Mathador-LM: A Dynamic Benchmark for Mathematical Reasoning on Large Language Models

Eldar Kurtic* ISTA & Neural Magic, Inc. eldar.kurtic@ist.ac.at

Amir Moeini* ISTA amir.moeini@ist.ac.at

Dan Alistarh ISTA & Neural Magic, Inc. dan.alistarh@ist.ac.at

Abstract

We introduce Mathador-LM, a new benchmark for evaluating the mathematical reasoning on large language models (LLMs), combining ruleset interpretation, planning, and problem-solving. This benchmark is inspired by the Mathador game, where the objective is to reach a target number using basic arithmetic operations on a given set of base numbers, following a simple set of rules. We show that, across leading LLMs, we obtain stable average performance while generating benchmark instances *dynamically*, following a target difficulty level. Thus, our benchmark alleviates concerns about test-set leakage into training data, an issue that often undermines popular benchmarks. Additionally, we conduct a comprehensive evaluation of both open and closed-source state-of-the-art LLMs on Mathador-LM. Our findings reveal that contemporary models struggle with Mathador-LM, scoring significantly lower than average 3rd graders. This stands in stark contrast to their strong performance on popular mathematical reasoning benchmarks. The implementation of Mathador-LM benchmark is available at github.com/IST-DASLab/Mathador-LM.

1 Introduction

The ability of large language models (LLMs) to approach non-trivial tasks involving both information retrieval and mathematical reasoning has led to significant research interest in evaluating these properties. Yet, the popularity of reasoning benchmarks, such as the often-used Grade-School Math (GSM) (Cobbe et al., 2021) or MATH (Hendrycks et al., 2021b) datasets, is leading to performance saturation (see Figure 1), and can potentially lead to training set contamination. Thus, there is a stringent need to develop new strong benchmarks to evaluate LLM reasoning.

We address this by proposing *Mathador-LM*, a new benchmark for examining the mathematical reasoning properties of LLMs. At a high level, Mathador-LM follows the popular Mathador mathematical game (Puma et al., 2023), in which a human player is given five base numbers together with a target number, and has to provide a series of calculations, each using one of the four basic arithmetic operations, which result in the target number.[1] Each base number can only be used once, and solutions are scored on the number of operations used—a "perfect" solution uses each basic operation and each base number exactly once.

We define and implement Mathador-LM following the framework for few-shot evaluation of language models (Gao et al., 2021), and evaluate leading open and closed LLMs such as Llama (Meta AI, 2024), and Qwen2 (Bai et al., 2023), as well as Claude (Anthropic, 2023) and GPT3.5/4 (Achiam et al., 2023). Our key observations are:

- *Mathador is a hard benchmark for LLMs*: state-of-the-art open and closed models score below 15% on average, which is significantly below the mean of 43.7% across 3rd-grade students in 2023 (Mathador, 2023).
- We observe clear correlations between model size and game performance, where models below 3B parameters obtain negligible accuracy, state-of-the-art models in the 7-8B range obtain scores of 5-7%, and 70-72B models reach the top scores of 10-15%, together with Claude-Opus. Remarkably, GPT4 and Claude-Haiku models both obtain scores below 7%.
- We introduce a notion of difficulty to the classic Mathador game (Puma et al., 2023) to enhance the benchmark's robustness and future-

*Equal contribution

[1]Our game formulation follows the mathematical game organized in France for students between the 3rd and 8th grades, to which more than 10'000 pupils participated in 2023.

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 17020–17027
November 12-16, 2024 ©2024 Association for Computational Linguistics

proof it. We then perform a detailed analysis of the performance breakdown and failure modes at the base difficulty level, which aligns with the one used for human evaluation.

- Importantly, Mathador-LM has the property that model performance is *stable across randomly-generated problem instances of the same difficulty*. Thus, we can generate one-time *dynamic* instances of similar difficulty, preventing "over-fitting."

Our results are especially relevant in the context of recent work by Yang et al. (2023) and Gunasekar et al. (2023) raising concerns about contamination across popular benchmarks used to evaluate the performance of LLMs. Their findings span three different axes: 1) existing decontamination techniques often fail to identify problematic samples, 2) synthetic data generated by closed-source models (e.g., GPT-3.5/4 (Achiam et al., 2023)) exhibits subtle test-set contamination, and 3) popular open-source datasets (e.g., RedPajama (Together, 2023), StarCoder (Li et al., 2023), The Stack (Kocetkov et al., 2022), FLAN CoT (Longpre et al., 2023)) are also contaminated to varying degrees, ranging from 0.5% to 19% (Yang et al., 2023). This evidence, together with the fact that performance on the few standard benchmarks (Cobbe et al., 2021; Hendrycks et al., 2021b) for mathematical reasoning is rapidly saturating[2], as described in Figure 1, necessitates enhancing our existing evaluation protocols and significantly improving the decontamination of existing datasets with static benchmarks.

We propose an alternative pathway towards reliable examination of LLM performance via *dynamic, one-time benchmarks* that mitigate contamination by being created *on-the-fly, independently* for each evaluation run. Mathador-LM satisfies these properties: given its nature, the benchmark can be programmatically generated and verified, making it ideally suited for fresh, one-time evaluations of LLMs. This approach mitigates issues such as test-set leakage into training data and provides a reliable method to evaluate closed-source models, even in the absence of detailed information about their training data. Moreover, results reveal interesting trends across different model families and sizes, and allowing to isolate model proficiency across instruction-following, mathematical reasoning, planning, and combinatorial search.

[2]For instance, the best achieved accuracy on GSM at the time of writing is already of 97.1% (Zhong et al., 2024).

2 The Mathador-LM Benchmark

The informal definition of the Mathador-LM game we use is provided in Figure 2, which coincides with the prompt we provide to the LLM in the default version of the game. In Table 1 we present the scoring system for the benchmark. An example instance of the benchmark is provided in Figure 3, together with basic and "optimal" solutions.

Formal Definition. Given a set of operands $A = \{a_i \in \mathbb{N} | 1 \leq i \leq 5\}$ and target value $t \in \mathbb{N}$, let $P \in \{S! | S \in \mathcal{P}(A)\}$ be a permutation of a subset of operands and define the set of expressions

$$\mathcal{E}_P = \left\{(P^c, O) | P^c \in C(P), O \in \{+, \times, -, \div\}^{|P|}\right\}$$

where $C(P)$ is the set of all legal *parenthesization* of P. Consequently the set of all expressions $\mathcal{E} = \bigcup_P \mathcal{E}_P$. Each expression $E \in \mathcal{E}$ has the value $\mathrm{val}(E)$ which is derived by associating the ith opening parenthesis in P^c with the operator O_i. Given the score function $s : \mathcal{E} \to \mathbb{N}$ we are looking for $E^* = \mathrm{argmax}_{E \in \mathcal{E}}\, s(E)$ s.t. $\mathrm{val}(E) = t$.

Each expression E can be represented in an expanded form $\mathrm{repr}(E)$ by writing the evaluation of each parenthesis when both of its nested values have been evaluated. For instance, $\mathrm{repr}(E)$ of $E = \Big(((17, ((8, 4), 11)), 2), (\times, \div, -, +)\Big)$ is the Mathador solution illustrated in Figure 3. In Mathador-LM we use $\mathrm{repr}(E)$ as the representation since it is more human-readable and Table 1 for scoring. The *accuracy* of expression E is defined as $s(E)/s(E^*)$.

Difficulty Measure. For a specific set of operands, $E_t = \{E \in \mathcal{E} | \mathrm{val}(E) = t, s(E) > 0\}$ is the set of all *solutions* for target t. We define the difficulty measure of target t as $\sum_{E \in E_t} s(E)/|E_t|^2$, following the intuition that instances with few but higher-scoring solutions are harder.

3 Model Evaluations

Evaluation Setup. A dataset of Mathador-LM problems is generated for each model evaluation by sampling the operand dataset A based on the official rules (Puma et al., 2023) and then sampling from possible targets $\{t | \exists E \in \mathcal{E}$ s.t. $\mathrm{val}(E) = t\}$ based on the desired difficulty distribution. The restrictions for base numbers are closely following the official rules of the human game, so that results we obtain with LLMs are directly comparable to human results. This means that base numbers are sampled as integers, uniformly at random from the following ranges: $n_1 \in [1, 4]$, $n_2 \in [1, 6]$,

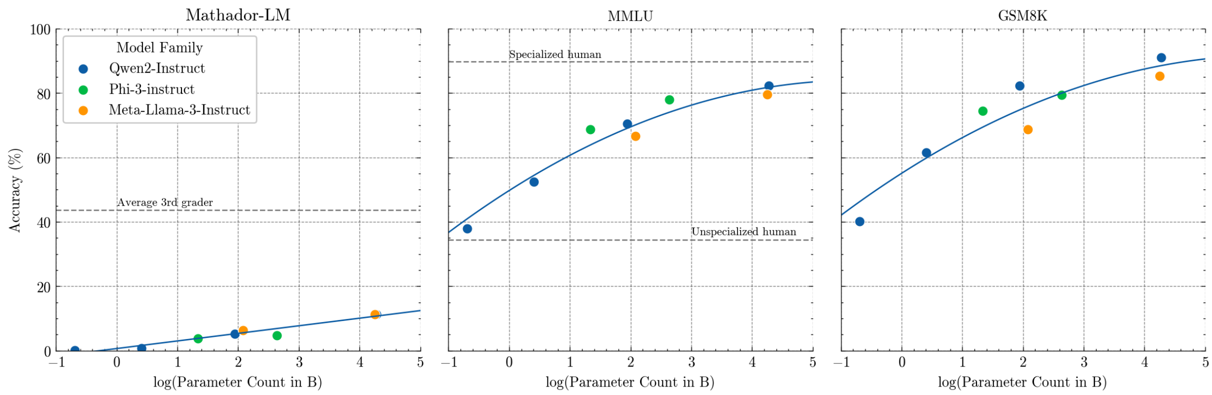

Figure 1: Comparative results on Mathador-LM, MMLU, and GSM8k, across the Llama-3-Instruct (8B and 70B), Phi-3-Instruct (small and medium), and Qwen2-Instruct model families. Interpolation lines show very high scores and clear saturation on MMLU and GSM8k at or beyond the level of specialized humans, whereas on Mathador-LM contemporary models are significantly below the average 3rd grader. MMLU and GSM8K results are obtained from Beeching et al. (2023), Hendrycks et al. (2021a), and Bai et al. (2023).

```
Game description:
In the Mathador game, players use the given base numbers,
and the operations of addition, subtraction, multiplication,
and division to reach a specified target number.

Scoring:
  - Each use of addition (+) is worth 1 point.
  - Each use of multiplication (*) is worth 1 point.
  - Each use of subtraction (-) is worth 2 points.
  - Each use of division (/) is worth 3 points.
  - 6 bonus points are awarded for using all four
    operations exactly once.

Rules:
  - You should reach the target number.
  - You should only use the base and intermediate numbers.
  - You shouldn't use a base or intermediate number more
    than once in later steps.
  - You should only produce nonnegative and integer
    intermediate results.
  - Your solution should be 4 lines at most.

Only the solution you write at the end will be considered
for scoring. Find the highest scoring solution. If you are
not able to find it, find a simple solution to earn at least
some points.

{shots}

Target number: {target}
Base numbers: {base_numbers}
```

Figure 2: The prompt for Mathador-LM benchmark.

```
Target number: 34
Base numbers: 4, 2, 8, 11, 17
-------------------------------------------------
Simple solution:       Best (Mathador) solution:
 2 x 17 = 34            8 + 4 = 12
-> Score: 6 points      12 - 11 = 1
                        17 / 1 = 17
                        17 x 2 = 34
                       -> Score: 18 points
```

Figure 3: An example problem demonstrating both simple and best (Mathador) solutions.

$n_3 \in [1, 8]$, $n_4 \in [1, 12]$, and $n_5 \in [1, 20]$. The prompt in Figure 2 is populated based on a newly generated problem set to get the final prompt. The model's generated answer to the prompt is parsed to get the solution block which is then scored. In addition to the prompt shown in Figure 2 we have experimented with everything from concise descriptions to extensive, detailed explanations. In terms of few-shot prompts, we have also tried quite a few approaches to present the model with step-by-step solutions. Some prompts were designed and tested following an error analysis that helped identify the most common mistakes made by the models. For instance, we developed specific prompts after observing that the majority of errors involved the use of illegal operands. These prompts explicitly specify the set of permissible operands at each step. Unfortunately, despite these additional instructions, we have not seen any noticeable accuracy gains.

Figure 4 presents evaluations on several popular open and closed models. We observe that small models ($\leq 3B$) and Mistral-7B tend to per-

Table 1: Scoring system for Mathador-LM benchmark. The Mathador Bonus refers to the optimal solution, achieved by using all five base numbers and each of the four operators exactly once.

Category	**Points**
Target number reached	5 points
Operators	
Addition	1 point
Multiplication	1 point
Subtraction	2 points
Division	3 points
Mathador Bonus	6 points
Invalid Solutions	
Target number not reached	0 points
Reuse of numbers	0 points
Negative numbers	0 points
Non-integer numbers	0 points

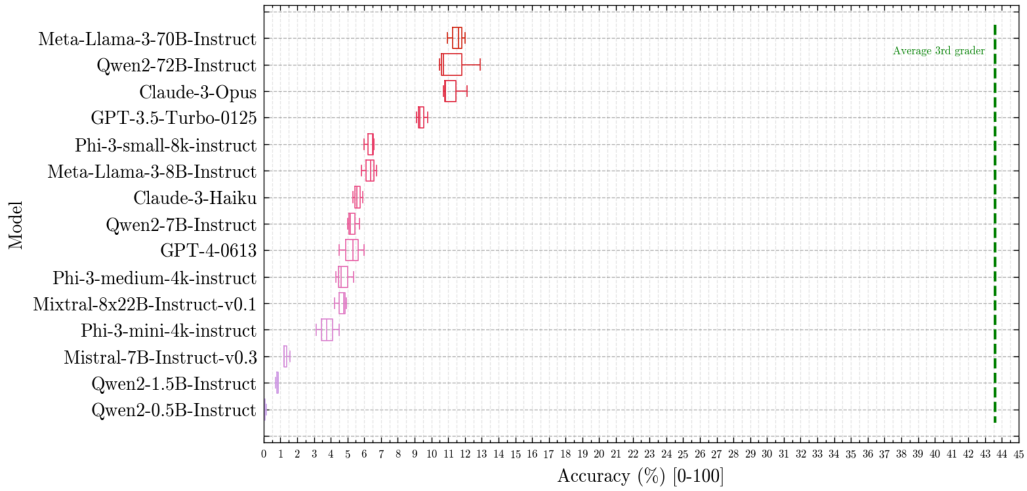

Figure 4: Detailed results on Mathador-LM across open and closed models, including confidence intervals.

form below $< 2\%$ average accuracy (0.36 points per instance, on average), meaning that they reach a correct solution (worth $\geq$ 6 points) less than 6% of the time. Surprisingly, well-performing medium models such as Qwen2-7B, Llama-3-8B, and Phi-3-medium perform on par with GPT 3.5 and GPT4, as well as Claude-Haiku (5 to 7%), at a level corresponding to reaching a correct solution less than 20% of the time. Further, we observe a higher tier for 70B models and Claude-Opus, which reach similar $\sim 12\%$ performance. In Appendix A we expand our analysis, and detail the score distribution across models. In Appendix B we investigate how allowing multiple attempts to solve each question affects the accuracy of LLMs.

Stability. A reliable benchmark must be reproducible, which is why most benchmarks are *static*. Table 2 shows that we can obtain consistent scores on Mathador-LM even when we *dynamically regenerate* the benchmark, by sampling instances with a similar difficulty mix. The *easy*, *medium*, and *hard* datasets are taken from the beginning, middle, and end of the sorted list of targets, based on difficulty (see Section 2). The *mixed* dataset contains equal fractions from each type.

Impact of Number of Shots. We investigate whether increasing the number of "shots" in the few-shot evaluation setup helps performance on Mathador-LM, as few-shot prompting (Brown et al., 2020) is known to enhance in-context learning abilities of LLMs (Wei et al., 2022). We report results in Table 3. Surprisingly, for Mathador-LM, we found that two shots are sufficient to grasp the formatting and evaluation flow. Further increasing of this number only marginally improves results.

Table 2: Stability across 5 evaluations of LLama-3-70B-Instruct on datasets of varying sizes and difficulties. Observe that the performance on the standard "mixed" benchmark is very stable across number of samples.

# Samples	Difficulty	Accuracy (%)
100	mixed	12.3 ± 1.7
250	mixed	11.8 ± 1.1
500	mixed	11.5 ± 0.5
1000	easy	15.1 ± 0.8
	medium	12.1 ± 0.6
	hard	4.3 ± 0.2
	mixed	11.3 ± 0.5
1500	mixed	12.0 ± 0.5

In Appendix C we further explore how the results are affected by different text-generation (decoding) strategies, such as greedy (Radford et al., 2019) and nucleus sampling (Holtzman et al., 2019).

Table 3: Impact of the number of shots on the evaluation of Llama-3-70B-Instruct on Mathador-LM.

# shots	2	5	10	20
Accuracy (%)	13.1 ± 0.6	13.9 ± 0.7	14.25 ± 0.6	14.34 ± 0.9

Errors Analysis. In Table 4 we present a breakdown of the errors that LLMs make when evaluated on Mathador-LM benchmark, categorized into four types: Formatting, Calculation, Missed Target, and Illegal Operand. Formatting errors occur when the model fails to adhere to the expected format for the intermediate steps. Calculation errors happen when the model makes mistakes in basic arithmetic operations while generating a solution. Missed Target refers to cases where the model correctly follows the expected format and performs all calculations accurately but arrives at a different target number than the expected solution. Illegal Operand errors arise when the model uses a number not included in the set of allowed values to produce the final solution. This can happen if a number is either not part of the base numbers provided or has already been used in previous steps.

The results in Table 4 highlight that the most significant challenges faced by the model are related to the use of illegal operands, which collectively make up over 60% of the errors. This indicates that existing models still struggle even with moderate reasoning abilities. (This complements the recent findings of Nezhurina et al. (2024).) To address the most common error made by LLMs (Illegal Operand), we augmented our prompting strategy to explicitly show the model the set of allowed operands at each step of the calculation process. Surprisingly, this *did not* improve results.

Table 4: Error types of instruction-following models on Mathador-LM, in percentages.

	Formatting Error	Calculation Error	Missed Target	Illegal Operand
Qwen2-7B	5.5	20.9	6.8	66.8
Llama-3-8B	0.3	17.3	7.1	75.3
Llama-3-70B	0.9	3.1	32.5	63.5

4 Limitations

We introduced a new challenging LLM mathematical reasoning benchmark. Our benchmark is dynamic, as it can be generated on-the-fly, mitigating the risks of test-set leakage and overfitting. The current setup can be easily extended to vary difficulty levels by, for example, adjusting the ranges of base numbers, or the total number of operands.

By design, Mathador-LM is limited to a search-based mathematical task, which has been linked to both conceptual and procedural skills (Puma et al., 2023). Another limitation we plan to investigate in future work are more advanced prompting techniques, which might alleviate the relatively low LLM performance on this task. Additionally, we plan to explore supervised fine-tuning strategies.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2023. Claude: Conversational Language Understanding AI. Anthropic Website.

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:2309.16609*.

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. 2023. Open llm leaderboard. `https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard`.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. 2021. A framework for few-shot language model evaluation. *Version v0. 0.1. Sept*, page 8.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring Massive Multitask Language Understanding. *Preprint*, arxiv:2009.03300.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, et al. 2022. The stack: 3 tb of permissively licensed source code. *arXiv preprint arXiv:2211.15533*.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2023. Starcoder: may the source be with you!

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, pages 22631–22648. PMLR.

Mathador. 2023. Mathador résultats du concours 2023.

Meta AI. 2024. Llama 3: Advanced Language Models for Open Research. GitHub repository.

Marianna Nezhurina, Lucia Cipolina-Kun, Mehdi Cherti, and Jenia Jitsev. 2024. Alice in wonderland: Simple tasks showing complete reasoning breakdown in state-of-the-art large language models. *arXiv preprint arXiv:2406.02061*.

Sébastien Puma, Emmanuel Sander, Matthieu Saumard, Isabelle Barbet, and Aurélien Latouche. 2023. Reconsidering conceptual knowledge: Heterogeneity of its components. *Journal of Experimental Child Psychology*, 227:105587.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Together. 2023. Redpajama: An open source recipe to reproduce llama training dataset.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E Gonzalez, and Ion Stoica. 2023. Rethinking benchmark and contamination for language models with rephrased samples. *arXiv preprint arXiv:2311.04850*.

Qihuang Zhong, Kang Wang, Ziyang Xu, Juhua Liu, Liang Ding, Bo Du, and Dacheng Tao. 2024. Achieving> 97% on gsm8k: Deeply understanding the problems makes llms perfect reasoners. *arXiv preprint arXiv:2404.14963*.

A Score Distribution

Models are instructed that only their last answer will be scored, and there is no obvious strategy for reaching a more complicated and higher scoring answer from a lower scoring one, as this is part of the task. Consequently, it is natural that even similarly performing models may have quite different score distributions as they may aim to obtain answers with different complexity levels (e.g., one may aim to obtain only highest-scoring answers, but may fail to obtain one more often than if simply aiming to reach the target). Figure 5 shows the score distribution for several low and high performing models. For instance, it is interesting to observe that Claude-3-opus outputs several times more max-scoring solutions than Llama-3-70b-instruct, while the models score about the same on average, based on Figure 4, or that Phi-3-small focuses on obtaining simple answers correct (just reaching the target, but not focusing on reaching high scores), which has resulted in a *higher overall performance* relative to Phi-3-medium, which produces higher-scoring solutions. x

B Evaluation with Multiple Attempts

In this section, we analyze the impact of allowing multiple attempts per question. During evaluation, we permit the model up to K attempts for each question. For $K = 5$, we observe a noticeable improvement in accuracy; however, it still falls short of being competitive with human performance. The results are presented in Table 5.

Table 5: Results with multiple attempts allowed to solve each question in Mathador benchmark.

Model	1 attempt	5 attempts	Gain
Llama-3-8B	6.32 ± 0.47	8.15 ± 0.12	29%
Llama-3-70B	11.52 ± 0.52	13.70 ± 0.58	19%

C Text Generation Strategies

Given that the nature of Mathador-LM benchmark is based on generating text to arrive at a solution, we investigate whether different decoding methods for language generation have any effect on the results. Therefore we consider both, the simple greedy decoding (Radford et al., 2019) and the more advanced nucleus sampling (Holtzman et al., 2019). We conduct an extensive search, exploring all possible combinations of *temperature* (0.0, 0.3, 0.5, 0.7, 0.9) and *Top-p* (0.1, 0.3, 0.5, 0.7, 1.0) hyper-parameters. As can be seen from Table 6, the results are not affected by choices of different text-generation strategies.

Table 6: Results with Llama-3-70B-Instruct on Mathador-LM benchmark under different text decoding techniques, evaluated across three few-shot configurations.

	2-shots	5-shots	20-shots
Greedy	12.8 ± 0.5	13.9 ± 0.1	14.2 ± 1.1
Nucleus	13.1 ± 0.6	13.8 ± 0.7	14.2 ± 0.9

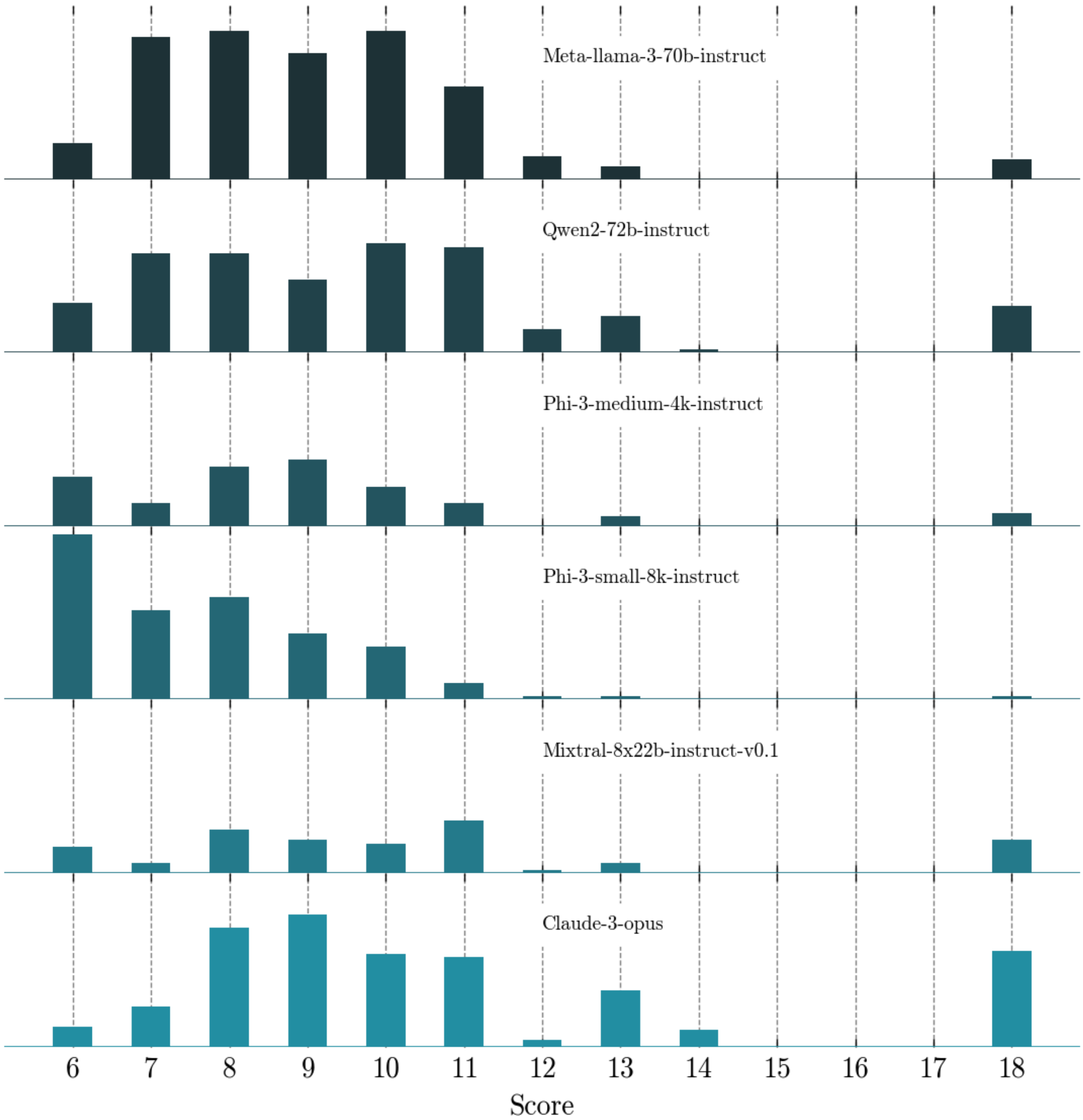

Figure 5: Distribution of scores for several models showing low correlation of higher overall performance with number of high scoring solutions.