rules are equally straightforward for larger models. Notably, QwQ-32B-Preview reaches a lower average accuracy than Qwen2.5-32B, primarily due to a high failure rate in parsing its predicted answers.

4.3.2 Exploring the Factors Affecting Model Performance

Due to the reasoning gap of models smaller than 7B being nearly 100%, our analysis focuses exclusively on models with 7B parameters and above. Since some factors do not simultaneously appear in both the problem-jumping and answer combination strategies, they are analyzed only within

	2-Cands	4-Cands	6-Cands	10-Cands	20-Cands
LLaMA3.1-8B	33.78	24.27	30.26	29.56	43.65
Gemma-2-9B	20.81	14.67	27.26	32.17	61.33
Qwen2.5-7B	16.57	33.90	34.49	39.48	57.11
Avg	23.72	24.28	30.67	33.74	54.03
Mistral-Nemo	27.32	33.35	35.57	44.14	54.30
Qwen2.5-14B	19.83	26.37	14.92	19.01	28.01
Avg	23.57	29.86	25.25	31.58	41.15
Gemma-2-27B	9.85	10.45	13.49	18.04	24.72
Qwen2.5-32B	11.30	34.08	29.47	35.44	29.47
Avg	10.57	22.27	21.48	26.74	27.10
LLaMA3.1-70B	4.71	4.71	4.99	9.70	14.68
Qwen2.5-72B	0.74	11.92	11.38	12.19	10.01
Mistral-Large	10.80	13.02	12.47	4.99	1.11
Avg	5.42	9.88	9.61	8.96	8.60

Table 3: This table illustrates the reasoning gap (%) of the tested models when addressing candidate subproblems of varying scales.

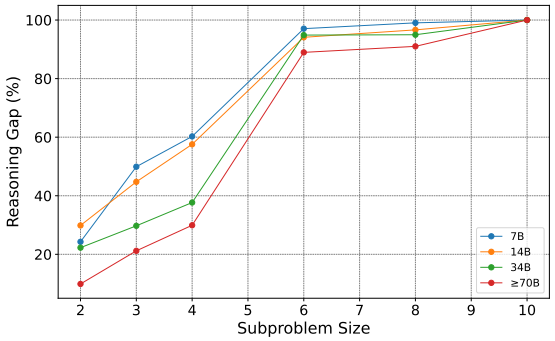

Figure 2: This figure illustrates the trend of the reasoning gap as the subproblem size increases within the problem-jumping setting.

the strategy where they are present. Unless stated otherwise, each analysis is based on 400 samples.

We first explore how the size of candidate subproblems affects model performance, using subproblems sourced from the GSM8K portion in the subproblem set, with a subproblem size of 2. As shown in Table 3, increasing the candidate subproblem size from 2 to 4 results in a widening reasoning gap among the models. When the size is further increased to 20, models smaller than 70B experience an ongoing rise in the reasoning gap. In contrast, models with 70B parameters or more show minimal change, highlighting their enhanced robustness.

Second, we analyze the impact of different logical jumping methods on model performance, utilizing subproblems from the GSM8K portion with a subproblem size of 4. As shown in columns two to four of Appendix Table 6, the backtracking jumping method results in the largest reasoning gap. In this scenario, the model often overlooks the backtracking conditions and continues to follow a sequential jumping approach. This suggests that the model lacks adequate dynamic decision-making capabilities.

Third, we examine how increasing the subproblem size affects model performance, using subproblems derived from the GSM8K portion. Fig. 2 illustrates the results utilizing the problem-jumping strategy. The reasoning gap gradually increases as the subproblem size expands from 2 to 4. However, when the size increases from 4 to 6, the reasoning gap jumps sharply to nearly 90%. Extending the size to 10 results in the reasoning gap reaching 100%. A similar trend is evident in Appendix Fig. 5, which shows results utilizing the answer-combination strategy. This substantial increase beyond a certain threshold suggests that the complexity of long-chain reasoning tasks exhibits non-linear characteristics.

Fourth, we explore the impact of various string concatenation formats including "====" (C_1), "####" (C_2), and "****" (C_3), on model performance, utilizing subproblems from the GSM8K portion with a subproblem size of 2. Results for long-chain problems employing the problem-jumping strategy are presented in columns five to seven of Appendix Table 6. The reasoning gap varies significantly across these three formats, with the highest difference reaching 57.52%. This underscores the fragility of the models' reasoning capabilities. Notably, using "===" as a separator results in a much lower reasoning gap compared to the other formats, suggesting that it may help the model organize and integrate information from diverse sources more effectively. However, a different phenomenon emerges in columns eight to ten of Appendix Table 6, which displays results utilizing the answer-combination strategy. In this case, the reasoning gap across the three string concatenation methods is closely similar. This consistency might be attributed to the inherently lower complexity and clearer structure of the answer combination strategy, which may minimize the influence of separator choice on model reasoning.

Finally, within the answer combination strategy, we examine the impact of swapping the order of subproblems. We utilize the long-chain reasoning problems with a subproblem size of 2. The comparison of outcomes from the two configurations is shown in columns 11 to 12 of Appendix Table 6. The findings reveal that changing the order of subproblems results in variations in model perfor-

mance, underscoring the models' fragile reasoning capabilities. Specifically, when the math problem (GSM8K) precedes the multiple-choice question (LogiQA), denoted as "G+L", the reasoning gap is larger for Qwen, Mistral, and Gemma models. In contrast, when the math problem follows the multiple-choice question, denoted as "L+G", the reasoning gap is greater for LLaMA3.1. This discrepancy likely stems from the differing training data and strategies employed by each model, highlighting how model-specific characteristics can influence reasoning performance when presented with varied problem sequences.

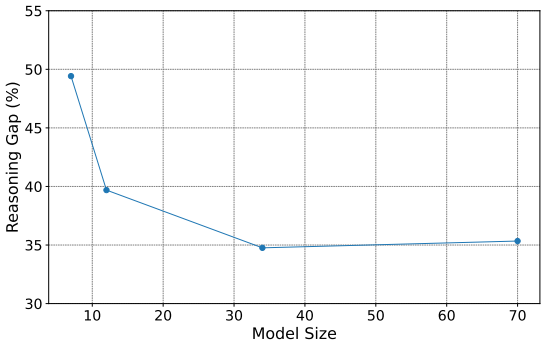

Figure 3: This figure illustrates the reasoning gap on the long-chain problems solved in the order of language sequence.

5 Multilingual Evaluation

This section presents the evaluation and analyses on the Chinese portion of LREval shown in Appendix Table 5. Consistent with the results on English data shown in Table 2, the model's long-chain reasoning capability is insufficient.

Additionally, we examine the multilingual understanding capabilities of the models using the MGSM and MLogiQA datasets. These datasets are sourced from the P-MMEval benchmark (Zhang et al., 2024a) and feature test data in ten languages including English, Chinese, Arabic, Spanish, Japanese, Korean, Thai, French, Portuguese, and Vietnamese. Specifically, for each English problem in MGSM or MLogiQA, we retrieve parallel problems in nine other languages to construct a long-chain reasoning problem, employing an answer concatenation strategy. The English subproblem is prompted to be solved first. An example of the evaluation prompt is shown in Appendix Fig. 8. The reasoning gap is measured by how often models can correctly answer the English subproblems but not generate the overall solution. We report the average reasoning gap for models of similar sizes, as categorized in Table 5. As shown in Appendix Fig. 4, even large models with 70B parameters or more still exhibit an average reasoning gap of approximately 10%. This result demonstrates that although LLMs exhibit strong multilingual capabilities when handling questions in different languages individually, their performance remains inadequate in scenarios that require simultaneous comprehension of multiple languages.

To intuitively assess the model's multilingual understanding capabilities, we combine all test data from ten different languages in MGSM and MLogiQA into a multilingual subproblem set. For each instance, we randomly select three subproblems from this set to form a long-chain problem using the answer combination strategy, producing a total of 400 samples. The problems are prompted to be solved sequentially by language, such as solving the problem in Chinese first, followed by English, and then Korean. An example of the evaluation prompt is shown in Appendix Fig. 8. We also report the average reasoning gap for models of similar sizes, adhering to the model size classification method detailed in Table 5. The result presented in Fig. 3 reveals a significant reasoning gap for all tested models. For example, the average reasoning gap for models with 70B parameters or more reaches 35.34%. These findings highlight the model's shortcomings in multilingual comprehension.

6 Conclusion

In this paper, we introduce a pipeline that leverages existing reasoning problems to create more complex long-chain reasoning tasks. Then, we introduce a new benchmark, LREval, including both Easy and Hard versions. The former emphasizes the model's proficiency in information integration and structured reasoning, while the latter primarily examines the model's capabilities in dynamic decision-making and flexible adaptation. Furthermore, we conduct extensive experiments on representative LLMs, including both general and reasoning-focused models. Our evaluations uncover huge reasoning challenges and failures faced by current LLMs, underscoring the fragility of their reasoning capabilities and emphasizing the need for robust evaluation frameworks.

Limitations

Through the above experiments and analyses, we summarize the following limitations:

1) Scope of Reasoning Tasks: While LREval introduces a novel framework for assessing the reasoning capabilities in LLMs, the dataset may not encompass the full spectrum of reasoning skills required in various real-world applications. Future work should focus on expanding the answer combination strategy to comprehensively evaluate the performance and adaptability of LLMs across a wider range of contexts.

2) Limited problem jump rules: The current set of problem-jumping rules in LREval is somewhat narrow. By introducing a greater variety of jumping rules that mimic more complex, real-world problem structures, future iterations of the dataset could offer a more comprehensive challenge to LLMs, better evaluating their capabilities in handling intricate reasoning problems.

Ethics Statement

All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki Declaration and its later amendments or comparable ethical standards. This article does not contain any studies with animals performed by any of the authors. Informed consent was obtained from all individual participants included in the study.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv preprint arXiv:abs/2309.16609*.

Neeladri Bhuiya, Viktor Schlegel, and Stefan Winkler. 2024. Seemingly plausible distractors in multi-hop reasoning: Are large language models attentive readers? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 2514–2528. 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. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2024. A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.*, 15(3):39:1–39:45.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. 2023. Theoremqa: A theorem-driven question answering dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 7889–7901. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan

Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024a. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024b. The llama 3 herd of models. *arXiv preprint arXiv:abs/2407.21783*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Arian Hosseini, Alessandro Sordoni, Daniel Toyama, Aaron C. Courville, and Rishabh Agarwal. 2024. Not all LLM reasoners are created equal. *arXiv preprint arXiv:2410.01748*.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1049–1065. Association for Computational Linguistics.

Jon A Krosnick, Stanford University, Stanley Presser, and Art Sociology Building. 2010. Question and questionnaire design. *Handbook of Survey Research*.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3622–3628. ijcai.org.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 5687–5711. Association for Computational Linguistics.

Morgane Rivière, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozinska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucinska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju-yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjösund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, and Lilly McNealus. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:abs/2408.00118*.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2023. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2022. Large language models still can't plan (a benchmark for llms on planning and reasoning about change). In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Xiaobao Wu, Liangming Pan, Yuxi Xie, Ruiwen Zhou, Shuai Zhao, Yubo Ma, Mingzhe Du, Rui Mao, Anh Tuan Luu, and William Yang Wang. 2024. Antileak-bench: Preventing data contamination by automatically constructing benchmarks with updated real-world knowledge. *arXiv preprint arXiv:2412.13670*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024a. Qwen2 technical report. *arXiv preprint arXiv:abs/2407.10671*.

Bo Yang, Qingping Yang, and Runtao Liu. 2024b. Utmath: Math evaluation with unit test via reasoning-to-coding thoughts. *arXiv preprint arXiv:2411.07240*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. Association for Computational Linguistics.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. 2023a. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. 2023b. Evaluating the performance of large language models on GAOKAO benchmark. *arXiv preprint arXiv:2305.12474*.

Yidan Zhang, Yu Wan, Boyi Deng, Baosong Yang, Haoran Wei, Fei Huang, Bowen Yu, Junyang Lin, and Jingren Zhou. 2024a. P-mmeval: A parallel multilingual multitask benchmark for consistent evaluation of llms. *arXiv preprint arXiv:2411.09116*.

Yidan Zhang, Mingfeng Xue, Dayiheng Liu, and Zhenan He. 2024b. Rationales for answers to simple math word problems confuse large language models. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 8853–8869. Association for Computational Linguistics.

A Model Performance on the Designed Jumping Rules

Table 4 presents model accuracy on the designed jumping rules.

B Multilingual Evaluation

Table 5 presents the evaluation results on the Chinese portion of LREval. Fig. 4 illustrates the reasoning gap on the long-chain problems composed of 10 parallel subproblems.

C Investigation of the Factors Affecting Model Performance

Table 6 shows how the three factors impact model performance. Fig. 5 shows the trend of the reasoning gap as the subproblem size increases within the answer combination setting.

D Algorithm for Data Constructing

This section introduces the algorithm for constructing the Hard subset, with each set comprising four subproblems, as detailed in Algorithm 2.

Model	R_{o1}	R_{i1}	R_{i2}	R_{i3}	R_{i4}	R_{s1}	R_{s2}	R_{s3}	Avg
Open-source models (<7B)									
LLaMA3.2-1B	21.50	32.25	12.25	19.00	24.75	19.50	17.75	17.50	20.56
LLaMA3.2-3B	25.25	29.75	11.25	18.00	20.25	22.25	17.50	17.25	20.19
Qwen2.5-0.5B	30.50	38.50	28.25	26.25	25.25	17.50	20.00	20.75	25.88
Qwen2.5-1.5B	74.00	78.75	57.50	64.50	82.75	62.25	60.25	56.50	67.06
Qwen2.5-3B	90.50	94.75	90.00	88.25	88.75	85.00	82.25	78.00	**87.19**
Gemma-2-2B	38.50	77.00	67.75	58.25	88.50	66.50	62.75	69.00	66.03
Open-source models (7B)									
LLaMA3.1-8B	96.25	99.25	90.25	96.75	99.00	86.25	81.25	82.50	91.44
Gemma-2-9B	97.25	99.75	99.50	90.75	98.75	91.50	94.00	93.50	95.63
Qwen2.5-7B	99.75	99.50	98.25	99.50	98.50	89.75	97.00	90.50	**96.59**
Open-source models (14B)									
Mistral-Nemo	98.25	96.00	81.50	88.25	93.50	87.00	88.75	86.50	89.97
Qwen2.5-14B	100.00	97.00	95.00	98.50	98.25	95.25	97.00	93.75	**96.84**
Open-source models (34B)									
Gemma-2-27B	100.00	99.75	99.75	99.75	100.00	97.75	98.50	98.50	**99.25**
Qwen2.5-32B	100.00	99.75	99.75	99.50	100.00	96.75	99.25	96.75	98.97
Open-source models (≥70B)									
LLaMA3.1-70B	99.75	99.50	99.00	99.75	100.00	96.50	99.50	98.00	99.00
Qwen2.5-72B	100.00	99.75	99.75	100.00	100.00	96.75	99.00	97.00	99.03
Mistral-Large	100.00	100.00	100.00	99.75	99.75	97.50	99.25	96.75	**99.13**
Open-source reasoning models									
QwQ-32B-Preview	84.75	93.00	90.00	95.00	93.00	90.00	78.25	83.00	88.38
Closed-source models									
GPT-4o	100.00	100.00	100.00	100.00	100.00	99.00	100.00	99.00	99.75
o1-mini	100.00	100.00	100.00	100.00	100.00	99.00	100.00	100.00	**99.88**
Claude-3.5	100.00	100.00	100.00	100.00	100.00	100.00	100.00	99.00	**99.88**
Human	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Table 4: This table presents model accuracy (%) on the designed jumping rules. The third column shows the accuracy of direct jumping utilizing the jumping rule of R_{i1}.

Model	Reasoning Gap ↓ (%)		Accuracy ↑ (%)	
	Easy	Hard	Easy	Hard
Open-source models (<7B)				
LLaMA3.2-1B	100.00	100.00	0.00	0.00
LLaMA3.2-3B	84.38	100.00	0.60	0.00
Qwen2.5-0.5B	100.00	100.00	0.00	0.00
Qwen2.5-1.5B	46.60	91.10	12.00	2.00
Qwen2.5-3B	17.23	87.61	29.40	4.40
Gemma-2-2B	51.92	98.86	8.40	0.20
Open-source models (7B)				
LLaMA3.1-8B	9.17	53.72	31.40	16.00
Gemma-2-9B	14.15	73.47	35.60	11.00
Qwen2.5-7B	5.70	57.39	47.80	21.60
Open-source models (14B)				
Mistral-Nemo	16.26	47.27	32.40	20.40
Qwen2.5-14B	2.50	34.38	62.40	42.00
Open-source models (34B)				
Gemma-2-27B	6.10	55.86	46.80	22.00
Qwen2.5-32B	0.88	29.36	69.60	49.60
Open-source models (≥70B)				
LLaMA3.1-70B	-1.54	41.10	56.20	32.60
Qwen2.5-72B	2.13	13.01	68.40	60.80
Mistral-Large	0.21	17.11	62.60	52.00

Table 5: Evaluation results of different models on the Chinese portion of LREval.

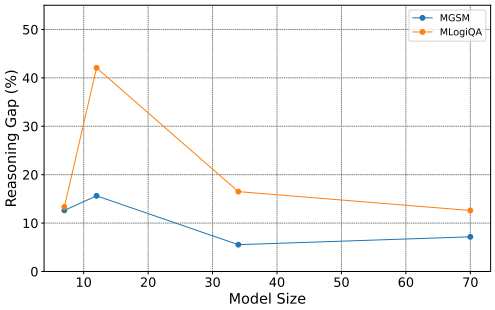

Figure 4: This figure illustrates the reasoning gap on the long-chain problems composed of 10 parallel sub-problems, employing the answer combination strategy. The line labeled "MGSM" corresponds to all 10 parallel samples derived from the MGSM dataset.

E Example Prompts

Fig. 8 illustrates several prompts. Fig. 6 illustrates backtracking jumping, while Fig. 7 depicts composite jumping.

Model	Problem Jumping						Answer Combination				
	Logical Jumping Way			String Concatenation Format			String Concatenation Format			Subproblem Order	
	Sequential	Backtracking	Composite	C_1	C_2	C_3	C_1	C_2	C_3	G+L	L+G
LLaMA3.1-8B	56.33	89.58	63.78	24.27	25.33	67.24	9.13	11.24	8.78	1.69	31.88
Gemma-2-9B	59.30	76.26	64.20	14.67	99.69	100.00	11.91	8.53	10.07	12.50	9.66
Qwen2.5-7B	65.13	82.74	76.53	33.90	99.12	33.31	8.34	8.34	9.22	18.58	15.69
Avg	60.26	82.86	68.17	24.28	74.71	66.85	9.79	9.37	9.36	10.92	19.08
Mistral-Nemo	67.76	81.87	81.87	33.35	54.93	30.18	18.43	20.97	18.43	15.01	3.43
Qwen2.5-14B	47.35	68.47	57.76	26.37	74.09	40.28	3.47	4.01	3.74	14.55	7.94
Avg	57.56	75.17	69.81	29.86	64.51	35.23	10.95	12.49	11.09	14.78	5.68
Gemma-2-27B	30.70	69.04	43.97	10.45	70.86	100.00	2.87	2.56	5.90	1.44	3.46
Qwen2.5-32B	44.66	76.75	83.52	34.08	35.17	59.58	0.99	1.80	2.62	5.52	3.91
Avg	37.68	72.89	63.74	22.27	53.01	79.79	1.93	2.18	4.26	3.48	3.68
LLaMA3.1-70B	32.47	80.36	30.63	4.71	69.53	86.43	3.60	2.22	2.49	-0.94	4.35
Qwen2.5-72B	32.78	68.77	48.84	11.92	13.56	13.56	-0.62	1.29	-0.08	11.39	6.65
Mistral-Large	24.49	76.06	38.00	13.02	13.85	41.55	-0.83	0.00	-0.83	5.77	4.07
Avg	29.92	75.06	39.16	9.88	32.31	47.18	0.72	1.17	0.53	5.41	5.02

Table 6: This table provides a detailed exhibition of how different factors affect the reasoning gap (%) when employing problem-jumping and answer combination strategies. "G+L" signifies that, in the long-chain problem with a subproblem size of 2, the math subproblem (GSM8K) is arranged before the multiple-choice subproblem (LogiQA).

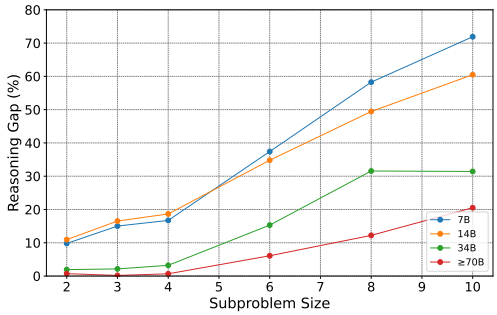

Figure 5: This figure illustrates the trend of the reasoning gap as the subproblem size increases within the answer combination setting.

Algorithm 2 Algorithm for Constructing the Hard Subset with Each Consisting of 4 Subproblems

Require: Subproblem set $\mathbf{Q}$, string concatenation format set $\mathbf{Comb} = [\#\#\#\#, ====, **]$, dataset size $\mathbf{M_H} = 1000$, subproblem size of the long-chain reasoning problem $\mathbf{S} = 4$, problem jumping template $\mathbf{T_H}$, jumping rule set $\mathbf{Jump}$, candidate subproblem size list $\mathbf{N_C} = [2, 3, 4]$

Ensure: An Hard subset $\mathbf{D_H}$ contains $\mathbf{M_H}$ long-chain reasoning problems with each consisting of $\mathbf{S}$ subproblems combined using the problem jumping strategy

Initialize with an empty list $\mathbf{D_H}$
$\mathbf{i} = 1$
while $\mathbf{i} \leq \mathbf{M_H}$ **do**
    $\mathbf{j} \leftarrow 1$
    Randomly sample one subproblem from the subproblem set $\mathbf{Q}$ without replacement to serve as the first-hop problem $\mathbf{q_1}$
    Utilize the first-hop problem $\mathbf{q_1}$ as the preceding problem $\mathbf{q_{pre}}$ for the first jump, with its answer serving as the preceding answer $\mathbf{a_{pre}}$
    If the preceding answer $\mathbf{a_{pre}}$ is an option label, the number of candidate problems must match the number of available options. Otherwise, randomly sample one number $\mathbf{n_j}$ from $\mathbf{N_C}$. Then, randomly sample $\mathbf{n_j}$ subproblems $\mathbf{C_j}$ with different answers from $\mathbf{Q}$ without replacement
    Based on the preceding answer $\mathbf{a_{pre}}$, randomly sample a problem jumping rule $\mathbf{R_j}$
    Determine the second-hop problem $\mathbf{q_2}$ from the candidate subproblems $\mathbf{C_j}$ according to the jumping rule $\mathbf{R_j}$ and the preceding answer $\mathbf{a_{pre}}$
    $\mathbf{j} \leftarrow 2$
    Randomly select one from [$\mathbf{q_1}$, $\mathbf{q_2}$] as the preceding problem to the second jump, using its answer as the preceding answer
    Repeat Steps 7 and 8
    Determine the third-hop problem from the candidate subproblems $\mathbf{C_j}$ according to the jumping rule and the preceding answer
    $\mathbf{j} \leftarrow 3$
    Check if there are two integer answers to [$\mathbf{q_1}$, $\mathbf{q_2}$, $\mathbf{q_3}$]. If so, perform a combination judgment: add the two integer answers together to form the preceding answer. Then, repeat Steps 7 and 8. Otherwise, randomly select one problem from [$\mathbf{q_1}$, $\mathbf{q_2}$, $\mathbf{q_3}$] as the preceding problem and repeat Steps 7 and 8.
    Determine the fourth-hop problem from the candidate subproblems according to the jumping rule and the preceding answer
    Sample one string concatenation format $\mathbf{c}$ from $\mathbf{Comb}$
    Construct a sample $\mathbf{p_i}$ by filling the problem jumping template $\mathbf{T_H}$ with parameters, including $[\mathbf{q_1}, \mathbf{q_2}, \mathbf{q_3}, \mathbf{q_4}]$, $\mathbf{c}$, $[\mathbf{n_1}, \mathbf{n_2}, \mathbf{n_3}]$,$[\mathbf{R_1}, \mathbf{R_2}, \mathbf{R_3}]$,$[\mathbf{C_1}, \mathbf{C_2}, \mathbf{C_3}]$, etc.
    Appending the new sample $\mathbf{p_i}$ into $\mathbf{D_H}$
    $\mathbf{i} \leftarrow \mathbf{i} + 1$
Return $\mathbf{D_H}$

The following is a multi-hop long-chain reasoning problem that requires jumping to the subsequent question based on the answer to the previous question.

Initial question: {Q0}

Square each digit in the answer to the initial question, sum these squares, and then take the result modulo 4 to obtain a key number X.
First jump: If X is equal to 2, proceed to Q1_3. If X is equal to 3, proceed to Q1_1. If X is equal to 1, proceed to Q1_0. If X is equal to 0, proceed to Q1_2.

Q1_1: {Q1}
Q1_3: {Q2}
Q1_2: {Q3}
Q1_0: {Q4}

For the answer to the first-jump question:
1. Identify and extract each individual numeric digit (0-9) present in the input, treating each digit as a separate character regardless of whether they appear together in numbers, fractions, radicals, or any complex expressions.
2. Sum all the extracted numeric digits.
3. If there are no numeric digits, return the sum as 0.
4.Take the sum modulo 2 to obtain a key number Y.
Second jump: If Y is equal to 1, proceed to Q2_1. If Y is equal to 0, proceed to Q2_0.

Q2_0: {Q5}
Q2_1: {Q6}

Sum each digit of the answer to the initial question, then take the result modulo 3 to obtain a key number Z.
Third jump: If Z is equal to 0, proceed to Q3_1. If Z is equal to 2, proceed to Q3_0. If Z is equal to 1, proceed to Q3_2.

Q3_0: {Q7}
Q3_2: {Q8}
Q3_1: {Q9}

Solve the multi-hop question. Explain your answer step by step and give the final answer on the last line by itself in the format of
'The answer of the initial question is [answer0].
The first-jump question is [question1].
The answer of the first-jump question is [answer1].
The second-jump question is [question2].
The answer of the second-jump question is [answer2].
The third-jump question is [question3].
The answer of the third-jump question is [answer3].
The final answer is \boxed{[answer0]####[answer1]####answer2]####[answer3]}.'
Where [answer0] must be integers, and the final answer must be put within \boxed{}.

Let's think step by step.

Figure 6: This figure illustrates the backtracking jumping.

The following is a multi-hop long-chain reasoning problem that requires jumping to the subsequent question based on the answer to the previous question.

Initial question: {Q0}

Let X be the answer to the initial question.
First jump: If X is equal to C, proceed to Q1_2. If X is equal to A, proceed to Q1_3. If X is equal to D, proceed to Q1_1. If X is equal to B, proceed to Q1_0.

Q1_1: {Q1}
Q1_3: {Q2}
Q1_2: {Q3}
Q1_0: {Q4}

Subtract the smallest digit from the largest digit in the answer to the initial question (ignore the negative sign for this calculation). If there are fewer than two digits, return the difference as 0. Then take the difference modulo 2 to obtain a key number Y.
Second jump: If Y is equal to 1, proceed to Q2_1. If Y is equal to 0, proceed to Q2_0.

Q2_0: {Q5}
Q2_1: {Q6}

Add the answer from the first-jump question to the answer from the second-jump question to get the sum. Then, add each digit of this sum. Finally, take the result modulo 4 to obtain a key number Z.
Third jump: If Z is equal to 0, proceed to Q3_1. If Z is equal to 2, proceed to Q3_0. If Z is equal to 1, proceed to Q3_2. If Z is equal to 3, proceed to Q3_3.

Q3_0: {Q7}
Q3_2: {Q8}
Q3_1: {Q9}
Q3_3: {Q10}

Solve the multi-hop question. Explain your answer step by step and give the final answer on the last line by itself in the format of
'The answer of the initial question is [answer0].
The first-jump question is [question1].
The answer of the first-jump question is [answer1].
The second-jump question is [question2].
The answer of the second-jump question is [answer2].
The third-jump question is [question3].
The answer of the third-jump question is [answer3].
The final answer is \boxed{[answer0]****[answer1]****answer2]****[answer3]}.'
Where [answer0] must be integers, and the final answer must be put within \boxed{}.

Let's think step by step.

Figure 7: This figure illustrates the composite jumping.

Prompt for the long-chain problem composed of three parallel subproblems

The following is a long-chain reasoning problem that requires solving several sub-problems and then combining their answers into a single formatted string.

Let's break down the problem: Answer question Q1. Answer question Q2. Answer question Q3. Concatenate the answers of Q1, Q2, and Q3 into a single string, separating them with '====='. Explain your answer step by step and give the final answer on the last line by itself in the format of

'The answer of Q1 is [answer1]. The answer of Q2 is [answer2]. The answer of Q3 is [answer3]. The final answer is \boxed{[answer1]====[answer2]====[answer3]}'.

Where [answer1], [answer2], and [answer3] must be integers, and the final answer must be put within \boxed{}.

Q1: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

Q2: Les canes de Janet pondent 16 œufs par jour. Chaque matin, elle en mange trois au petit déjeuner et en utilise quatre autres pour préparer des muffins pour ses amis. Ce qui reste, elle le vend quotidiennement au marché fermier, au prix de 2 $ l'œuf de cane frais. Combien (en dollars) gagne-t-elle chaque jour au marché fermier ?

Q3: Os patos de Janet põem 16 ovos por dia. Ela come três no café da manhã todas as manhãs e usa quatro para fazer muffins para seus amigos todos os dias. Ela vende o restante no mercado de agricultores diariamente por $2 por ovo fresco de pato. Quanto ela ganha em dólares todos os dias no mercado de agricultores?

Let's think step by step.

Prompt for the long-chain problem solved in the order of language sequence

The following is a long-chain reasoning problem that requires solving several multilingual sub-problems and then combining their answers into a single formatted string.

Let's break down the problem: Answer the question written in Korean. Answer the question written in French. Answer the question written in Chinese. Concatenate the answers of all the questions into a single string, separating them with '====='. Explain your answer step by step and give the final answer on the last line by itself in the format of

'The answer of the Korean question is [answer1]. The answer of the French question is [answer2]. The answer of the Chinese question is [answer3]. The final answer is \boxed{[answer1]====[answer2]====[answer3]}'.

Where [answer1] must be integer, [answer2] must be integer, [answer3] must be integer, and the final answer must be put within \boxed{}.

Q1: Deux trains partent de San Rafael au même moment. Ils se dirigent d'abord vers l'ouest et parcourent tous les deux 80 miles. Le jour suivant, ils voyagent vers le nord, couvrant 150 miles. Quelle est la distance parcourue par chaque train au cours de ces deux jours ?

Q2: 존은 3시간 동안 복숭아를 따왔습니다. 그는 1분에 복숭아 2개를 딸 수 있습니다. 그가 복숭아를 몇 개 따냈나요?

Q3: 弗兰基每天晚上做完作业后看电视。周一和周二，他每晚看一集自己最喜欢的节目，一集 1 小时。周三，他看了几集每集 30 分钟的节目。周四，他提早做完了作业，看了一集每集 1 小时的节目和一集每集 30 分钟的节目。周五，因为有周末，他可以熬夜，所以他看了 2 集每集 1 小时的节目。如果他一共看了 7 小时的电视，他在周三看了多少集每集 30 分钟的节目？

Let's think step by step.

Prompt for direct jumping

The following is a multi-hop problem that requires jumping to the subsequent question based on the input.

Input: 36

Subtract the smallest digit from the largest digit in the input (ignore the negative sign for this calculation). If there are fewer than two digits, return the difference as 0. Then take the difference modulo 4 to obtain a key number X.

First jump: If X is equal to 3, proceed to Q2. If X is equal to 1, proceed to Q1. If X is equal to 0, proceed to Q4. If X is equal to 2, proceed to Q3.

Solve the multi-hop question. Explain your answer step by step and give the final answer on the last line by itself in the format of

'The first-jump question is [question1]'. Where [question1] must be one of Q1, Q2, Q3, and Q4.

Let's think step by step.

Figure 8: This figure displays several example prompts.