RoMath: A Mathematical Reasoning Benchmark in Romanian

Anonymous submission

Abstract

Mathematics has long been conveyed through natural language, primarily for human understanding. With the rise of mechanized mathematics and proof assistants, there is a growing need to understand informal mathematical text, yet most existing benchmarks focus solely on English, overlooking other languages. This paper introduces RoMath, a Romanian mathematical reasoning benchmark suite comprising three subsets: *Baccalaureate*, *Competitions* and *Synthetic*, which cover a range of mathematical domains and difficulty levels, aiming to improve non-English language models and promote multilingual AI development. By focusing on Romanian, a low-resource language with unique linguistic features, RoMath addresses the limitations of Anglo-centric models and emphasizes the need for dedicated resources beyond simple automatic translation. We benchmark several open-weight language models, highlighting the importance of creating resources for underrepresented languages. Code and datasets will be made available.

1 Introduction

Mathematics has been a central intellectual preoccupation to humans since the beginning of civilization, the first mathematical writings dating back approximately 4000 years (Friberg, 1981). Historically and in the present, mathematics has been mostly written, spoken and taught in natural language, albeit with its own specialized vocabulary, having strict formalism only sparsely introduced between free-text explanations and reasoning. The primary audience of mathematical reasoning is other humans, not computers. The natural language of mathematics contains a mix of formulas, symbols, neologisms, jargon and words with different meanings than their common meaning (e.g., "real" / "imaginary" numbers). Mathematics implies rigor and precise reasoning, qualitatively different from general NLP. There is a pressing need to automatically process and understand the existing large amount of mathematical text written in natural language to enable efficient knowledge extraction, facilitate automated theorem proving, and enhance accessibility for both researchers and automated systems.

Recently, Large Language Models (LLMs) have shown great promise in handling a multitude of natural language tasks, including tackling mathematical reasoning problems (Ahn et al., 2024; Yue et al., 2023; Azerbayev et al., 2024; Shao et al., 2024). Out of the common benchmark suite for evaluating LLMs, datasets such as GSM8k (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) remained central in the development of reasoning models (Jaech et al., 2024; DeepSeek-AI et al., 2025), and continue to be challenging even for the larger, proprietary models (Arora et al., 2023).

Current mathematics benchmarks and datasets have focused solely on English, mostly disregarding other low-resourced languages. The tacit requirement for using AI tools is fluency in English (Shi et al., 2022). However, mathematical reasoning ability is independent of the underlying language (Rescorla, 2024) and Anglo-centric models have been shown to exhibit the same biases of the English language, even when prompted in other languages (Wendler et al., 2024; Wang et al., 2023; Liu et al., 2023). The focus on datasets and models in a language other than English allows the democratization of learning for underrepresented languages and cultures.

Recently, Romanian LLM development has started to flourish with initiatives such as OpenLLM-Ro (Masala et al., 2024), having fine-tuned several LLMs on Romanian text. However, for evaluation, the authors used translated versions of popular English datasets and several native Romanian benchmarks, but no evaluation is performed on dedicated reasoning tasks in Romanian. Aside from code generation (Cosma et al., 2024; Dumi-

tran et al., 2024), currently there is no reasoning benchmark for Romanian.

In this work, we propose **RoMath**, a Romanian mathematical reasoning benchmarking suite comprised of three datasets, *Baccalaureate*, *Competitions* and *Synthetic*, each with its own particularities. RoMath aims to provide a comprehensive benchmark suite, having high-school-level problems across multiple domains (linear and abstract algebra, calculus, limits, geometry, probabilities) and across multiple levels of difficulty, ranging from easy calculations, to baccalaureate-level problems, to more difficult, proof-centric, competition-level problems. The purpose of RoMath is to provide a mathematical benchmark for Romanian and to stimulate the development of enhanced reasoning capabilities of non-English LLMs.

This work makes the following contributions:

1. We construct and release **RoMath**, a novel mathematical reasoning benchmark suite with 76,910 problem statements in Romanian, consisting of three subsets, each with its own particularities and difficulty levels: *Baccalaureate* (5,777 problems), *Competitions* (1,133 problems) and *Synthetic* (63,000 problems). We collect and curate math problems using a semi-automatic workflow using foundational LLMs for providing structured output from unstructured raw OCR input and annotating problems with relevant metadata.

2. We provide a comprehensive benchmark of several English and Romanian open-weight LLMs under several common scenarios - zero-shot, LoRA fine-tuning (Hu et al., 2022) and training with verifiable rewards using GRPO (Shao et al., 2024). Furthermore, we provide an evaluation procedure using an LLM-as-a-judge paradigm (Zheng et al., 2023) for proofs, and analyze its performance to properly estimate solution correctness.

3. We show that simple translation of problem statements is not enough, as sub-par translations of precise mathematical language significantly reduces performance. Consequently, we emphasize the need for more dedicated resources in languages other than English.

2 Related Work

Pretraining datasets for mathematics. Interest in representation learning of mathematical expressions and text has existed in the past (Peng et al., 2021; Collard et al., 2022). However, beyond representation learning, with the recent success of LLMs in a wide range of tasks, there has been increased attention to training and evaluating mathematical reasoning of LLMs. For pretraining, the general approach is to filter Common Crawl web pages and PDFs to obtain high quality math tokens. For instance, datasets such as MathWebPages (Lewkowycz et al., 2022), ProofPile (Azerbayev et al., 2023a) and OpenWebMath (Paster et al., 2023) are used to pretrain high performing LLMs specialized in math such as Minerva (Lewkowycz et al., 2022) and LLema (Azerbayev et al., 2023b).

Mathematical reasoning benchmarks. Regarding benchmarks, the most popular dataset is GSM8K (Cobbe et al., 2021), containing middle-school Math Word Problems (MWPs). An improved variant that contains process supervision (i.e., supervision at each intermediary reasoning step) is PRM800K (Lightman et al., 2023). However, these datasets are regarded as too simple to demonstrate advanced mathematical reasoning of LLMs. Consequently, MATH (Hendrycks et al., 2021) is a comparatively more difficult dataset, containing high-school problems from domains such as calculus, linear algebra, geometry and number theory. MathVISTA (Lu et al., 2024) is another similar benchmark, that contains mathematical reasoning in visual contexts (e.g., plots, natural images, functions).

Aside from simple word problems (Cobbe et al., 2021) and datasets focused on QA-type problems, more difficult competition-level benchmarks have been proposed. For instance, ARB (Sawada et al., 2023) is a dataset comprised of problems from math competitions and problems from specialized books, with special care taken to avoid data contamination. While it contains problems that require proofs, ARB only contains 105 problems. MathOdyssey (Fang et al., 2024) contains difficult high-school and university-level problems but it is similarly small, as it contains only 387 problems.

Non-English benchmarks. Regarding datasets in languages other than English, there have been efforts in Arabic with datasets such as ArMATH (Alghamdi et al., 2022) and Chinese with Ape210k (Zhao et al., 2020), Math23k (Ling et al., 2017), CMath (Wei et al., 2023). Otherwise, outside of (automatically) translated versions of popular sets such as GSM8k (Masala et al., 2024), as far as we

2

Name	# Problems	Level	Language	Notes
APE210K (Zhao et al., 2020)	210K	E	Chinese	Requires basic arithmetic and common sense
MATH23K (Ling et al., 2017)	23K	E	Chinese	Contains questions, answers and rationales
CMATH (Wei et al., 2023)	1.7K	E	Chinese	Contains number of reasoning steps
ARMATH (Alghamdi et al., 2022)	6K	E	Arabic	Novel problems and inspired by MATH23K
GSM8K (Cobbe et al., 2021)	8.5k	M	English	Linguistically diverse.
MATH (Hendrycks et al., 2021)	12.5k	H	English	Problems are put into difficulty levels 1-5
PRM800K (Lightman et al., 2023)	12k	H	English	MATH w/ step-wise labels
MathOdyssey (Fang et al., 2024)	387	H C	English	Problems from GAIC Math 2024
ARB (Sawada et al., 2023)	105	C	English	Contest problems and university math proof
AQUA (Ling et al., 2017)	100K	C	English	GRE&GMAT questions
RoMath				
🇷🇴 *Baccalaureate*	5.8k	H	Romanian	Easy-medium, baccalaureate problems.
🇷🇴 *Competitions*	1.1k	M H	Romanian	Difficult, proofs, competition problems.
🇷🇴 *Synthetic*	63k	H	Romanian	Calculation-based, procedurally generated.

Table 1: Comparison with other mathematical reasoning benchmarks. RoMath is the only Romanian mathematics benchmark outside of translated versions of English benchmarks. Table adapted from Ahn et al. (2024).
E = Elementary, M = Middle School, H = High School, C = College.

know, no datasets currently exist for Romanian or other Latin languages.

Comparison with prior work. Table 1 shows a comparison between similar datasets and RoMath. RoMath comprises middle-school, high-school and competitive high-school problems in Romanian covering multiple subjects and types of problems (proofs, calculations, equations, etc.). Different from prior datasets, RoMath is the first dedicated resource for mathematical reasoning in Romanian, a low-resource language of ∼23M speakers, which has its unique linguistic particularities (Dinu and Dinu, 2005; Dinu and Enăchescu, 2007).

3 Method

We describe below the process for collecting *Baccalaureate* and *Competitions*, the two subsets that are collected by crawling publicly available PDFs. The *Synthetic* subset is comprised of programmatically generated problems directly in Romanian.

3.1 Dataset Construction

In order to construct a high quality set of mathematical problems paired with solutions, we crawl publicly available PDFs from country-wide mathematics competitions and questions from the Romanian baccalaureate exam. Figure 1 showcases our approach. After collecting raw PDFs, usually having separate documents for problem sets and their respective solutions, we utilize an academic document-focused OCR (i.e., MathPix (Mathpix, 2024)) to extract the underlying text and mathematical formulas / statements in LaTeX format. The final output is represented in Markdown format.

To parse the content, instead of relying on brittle handcrafted rules and regex expressions, we utilize a commercial LLM (i.e., Claude 3 Sonnet (Anthropic, 2024)) to parse the raw text and to output structured JSON from unstructured Markdown. The LLM is provided with several examples of how to structure the final JSON (see Appendix A, Table 6 for the system prompt). The JSON output contains the LaTeX-formatted problem statement and its appropriate solution. Finally, we again utilize a commercial LLM to annotate the domain of the problem and to extract final answers for non-proof problems for easier evaluation (similar to Hendrycks et al. (2021), we enclose the final answer, if it exists, into a \boxed{} tag). If a problem contains multiple sub-problems, we ensure that each sub-problem is self-contained and that the solution does not rely heavily on previous sub-problems' solutions. To split a problem into sub-problems, we used a prompt (presented in Appendix A, Table 6) with specific instructions for parsing the data and output sub-questions that are self-contained. For example, if a problem is structured as follows:

<center>

<problem_statement>
<question_1>
<question_2>

</center>

The output is formatted as two separate standalone problems:

<center>

<problem_statement> <question_1>
<problem_statement> <question_2>

</center>

Additionally, through manual inspection, we further removed any sub-questions that contained references to previous sub-questions (e.g. "Using the
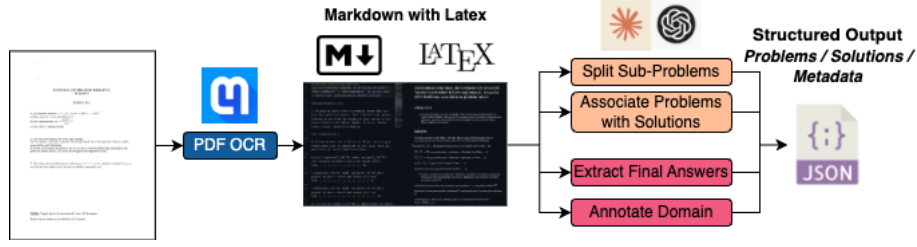
Figure 1: Overall diagram of our approach to curating problems from existing PDFs. We employ MathPix (Mathpix, 2024) to OCR PDFs and obtain markdown with LaTeX formatting for mathematical statements. We further process the markdown using proprietary LLMs to split into sub-problems, associate problems with the appropriate solution and annotate each problem with metadata.
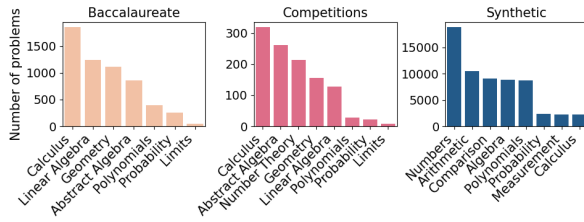

Figure 2: Distribution of the number of problems per domain for *Baccalaureate*, *Competitions* and *Synthetic*.

result from a) compute [...]"). Figure 2 shows the distribution of problems per domain.

3.2 RoMath Suite

RoMath is comprised of three subsets: *Baccalaureate*, *Competitions* and *Synthetic*. By its construction, each subset of RoMath features problems that require both single-step and multi-step reasoning for correctly solving problems. Usually, single-step reasoning problems involve simple calculations, while multi-step reasoning problems require solving intermediate solutions to reach a valid conclusion. Table 2 showcases selected examples from each subset.

Baccalaureate is composed of problems and solutions from the Romanian Baccalaureate exam. The Romanian Baccalaureate is a country-wide exam for graduating high-school students, comprised of three subjects, each with several problems and sub-problems. Students taking the Baccalaureate exam consider the calculus problems, such as solving an integral or computing a limit, to be the most difficult. However, the calculus problems rarely require more than 2 steps of reasoning and some calculation. This subset contains a total of 5777 problems: 4.3k problems for training and 1.48k testing. Most problems ($4617 / \sim 80\%$) in this subset are verifiable (i.e., have a single final answer), while some ($1160 / \sim 20\%$) require proofs.

Synthetic
Problem Statement: Care este rezultatul împărțirii lui -54 la -36495?
Solution: $\frac{6}{4055}$
Problem Statement: Fie u definit ca fiind $(2 - \frac{18}{15}) \cdot 5$. Găsește valoarea lui r din ecuațiile $u \cdot y + 8 = 0, -28 = 3 \cdot r + 4 \cdot y - 5$.
Solution: -5

Baccalaureate
Problem Statement: Se consideră funcția $f : \mathbf{R} \to \mathbf{R}$, $f(x) = e^x - x$. Să se calculeze $\int_0^1 f(x)\,dx$.

Solution: $\int_0^1 f(x)\,dx = e - \frac{3}{2}$. Soluția finală este $\boxed{e - \frac{3}{2}}$

Problem Statement: Se consideră funcțiile $f_n : \mathbf{R} \to \mathbf{R}$, $f_1(x) = x^3 - 3x^2 + 3x$ și $f_{n+1}(x) = (f_1 \circ f_n)(x), \forall n \in \mathbf{N}^*, \forall x \in \mathbf{R}$. Să se rezolve în mulțimea numerelor reale ecuația $f_1(x) + f_2(x) + f_3(x) - 3 = 0$.
Solution: Observăm că $x = 1$ este soluție. Dacă $x > 1 \Rightarrow f_1(x) + f_2(x) + f_3(x) > 3$. Analog dacă $x < 1 \Rightarrow f_1(x) + f_2(x) + f_3(x) < 3$. Deci $x = 1$ este soluție unică. Soluția finală este $\boxed{x = 1}$

Competitions
Problem Statement: Se consideră numerele complexe u, v și z astfel încât $|u| = |v| = 1$ și $|u + v| = \sqrt{3}$. Să se demonstreze că: $\mathrm{u} \cdot \overline{v} + \overline{u} \cdot v = 1$.
Solution: $|\mathrm{u} + \mathrm{v}|^2 = 3 \Leftrightarrow (\mathrm{u} + \mathrm{v})(\overline{\mathrm{u}} + \overline{\mathrm{v}}) = 3 \Leftrightarrow \mathrm{u} \cdot \overline{v} + \overline{\mathrm{u}} \cdot v = 1$

Problem Statement: Să se rezolve, în $\mathbb{R}$, inecuația: $\left(\frac{2}{5}\right)^{\frac{6-5x}{5x+2}} \leq \frac{25}{4}$

Solution: $\left(\frac{2}{5}\right)^{\frac{6-5x}{5x+2}} \leq \frac{25}{4} = \left(\frac{5}{2}\right)^{-2} \Rightarrow \frac{6-5x}{5x+2} \geq -2$, cu $x \neq -\frac{2}{5}$. $\frac{6-5x}{5x+2} \geq -2 \Rightarrow \frac{x+2}{5x+2} \geq 0 \Rightarrow x \in (-\infty, -2] \cup \left(-\frac{2}{5}, \infty\right)$

Table 2: Qualitative examples from each subset of Ro-Math.

Furthermore, $4038 / \sim 69\%$ problems in this category also have intermediate steps provided in the ground-truth solution. In this set, there are multiple domains, with varying difficulty: geometry, combinatorics, abstract algebra, linear algebra, calculus (integrals and derivatives), and limits. In all categories, we discarded any problem that required reasoning over images or plots. For instance, geometry problems do not have an accompanying drawing or figure. If we encountered images in the source PDFs, we removed the problem entirely through manual inspection. The *Baccalaureate* subset includes only standalone geometry problems: an example of such a problem would be the following (here, translated in English for convenience): *"In a Cartesian coordinate system xOy we consider the points $A_n(n, 0)$ and $B_n(0, n)$, with $n \in \{1, 2, 3\}$. Calculate the area of the triangle $A_1A_2B_2$."*

Competitions is the hardest subset of RoMath, containing 1133 problems sourced from mathematics competitions, with problems ranging from local to inter-county and olympiad events, out of which 804 problems are for training and 329 for testing. Different from *Baccalaureate*, this subset also contains middle-school problems. Around half of the problems (594 / $\sim 52\%$) require proofs for a complete solution, while the rest are directly verifiable. Almost all problems in this subset have intermediate explanations. The problems in *Competitions* are considered hard, requiring insight and problem-solving skills outside of simple symbol manipulations (Polya, 1971). The extraction and post-processing steps are identical to those in *Baccalaureate*.

Synthetic is programmatically generated, using the approach of Saxton et al. (2019), in which we manually translate the source key-phrases and formulations in Romanian. Problems in this subset have a single final answer. Problems are mostly algebraic in nature, and are split into arithmetic, calculus, derivatives, integrations, polynomials, composition of problems, comparisons, manipulating expressions (e.g., simplification), numbers, measurements. All problems in this subset are verifiable, having only a single final answer provided, without intermediate steps, making it difficult to directly provide an answer without the use of external tools or chain-of-thought prompting. In contrast to the other sub-sets in RoMath, there is less linguistic variation present in problem statements, but there is complete control over correctness and difficulty. We emphasize that *Synthetic* is *not* a direct translation of the problems contained in DeepMind Mathematics (Saxton et al., 2019), but rather a manual translation of the phrases that are used to generate the problems. As such, one could generate an indefinite number of problems. We make the code for generating *Synthetic* open-source and provide, for convenience, 63k generated problems, out of which 55.9k problems for training and 7.1k for testing.

3.3 Evaluation Procedure

Generally, there are two ways to evaluate solutions: *(i)* for verifiable problems (i.e., containing a single final answer), correctness is estimated by direct string comparison between the model answer and the correct answer after normalization (Hendrycks et al., 2021; Cobbe et al., 2021) and *(ii)* using a proof-checker for problems requiring proofs (Li et al., 2024).

Evaluating the correctness of a solution to a mathematics problem for proof problems is still an open problem. Using a proof-checker is not always feasible as it requires the problems and solutions to already be formalized into the language of the proof-checker (Trinh et al., 2024), an unrealistic requirement for most mathematics written in natural language. For proof-type problems, where it is necessary to check for correctness at every reasoning step in natural language, there is no consensus on the evaluation procedure outside of formal proof-checkers.

However, more recent methods (Fang et al., 2024) have adopted a "soft" evaluation of proof solutions by employing an external judge LLM tasked to output a correctness score given the problem statement, the correct solution and a provided solution to be scored.

To evaluate solutions to RoMath, we propose the following procedure: For evaluating verifiable problems, we adopt the procedure from (Hendrycks et al., 2021) for string comparison after the solutions are normalized; this requires the model to output solutions in a `\boxed{}` tag. However, if the model does not provide the solution in this format or if the problem requires a proof, we employ a judge LLM to estimate correctness, inspired by several other works (Zheng et al., 2023; Fang et al., 2024). Since the use of proprietary LLMs is prohibitively expensive, there are concerns with reproducibility, and there is no information on the architecture and training dataset, we use existing open-weight models.

4 Baselines and Results

4.1 Judge Evaluation

Very few analyses have been performed to gauge the performance of the judge LLM: for instance, a more recent study (Bavaresco et al., 2024) showed that LLMs exhibit a large variance across datasets in correlation to human judgments. However, there is no study estimating the performance of judge LLMs for mathematical reasoning in a language other than English. Using LLMs as judges is a reasonable proxy for estimating performance, and we show in Section 4.4 that performance is relatively robust across multiple judges.

In this section, we conduct an analysis of the performance of multiple open-weight judge models in

evaluating solution correctness in Romanian, using both Romanian and English system prompts (see Appendix A Tables 8 and 9).

We programmatically construct a dataset of 300 problems from the training sets of *Baccalaureate* and *Competitions* containing correct and incorrect solutions. Correct solutions are constructed by symbol changes (Meadows et al., 2023) and removal of natural language text (keeping only mathematical expressions) of the original ground-truth solution, and incorrect solutions are either original solutions with some operators / number modified (e.g., $+$ sign changed to $-$, or $<$ symbol changed to $\geq$, and others) or a similar solution, but not exactly the same, from another problem based on the Levenstein distance.

In Table 3, we showcase the performance of multiple LLMs-as-judges on our programmatically generated dataset to estimate judge performance. We tested Qwen2 (Yang et al., 2024) family of models, as well as the math-specialized variant Qwen2-Math-7B, deepseek-math (Shao et al., 2024), Phi-3 (Abdin et al., 2024), Llama3-70B (Dubey et al., 2024), Mathstral (Mistral AI, 2024), and Mixtral-8x7b (Jiang et al., 2024). For this synthetical dataset, we obtained that Qwen2-7B-Instruct prompted in English obtained the best overall results of 91% accuracy judging solution correctness. Surprisingly, the math-specialized models severely underperformed at this task. As such, unless otherwise specified, we used Qwen2-7B-Instruct prompted in English as a judge for the rest of the non-verifiable results.

Judge Model	System Prompt	Acc. ↑	FPR ↓	FNR ↓
deepseek-math-7b-instruct	ro	0.51	0.66	0.31
Meta-Llama-3-70B-Instruct	ro	0.86	0.24	0.03
Mixtral-8x7B-Instruct-v0.1	ro	0.84	0.27	0.03
Qwen2-Math-7B-Instruct	ro	0.87	0.22	0.03
Qwen2-7B-Instruct	ro	**0.90**	**0.17**	0.02
deepseek-math-7b-instruct	en	0.74	0.31	0.22
Meta-Llama-3-70B-Instruct	en	0.84	0.29	**0.01**
Mixtral-8x7B-Instruct-v0.1	en	0.84	0.27	0.03
Qwen2-Math-7B-Instruct	en	0.89	0.16	0.06
Qwen2-7B-Instruct	en	**0.91**	0.12	0.05

Table 3: Judge LLM performance on a programmatically generated dataset of correct and incorrect student solutions.

4.2 Model Benchmark

We chose to benchmark several open-weight LLMs, as opposed to proprietary models, to make the benchmark reproducible and to avoid unnecessary inference costs. We evaluated the performance under 0-shot and LoRA fine-tuned models for Qwen2-7B, Phi-3, Meta-Llama-8B and math-specialized variants such as Qwen2-Math-7B, deepseek-math-7b, Mathstral-7b. We evaluated larger models under 0-shot setting: Meta-Llama-70B and Mixtral-8x7B. Furthermore, we also evaluated Romanian-specialized models trained with continual pretraining on Romanian tokens, but with no focus on math tokens: RoLlama3-8B and RoMistral-7b (Masala et al., 2024). In Appendix A, Table 7.

For fine-tuning the models, we used LoRA (Hu et al., 2022), using a rank of 8, alpha of 32 and dropout of 0.1, applied on all linear layers. Due to hardware limitations, we used a small batch size of 4 and a learning rate of 0.00002 with a linear decay over the 3 training epochs.

	Model	Scenario	Baccalaureate		Competitions		Synthetic	
			Accuracy	F1	Accuracy	F1	Accuracy	F1
Romanian	OpenLLM-Ro/RoLlama3-8b-Instruct	0-shot	0.50	0.67	0.48	0.65	0.18	0.30
		fine-tuned	0.18	0.31	0.50	0.67	–	–
	OpenLLM-Ro/RoMistral-7b-Instruct	0-shot	0.50	0.66	0.44	0.61	0.16	0.27
		fine-tuned	0.18	0.31	0.36	0.53	–	–
General-Purpose	Qwen/Qwen2-7B-Instruct	0-shot	0.40	0.57	0.55	0.71	0.29	0.45
		fine-tuned	0.54	0.70	0.48	0.65	–	–
	microsoft/Phi-3-mini-4k-instruct	0-shot	0.36	0.53	0.33	0.50	0.07	0.14
		fine-tuned	0.25	0.40	0.41	0.58	–	–
	meta-llama/Meta-Llama-3-8B-Instruct	0-shot	0.34	0.51	0.53	0.69	0.25	0.40
		fine-tuned	0.18	0.31	0.33	0.49	–	–
Math-Specialized	Qwen/Qwen2-Math-7B-Instruct	0-shot	0.32	0.48	0.55	0.71	0.27	0.43
		fine-tuned	0.48	0.65	0.57	0.73	–	–
	deepseek-ai/deepseek-math-7b-instruct	0-shot	0.56	0.72	0.59	0.74	0.21	0.35
		fine-tuned	0.29	0.44	0.56	0.72	–	–
	mistralai/Mathstral-7b-v0.1	0-shot	0.30	0.46	0.61	0.75	0.36	0.53
		fine-tuned	0.21	0.34	0.56	0.71	–	–
Large	meta-llama/Meta-Llama-3-70B-Instruct	0-shot	0.25	0.40	0.22	0.36	0.10	0.19
		5-shot	0.08	0.15	0.09	0.16	0.07	0.13
	mistralai/Mixtral-8x7B-Instruct-v0.1	0-shot	0.43	0.60	0.60	0.75	0.32	0.48
		5-shot	0.25	0.40	0.25	0.40	0.24	0.38

Table 4: Results for various open-weight LLMs on *Baccalaureate*, *Competitions* and *Synthetic*, under 0-shot and fine-tuned scenarios.

In Table 4, we showcase the performance of the models under zero-shot, and LoRA-fine-tuned scenarios. The best performing model on the *Baccalaureate* subset is deepseek-math-7b, while on *Competitions* and *Synthetic* Mathstral-7b obtains the best results. However, the Romanian models, RoLlama-8b and RoMistral-7b obtain competitive results on all subsets, which can be attributed to their better understanding of Romanian text compared to English-focused models, since specialization on mathematical text did not receive a particular emphasis during training. Surprisingly, we obtained that fine-tuning does not always result in improved performance. Fine-tuning improves performance on *Baccalaureate* for Qwen2-7b and Qwen2-Math-7b, while on *Competitions*, RoLlama-7b, Phi-3, Qwen2-Math-7b benefit from further fine-tuning. One possible explanation is that the solutions present in RoMath are qualitatively dif-

6

ferent (different formatting, explanation style) than solutions present in other math datasets (Cobbe et al., 2021; Hendrycks et al., 2021) and Chain-of-Thought style prompting (Wei et al., 2022). Further investigation on this effect is left as future work. In Figure 3, we show extended results per problem domain for each dataset. Qualitative examples of generated solutions are shown in the Appendix A Tables 10 and 11.
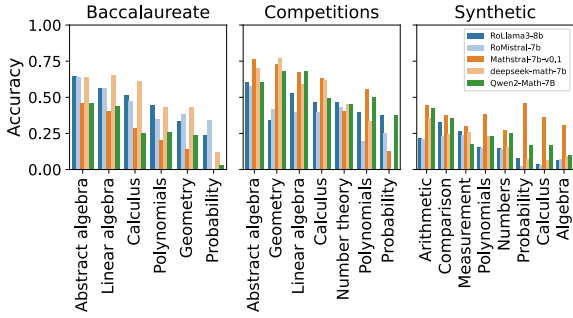

Figure 3: Performance of Romanian models and math-specialized models on each domain from each RoMath subset.

4.3 Training with Verifiable Rewards

Since a significant proportion of problems in Ro-Math includes intermediate steps and are verifiable, we tested if the problems are of sufficiently high quality to enable training with rewards. We adopt a part of the training procedure from Shao et al. (2024), and fine-tune two variants of the Llama3.2 (Dubey et al., 2024) (1B and 3B parameters) and Qwen2 (Yang et al., 2024) (0.5B and 1.5B) family of models. For supervised fine-tuning (SFT), we train on all problems from *Baccalaureate* and *Competitions* that contain intermediate steps to force the model to conform to the specified output format of *<raționament> [. . .] </raționament> <răspuns> [. . .] </răspuns>*.

Further, we train using GRPO (Shao et al., 2024) with 4 completions per prompt on all verifiable problems from *Baccalaureate* and *Competitions*, using only a correctness reward and a format reward. Figure 4 shows the performance on the verifiable problems from the *Baccalaureate* subset for this setting. Training with rewards reliably boosts performance compared to only supervised fine-tuning. As such, RoMath can be a useful resource for training Romanian reasoning models.
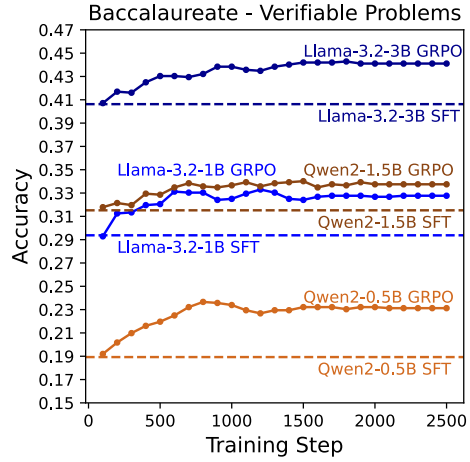

Figure 4: Performance of GRPO-trained LLama-3.2 and Qwen2 on on a subset of *Baccalaureate* that has verifiable answers.

4.4 Impact of the Judge Model

In Figure 5, we compared multiple judge models to gauge their effect on downstream performance. Based on Table 3, we used Qwen2-7B, Llama-70B and Mixtral-8x7b as judges and used them to evaluate the performance of the same Qwen2-7B, Llama-70B and Mixtral-8x7b. We chose the same judges and downstream models to check if judges prefer the output of their own model. From Figure 5 we find that judges do not have "favorites". However, we do find that, for example, in *Competitions*, where there are more proofs than in *Baccalaureate*, the Llama-70B and Mixtral-8x7b judges give higher scores on average, which might explain why results on the *Competitions* subset are higher: judges might artificially inflate results. While the differences between judges are small, there is a clear ascending trend between them.
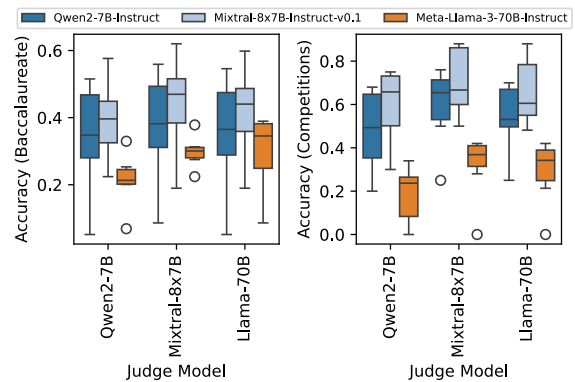

Figure 5: Performance using different judge models.

4.5 Translating Romanian Problems to English

Translating domain-specific technical language is non-trivial. Al-Tarawneh (2024) identified multiple linguistic challenges that make translation difficult. Translating mathematics is challenging due to the need for precise language, as even slight ambiguities can alter meaning. Although mathematical concepts are universal, their interpretation varies across cultures. Additionally, mathematical symbols and notations are not always standardized across languages, and mathematical terms lack direct equivalents in other languages leading to potential confusion if not properly accounted for.

We used the NLLB (NLLB Team et al., 2022) family of models (600M, 1.3B, and 3.3B) to translate from Romanian to English the test sets for *Baccalaureate* and *Competitions*, as the models have established numerical benchmarks on Romanian to English translation. Directly translating the full problem statement and solution resulted in "gibberish" translations due to the mathematical symbols present in the text. As such, we opted to keep the LaTeX-delimited section intact and only translate the surrounding natural language. While this approach might lose some of the larger context, we found it to be the only satisfactory approach. Still, the resulting translations contain unnatural English formulations and sometimes spurious text. For instance, the problem statement *"Se consideră funcția $f : \mathbf{R} \to \mathbf{R}, f(x) = e^x - x + 1$. Să se calculeze $\lim_{x\to 0} \frac{f(x)-f(0)}{x}$"* is translated as *"It's considered function $f : \mathbf{R} \to \mathbf{R}, f(x) = e^x - x + 1$. Let's figure it out. $\lim_{x\to 0} \frac{f(x)-f(0)}{x}$ [♩ I'm not gonna let you down ♩]"*, in which the part "*[♩ I'm not gonna let you down ♩]*" is introduced spuriously by the translation model.

In Table 5, we showcase the performance of math-specialized LLMs on the English-translated version of *Baccalaureate* and *Competitions* using the different sizes of NLLB. Compared to the original Romanian text, translating severely degrades performance. We found that performance improves with the translation model size, but up to a certain point. The main point of failure is handling the math LaTeX tokens without disrupting the surrounding text. The use of an LLM for translation might be more appropriate only if their reliability and control of their output are properly established, and proper benchmarks for translation in Romanian are in place.

	Model	Translation Model	Romanian Accuracy	English-Translated Accuracy	Diff.
Baccalaureate	Qwen/Qwen2-Math-7B-Instruct	nllb-200-distilled-600M	0.32	0.04	−0.28
		nllb-200-1.3B		0.03	−0.29
		nllb-200-3.3B		0.03	−0.29
	deepseek-ai/deepseek-math-7b-instruct	nllb-200-distilled-600M	0.56	0.09	−0.47
		nllb-200-1.3B		0.05	−0.51
		nllb-200-3.3B		0.07	−0.49
	mistralai/Mathstral-7b-v0.1	nllb-200-distilled-600M	0.30	0.07	−0.23
		nllb-200-1.3B		0.07	−0.23
		nllb-200-3.3B		0.07	−0.23
Competitions	Qwen/Qwen2-Math-7B-Instruct	nllb-200-distilled-600M	0.55	0.17	−0.38
		nllb-200-1.3B		0.20	−0.35
		nllb-200-3.3B		0.19	−0.36
	deepseek-ai/deepseek-math-7b-instruct	nllb-200-distilled-600M	0.59	0.09	−0.50
		nllb-200-1.3B		0.12	−0.47
		nllb-200-3.3B		0.10	−0.49
	mistralai/Mathstral-7b-v0.1	nllb-200-distilled-600M	0.61	0.19	−0.42
		nllb-200-1.3B		0.21	−0.40
		nllb-200-3.3B		0.20	−0.41

Table 5: Results on RoMath-Baccalaureate and RoMath-Competitions for math-specific LLMs in 0-shot setting with English-translated problems. Performance drops significantly due to poor quality translations.

5 Conclusions and Future Directions

In this paper, we proposed RoMath, a benchmarking suite consisting of three datasets with mathematical problems written in Romanian: *Baccalaureate*, *Competitions* and *Synthetic*. We detailed the construction process and composition for each subset and benchmarked several open-weight LLMs under different training and evaluation scenarios. We are the first to provide quantitative results for mathematical reasoning in Romanian.

Surprisingly, we found that mathematics problems written in Romanian can be properly handled by English-centric models, providing proper solutions in Romanian. It is unclear why this occurs, especially since such models are not explicitly trained on Romanian math tokens and most models have strong language filters to train only on English. Our results suggest that such LLMs would potentially receive a passing grade (i.e., more than 50%) on the Romanian baccalaureate exam, scoring an average of ∼56% across all problems in *Baccalaureate*.

An important future direction is reliable automatic annotations with chain-of-thought (CoT) traces for multilingual reasoning problems. Our results indicate that a significant factor in improving performance in mathematical reasoning is the presence of intermediate reasoning steps in the solutions. Performance is not reliably improved by fine-tuning without CoT, and the presence of more detailed solutions enables scalable training with reinforcement learning algorithms such as GRPO (Shao et al., 2024). Currently, only a subset of RoMath contains intermediate steps for problem solutions, and further structured annotations could significantly increase the data quality.

Limitations

The main limitation of this work is the use of an external LLM as a judge to estimate solution correctness, which might skew the results, artificially inflate performance. For example, some generated solutions for proof-type problems obtain the correct final result, but the intermediate steps are incorrect. In some cases, the judge model deemed these types of solutions as correct, whereas they are not. While this is an inherent limitation in literature for mathematics datasets that contain proofs, this is currently an open problem and there are on-going efforts to formalize proof verification (Gowers et al., 2024). Furthermore, we argued that the proper way to evaluate solutions of generated proofs is by using an external proof verification tool such as Lean (de Moura et al., 2015).

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone.

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*.

Alalddin Al-Tarawneh. 2024. Bridging languages and numbers: Exploring the intersection of translation studies and mathematics. *Appl. Math*, 18(3):513–519.

Reem Alghamdi, Zhenwen Liang, and Xiangliang Zhang. 2022. ArMATH: a dataset for solving Arabic math word problems. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 351–362, Marseille, France. European Language Resources Association.

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku.

Daman Arora, Himanshu Singh, and Mausam. 2023. Have LLMs advanced enough? a challenging problem solving benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7527–7543, Singapore. Association for Computational Linguistics.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W Ayers, Dragomir Radev, and Jeremy Avigad. 2023a. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics. *arXiv preprint arXiv:2302.12433*.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023b. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2024. Llemma: An open language model for mathematics.

Anna Bavaresco, Raffaella Bernardi, Leonardo Bertolazzi, Desmond Elliott, Raquel Fernández, Albert Gatt, Esam Ghaleb, Mario Giulianelli, Michael Hanna, Alexander Koller, et al. 2024. Llms instead of human judges? a large scale empirical study across 20 nlp evaluation tasks. *arXiv preprint arXiv:2406.18403*.

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*.

Jacob Collard, Valeria De Paiva, Brendan Fong, and Eswaran Subrahmanian. 2022. Extracting mathematical concepts from text. *arXiv preprint arXiv:2208.13830*.

Adrian Cosma, Ioan-Bogdan Iordache, and Paolo Rosso. 2024. RoCode: A dataset for measuring code intelligence from problem definitions in Romanian. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 14173–14185, Torino, Italia. ELRA and ICCL.

Leonardo Mendonça de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. 2015. The lean theorem prover (system description). In *CADE*, volume 9195 of *Lecture Notes in Computer Science*, pages 378–388. Springer.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen,

Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

Anca Dinu and Liviu P Dinu. 2005. On the syllabic similarities of romance languages. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 785–788. Springer.

Liviu P. Dinu and Denis Enăchescu. 2007. *On clustering Romance languages*, pages 521–528.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Adrian Marius Dumitran, Adrian Cătălin Badea, and Ștefan-Gabriel Muscalu. 2024. Evaluating the performance of large language models in competitive programming: A multi-year, multi-grade analysis. *THE 18TH INTERNATIONAL CONFERENCE ON INNOVATIONS IN INTELLIGENT SYSTEMS AND APPLICATIONS -INISTA2024*.

Meng Fang, Xiangpeng Wan, Fei Lu, Fei Xing, and Kai Zou. 2024. Mathodyssey: Benchmarking mathematical problem-solving skills in large language models using odyssey math data. *arXiv preprint arXiv:2406.18321*.

Jöran Friberg. 1981. Methods and traditions of babylonian mathematics: Plimpton 322, pythagorean triples, and the babylonian triangle parameter equations. *Historia Mathematica*, 8(3):277–318.

Prof Sir Timothy Gowers, AlphaProof, and AlphaGeometry. 2024. Ai achieves silver-medal standard solving international mathematical olympiad problems.

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. *NeurIPS*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Zhaoyu Li, Jialiang Sun, Logan Murphy, Qidong Su, Zenan Li, Xian Zhang, Kaiyu Yang, and Xujie Si. 2024. A survey on deep learning for theorem proving. *arXiv preprint arXiv:2404.09939*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Chen Cecilia Liu, Fajri Koto, Timothy Baldwin, and Iryna Gurevych. 2023. Are multilingual llms culturally-diverse reasoners? an investigation into multicultural proverbs and sayings. *arXiv preprint arXiv:2309.08591*.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2024.

Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*.

Mihai Masala, Denis Ilie-Ablachim, Alexandru Dima, Dragos-Georgian Corlatescu, Miruna Zavelca, Ovio Olaru, Simina Terian-Dan, Andrei Terian-Dan, Marius Leordeanu, Horia Velicu, Marius Popescu, Mihai Dascalu, and Traian Rebedea. 2024. "vorbeşti româneşte?" a recipe to train powerful romanian llms with english instructions.

Mathpix. 2024. Ai-powered document automation.

Jordan Meadows, Marco Valentino, Damien Teney, and Andre Freitas. 2023. A symbolic framework for systematic evaluation of mathematical reasoning with transformers. *arXiv preprint arXiv:2305.12563*.

Mistral AI. 2024. https://mistral.ai/news/mathstral/. Accessed: 2024-09-13.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, et al. 2022. No language left behind: Scaling human-centered machine translation.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. 2023. Openwebmath: An open dataset of high-quality mathematical web text.

Shuai Peng, Ke Yuan, Liangcai Gao, and Zhi Tang. 2021. Mathbert: A pre-trained model for mathematical formula understanding. *arXiv preprint arXiv:2105.00377*.

G. Polya. 1971. *How to Solve It*. Princeton University Press.

Michael Rescorla. 2024. The Language of Thought Hypothesis. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*, Summer 2024 edition. Metaphysics Research Lab, Stanford University.

Tomohiro Sawada, Daniel Paleka, Alexander Havrilla, Pranav Tadepalli, Paula Vidas, Alexander Kranias, John J Nay, Kshitij Gupta, and Aran Komatsuzaki. 2023. Arb: Advanced reasoning benchmark for large language models. *arXiv preprint arXiv:2307.13692*.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. 2022. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*.

Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. 2024. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482.

Wenxuan Wang, Wenxiang Jiao, Jingyuan Huang, Ruyi Dai, Jen-tse Huang, Zhaopeng Tu, and Michael R Lyu. 2023. Not all countries celebrate thanksgiving: On the cultural dominance in large language models. *arXiv preprint arXiv:2310.12481*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Tianwen Wei, Jian Luan, Wei Liu, Shuang Dong, and Bin Wang. 2023. Cmath: Can your language model pass chinese elementary school math test? *arXiv preprint arXiv:2306.16636*.

Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. 2024. Do llamas work in english? on the latent language of multilingual transformers. *arXiv preprint arXiv:2402.10588*.

An Yang, Baosong Yang, Binyuan Hui, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.

Wei Zhao, Mingyue Shang, Yang Liu, Liang Wang, and Jingming Liu. 2020. Ape210k: A large-scale and template-rich dataset of math word problems. *arXiv preprint arXiv:2009.11506*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

A   Appendix

Given the following mathematics problems in Romanian formatted in MathPix markdown, make a JSON with subject and solution pairs, removing unnecessary boilerplate and extra problem identifiers. The JSON must contain the full problem definition and subject number (e.g. subject 1b). Each sub-question must contain the whole problem definition for completeness. Each subject must be self-contained. Do not output anything else besides the required JSON. Do not modify the latex describing the mathematical formulas.
Example (truncated): "

PROBLEMS:

Se consideră matricea $A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ şi şirul $(F_n)_{n \geq 0}$ definit prin relaţia de recurenţă $F_{n+1} = F_n + F_{n-1}, n \in \mathbf{N}^*$, cu $F_0 = 0, F_1 = 1$.

a) Să se calculeze determinantul şi rangul matricei A.

b) Să se calculeze F_2 şi F_3.

SOLUTIONS:
a) $\det A = -1 \neq 0 \Rightarrow \operatorname{rang} A = 2$; b) $F_2 = 1, F_3 = 2$.
"

Example Output JSON:
[{
"subject": "1a",

"definition": "Se consideră matricea $A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ şi şirul $(F_n)_{n \geq 0}$ definit prin relaţia de recurenţă $F_{n+1} = F_n + F_{n-1}, n \in \mathbf{N}^*$, cu $F_0 = 0, F_1 = 1$. Să se calculeze determinantul şi rangul matricei A.",
"solution": "$\det A = -1 \neq 0 \Rightarrow \operatorname{rang} A = 2$"
},
{
"subject": "1b",

"definition": "Se consideră matricea $A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ şi şirul $(F_n)_{n \geq 0}$ definit prin relaţia de recurenţă $F_{n+1} = F_n + F_{n-1}, n \in \mathbf{N}^*$, cu $F_0 = 0, F_1 = 1$. Să se calculeze F_2 şi F_3.",
"solution": "$F_2 = 1, F_3 = 2$"
}
]
In this example, each sub-problem is self-contained and is paired with the appropriate solution. The sub-problem identifiers (e.g., "a)" and "b)") are stripped. The latex markdown is left intact.

Real Input:

Table 6: Claude 3 Sonnet prompt to format raw Markdown into structured JSON.

{
"role": "system",
"content": """Ești un student olimpic la matematică care a participat și câștigat multiple concursuri internaționale de matematică. Rolul tău este să rezolvi probleme de matematică de liceu și să oferi soluții complete și corecte. Problemele care necesită demonstrații trebuie rezolvate complet cu toți pașii intermediari necesari. Problemele care au un singur raspuns final trebuie furnizate într-un format încadrat ('\boxed'). Matematica trebuie scrisă în format LaTeX pentru a asigura claritatea și precizia soluțiilor. Textul in format LaTeX trebuie delimitat folosind simbolurile ' \(' și '\)'. Rezolvările incomplete sau incorecte vor fi evaluate cu scoruri mai mici. Asigură-te că răspunsurile sunt concise, fără prea multe explicații inutile.""",
},
# add few shot examples here: User => Problem statement, Assistant => Solution
{
"role": "user",
"content": """Care este rezolvarea următoarei probleme?

{problem_statement}""",
}

Table 7: Romanian prediction prompt.

{ "role": "system",
"content": """Asumă-ți rolul unui profesor de matematică responsabil cu evaluarea răspunsurilor studenților pentru o problemă de matematică în raport cu soluțiile corecte furnizate. Soluțiile pot include demonstrații, valori exacte, răspunsuri cu alegere multiplă sau aproximări numerice.

Criterii de Evaluare:
1. **Echivalență Matematică**: Evaluează răspunsurile pe baza echivalenței matematice, nu doar a preciziei numerice. Verifică dacă diferite expresii algebrice sau simbolice sunt echivalente. Asigură-te că sunt echivalente precum \(\frac{\sqrt{6}-\sqrt{2}}{2} \) fiind echivalent cu \(\sqrt{2 - \sqrt{3}} \).
2. **Scor**: Atribuie un scor de '1' pentru orice răspuns care se potrivește sau este echivalent cu soluția furnizată, fie că este o valoare exactă, o variantă de răspuns (de exemplu, A, B, C) sau o aproximare numerică corect rotunjită. Atribuie un scor de '0' pentru răspunsuri incorecte. Nu furniza niciun fel de explicație.
3. **Tratarea Alegerii Multiple**: Dacă soluția furnizată este o variantă de răspuns (de exemplu, A, B, C, D, E, F) și studentul identifică această alegere corect, trateaz-o ca fiind corectă. Dacă soluția este o valoare exactă și studentul furnizează alegerea corespunzătoare care reflectă corect această valoare în conformitate cu contextul problemei, tratează-o de asemenea ca fiind corectă.
4. **Echivalență Numerică**: Tratează răspunsurile numerice ca fiind echivalente dacă sunt corecte cu cel puțin două zecimale sau mai mult, în funcție de precizia furnizată în soluție. De exemplu, atât 0.913, cât și 0.91 ar trebui acceptate dacă soluția este exactă cu două zecimale.
5. **Identități Algebrice și Simbolice**: Recunoaște și acceptă forme algebrice echivalente, cum ar fi \(\sin^2(x) + \cos^2(x) = 1 \) sau \(e^{i\pi} + 1 = 0 \), ca fiind corecte.
6. **Forme Trigonometrice și Logaritmice**: Acceptă expresii trigonometrice și logaritmice echivalente, recunoscând identități și transformări care ar putea modifica forma, dar nu și valoarea.
7. **Demonstrații Matematice**: Evaluează demonstrațiile matematice pe baza corectitudinii și a logicii, nu a stilului sau a formei. Asigură-te că demonstrațiile sunt complete și corecte, chiar dacă sunt prezentate într-un mod diferit de soluția furnizată.

Formatul Așteptat al Răspunsului: Prezintă răspunsul final cu un scor doar de '0' sau '1', unde '0' semnifică o soluție greșită, iar '1' semnifică o soluție corectă. Nu include nicio altă informație sau explicații suplimentare în răspuns.

Problema de matematică este:
{question}.

Soluția corectă din baremul de corectare este:
{true}.

Te rog să evaluezi soluția studentului cu precizie pentru a asigura o evaluare exactă și corectă."""
},
{
"role": "user", "content": "Soluția studentului este {prediction}. Furnizează un doar scor de '0' sau '1', unde '0' semnifică o soluție greșită, iar '1' semnifică o soluție corectă. Bazează-ți evaluarea pe criteriile de evaluare furnizate si pe soluția corecta din barem.",
}

Table 8: Romanian judge prompt.

{
"role": "system",
"content": """Assume the role of a math teacher responsible for evaluating student responses for a math problem against the provided correct solutions. Solutions may include proofs, exact values, multiple-choice answers, or numerical approximations.

Evaluation Criteria:
1. **Mathematical Equivalence**: Evaluate answers based on mathematical equivalence, not just numerical accuracy. Check if different algebraic or symbolic expressions are equivalent. Ensure that there are equivalences such as $\frac{\sqrt{6}-\sqrt{2}}{2}$ being equivalent to $\sqrt{2 - \sqrt{3}}$.
2. **Scoring**: Assign a score of '1' for any answer that matches or is equivalent to the provided solution, whether it is an exact value, a choice label (e.g., A, B, C), or a correctly rounded numerical approximation. Assign a score of '0' for incorrect answers. Do not provide any explanatory feedback.
3. **Handling Multiple Choices**: If the solution provided is a choice (e.g., A, B, C, D, E, F) and the student identifies this choice correctly, treat it as correct. If the solution is an exact value and the student provides the corresponding choice that reflects this value correctly according to the problem's context, also treat it as correct.
4. **Numerical Equivalence**: Treat numerical answers as equivalent if they are correct to at least two decimal places or more, depending on the precision provided in the solution. For instance, both 0.913 and 0.91 should be accepted if the solution is accurate within two decimal places.
5. **Symbolic and Algebraic Identities**: Recognize and accept equivalent algebraic forms, such as $\sin^2(x) + \cos^2(x) = 1$ or $e^{i\pi} + 1 = 0$, as correct.
6. **Trigonometric and Logarithmic Forms**: Accept equivalent trigonometric and logarithmic expressions, acknowledging identities and transformations that might alter the form but not the value.
7. **Mathematical Proofs**: Evaluate mathematical proofs based on correctness and logic, not style or form. Ensure proofs are complete and correct, even if presented differently from the provided solution.
Expected Output Format: Present your final answer with a score of '0' or '1' only, where '0' signifies an incorrect solution, and '1' signifies a correct solution. Do not include any additional information or feedback in your response.

The math problem is:
{question}.

The correct solution from the answer key is:
{true}.

Please evaluate the student's response with precision to ensure accurate and fair grading."""
},
{
"role": "user",
"content": "The student answer is prediction. Provide a score of '0' or '1' only, where '0' signifies an incorrect solution, and '1' signifies a correct solution. Base your evaluation on the provided evaluation criteria and the correct solution from the answer key.",
}

Table 9: English judge prompt.

RoMath-Baccalaureate

deepseek-ai/deepseek-math-7b-instruct

***Problem Statement*:** Să se calculeze modulul numărului complex $\sqrt{3} + i$.

***Solution*:** 2

***Generated Solution*:** Pentru a calcula modulul unui număr complex, folosim formula $|z| = \sqrt{a^2 + b^2}$, unde $z = a + bi$.

În cazul nostru, $a = \sqrt{3}$ și $b = 1$.

Aplicând formula, obținem:

$|z| = \sqrt{(\sqrt{3})^2 + 1^2} = \sqrt{3 + 1} = \sqrt{4} = 2$.

Deci, modulul numărului complex $\sqrt{3} + i$ este 2.

Răspuns: $\boxed{2}$.

OpenLLM-Ro/RoLlama3-8b-Instruct

***Problem Statement*:** Să se demonstreze că vectorii $\vec{u} = -2\vec{i} + 3\vec{j}$ și $\vec{v} = 3\vec{i} + 2\vec{j}$ sunt perpendiculari.

***Solution*:** Din $\vec{a} \cdot \vec{b} = -2 \cdot 3 + 3 \cdot 2 = 0 \Rightarrow \vec{a} \perp \vec{b}$

***Generated Solution*:** Pentru a demonstra că vectorii $\vec{u} = -2\vec{i} + 3\vec{j}$ și $\vec{v} = 3\vec{i} + 2\vec{j}$ sunt perpendiculari, trebuie să arătăm că produsul lor punct este egal cu 0.

Produsul punct al celor doi vectori $\vec{u}$ și $\vec{v}$ este dat de:

$\vec{u} \cdot \vec{v} = (-2\vec{i} + 3\vec{j}) \cdot (3\vec{i} + 2\vec{j})$

Acum, putem calcula produsul punct:

$\vec{u} \cdot \vec{v} = (-2)(3) + (3)(2) = -6 + 6 = 0$

Deoarece produsul punct al celor doi vectori este egal cu 0, vectorii $\vec{u}$ și $\vec{v}$ sunt perpendiculari.

RoMath-Competitions

deepseek-ai/deepseek-math-7b-instruct

***Problem Statement*:** Într-o clasă sunt 22 de elevi, dintre care 12 sunt fete. Să se determine în câte moduri se poate alege un comitet al clasei format din 3 fete și 2 băieți.

***Solution*:** Numărul băieților este $22 - 10 = 12$. Fetele pot fi alese în $C_{12}^3 = 220$ moduri. Băieții pot fi aleși în $C_{10}^2 = 45$ moduri. Comitetul format din 3 fete și 2 băieți dintre cei 22 de elevi ai clasei poate fi format în $C_{12}^3 \cdot C_{10}^2 = 9900$ moduri.

***Generated Solution*:** Pentru a rezolva această problemă, trebuie să utilizăm formula combinării. Combinarea este o modalitate de a alege un set de elemente dintr-o mulțime mai mare, fără a uita la ordinea elementelor.

În acest caz, trebuie să alegem 3 fete din 12 și 2 băieți din 10 (deoarece 22 - 12 = 10).

Formula combinării este:

$$C(n, k) = \frac{n!}{k!(n - k)!}$$

unde n este numărul total de elemente, k este numărul de elemente ce trebuie alese, iar ""!"" semnifică factorialul unui număr.

Pentru a calcula numărul de moduri în care putem alege 3 fete din 12, aplicăm formula:

$$C(12, 3) = \frac{12!}{3!(12 - 3)!} = \frac{12!}{3!9!} = \frac{12 \times 11 \times 10}{3 \times 2 \times 1} = 220$$

Pentru a calcula numărul de moduri în care putem alege 2 băieți din 10, aplicăm formula:

$$C(10, 2) = \frac{10!}{2!(10 - 2)!} = \frac{10!}{2!8!} = \frac{10 \times 9}{2 \times 1} = 45$$

Pentru a calcula numărul total de moduri în care putem forma comitetul, înmulțim cele două rezultate:

$$220 \times 45 = 9900$$

Deci, există 9900 de moduri în care se poate forma comitetul.

Răspunsul final este $\boxed{9900}$.

OpenLLM-Ro/RoLlama3-8b-Instruct

***Problem Statement*:** Un grădinar a plantat într-una dintre grădinile pe care le îngrijește parcele cu tufe de trandafir din soiuri distincte, astfel încât fiecare parcelă conține trandafiri dintr-un alt soi. Într-o zi are de realizat un aranjament floral din acești trandafiri. Analizează tufele și procedează în felul următor: din prima parcelă taie trei trandafiri, din cea de a doua taie de două ori mai mulți decât din prima ș.a.m.d., tăind dintr-o parcelă de două ori mai mulți trandafiri decât din parcela precedentă. Care este numărul minim de trandafiri pe care trebuie să-l aibă parcela a IX-a pentru a putea fi folosită la realizarea aranjamentului floral?

***Solution*:** Parcela a IX a trebuie să aibă cel puțin: $3 \cdot 2^8 = 3 \cdot 256 = 768$

***Generated Solution*:** Să notăm numărul de trandafiri din fiecare parcelă ca P1, P2, P3,..., P9. Conform problemei, numărul de trandafiri din fiecare parcelă este dat de:

P1 = 3
P2 = 2 * P1 = 2 * 3 = 6
P3 = 2 * P2 = 2 * 6 = 12
P4 = 2 * P3 = 2 * 12 = 24
P5 = 2 * P4 = 2 * 24 = 48
P6 = 2 * P5 = 2 * 48 = 96
P7 = 2 * P6 = 2 * 96 = 192
P8 = 2 * P7 = 2 * 192 = 384
P9 = 2 * P8 = 2 * 384 = 768
Deci, numărul minim de trandafiri din parcela a IX-a este 768.

Table 10: Qualitative examples of correct zero-shot predictions for RoMath-Baccalaureate and RoMath-Competitions.

RoMath-Baccalaureate

deepseek-ai/deepseek-math-7b-instruct
Problem Statement: Să se rezolve în mulțimea numerelor reale ecuația $2^x = 4^x$.
Solution: x=0
Generated Solution: Pentru a rezolva ecuația $2^x = 4^x$, putem scrie 4^x ca $(2^2)^x$, adică 2^{2x}.
Astfel, ecuația devine $2^x = 2^{2x}$.
Prin urmare, putem scrie $x = 2x$.
Împărțind ambii membri ai ecuației cu x, obținem $1 = 2$.
Această ecuație nu are soluții în mulțimea numerelor reale, deoarece 1 nu este egal cu 2.
Deci, ecuația $2^x = 4^x$ nu are soluții în mulțimea numerelor reale.

Răspuns: $\boxed{\text{Nu are soluții}}$

OpenLLM-Ro/RoLlama3-8b-Instruct
Problem Statement: Să se calculeze determinantul $\begin{vmatrix} 1 & 3 & 5 \\ 7 & 9 & 11 \\ 2 & 4 & 6 \end{vmatrix}$.
Solution: 0
Generated Solution: Pentru a calcula determinantul matricei date, putem folosi metoda de expansiune prin rânduri. Să calculăm determinantul:
$\begin{vmatrix} 1 & 3 & 5 \\ 7 & 9 & 11 \\ 2 & 4 & 6 \end{vmatrix}$
În primul rând, putem elimina primul rând și primul element din al doilea rând:
$\begin{vmatrix} 9 & 11 \\ 2 & 4 \end{vmatrix}$
Acum, putem calcula determinantul matricei 2x2 rămase:
$\begin{vmatrix} 9 & 11 \\ 2 & 4 \end{vmatrix} = 9 \cdot 4 - 11 \cdot 2 = 36 - 22 = 14$
Deci, determinantul matricei date este 14.

RoMath-Competitions

deepseek-ai/deepseek-math-7b-instruct

Problem Statement: Într-un plan, raportat la reperul ortogonal de axe de coordonate $(xxOy)$ se dau punctele: A$(0; 6)$; B$(5; 4)$; C$(-1; 4)$. Pentru a = 5 să se scrie ecuația medianei corespunzătoare laturii BC.

Solution: Determină coordonatele mijlocului lui [BC] notat cu M : $x_M = \frac{5-1}{2} = 2$, $y_M = \frac{4+4}{2} = 4 \Leftrightarrow$ M$(2, 4)$ Ecuația dreptei (AM): $\begin{vmatrix} x & y & 1 \\ 0 & 6 & 1 \\ 2 & 4 & 1 \end{vmatrix} = 0 \Rightarrow AM:$
$x + y - 6 = 0$.
Generated Solution: Pentru a scrie ecuația medianei corespunzătoare laturii BC, trebuie să găsim mijlocul segmentului BC, notat M.
Mijlocul unui segment este dat de media aritmetică a coordonatelor punctelor de capăt ale segmentului. Astfel, coordonatele punctului M sunt:
M(xM; yM) = ((xB + xC) / 2; (yB + yC) / 2)
M(-1 + 5 / 2; 4 + 4 / 2) = M(2; 4)
Mediana AM este dreapta care unește punctul A cu mijlocul M al segmentului BC.
Ecuația unei drepte în plan, cunoscând coordonatele a două puncte A(x1; y1) și B(x2; y2), este: (y - y1) / (y2 - y1) = (x - x1) / (x2 - x1)
Aplicând formula pentru mediana AM, avem: (y - 6) / (4 - 6) = (x - 0) / (2 - 0)
Simplificând, obținem: (y - 6) / (-2) = x / 2
Ecuația medianei AM este: y - 6 = -2x sau 2x + y - 6 = 0
Pentru a = 5, ecuația medianei corespunzătoare laturii BC este: 2x + y - 6 = 0

OpenLLM-Ro/RoLlama3-8b-Instruct
Problem Statement: În raport cu un reper cartezian xOy, considerăm un purice P care sare doar în puncte având ambele coordonate întregi. La o săritură, puricele se deplasează doar pe verticală sau pe orizontală. După o săritură nu este obligatoriu să schimbe direcția de deplasare, dar respectă următoarea regulă: sare 3 unități, apoi 2 unități, 3 unități, 2 unități etc. De exemplu, dacă M se află inițial în punctul $(1, 2)$, la prima mutare ar putea fi în punctul $(1, 5)$, iar apoi în $(1, 7)$ dacă păstrează direcția, sau în ($3,5$), dacă își schimbă direcția. Dacă P se află inițial în origine, demonstrați că poate ajunge în punctul $(2013, 0)$.
Solution: Puricele se poate deplasa doar orizontal, astfel: după două sărituri ajunge în punctul $(5, 0)$, după patru ajunge în $(10, 0)$ și, tot așa, după 804 sărituri ajunge în $(2010, 0)$. La următoarea săritură va ajunge în $(2013, 0)$.
Generated Solution: Pentru a demonstra că puricele poate ajunge în punctul $(2013, 0)$, vom arăta că poate ajunge în orice punct cu coordonate întregi prin repetarea ciclului de deplasare.
Să notăm deplasarea verticală ca v și deplasarea orizontală ca h. Ciclul de deplasare este dat de secvența $(v, h, v, h, v, h, \dots)$, unde $v = 3$ și $h = 2$. Aceasta înseamnă că puricele se deplasează în sus 3 unități, apoi în dreapta 2 unități, apoi în sus 3 unități, apoi în dreapta 2 unități, și așa mai departe.
Acum, să luăm în considerare punctul $(2013, 0)$. Deoarece puricele poate sări doar în puncte cu coordonate întregi, trebuie să găsim o secvență de deplasări care să conducă la acest punct.
Putem observa că ciclul de deplasare dat poate fi repetat de 2013 ori pentru a ajunge la punctul $(2013, 0)$.
Pentru a vedea acest lucru, să luăm în considerare deplasările necesare pentru a ajunge la punctul $(2013, 0)$ din origine:
1. Deplasare verticală de 3 unități (de la $(0, 0)$ la $(0, 3)$).
2. Deplasare orizontală de 2 unități (de la $(0, 3)$ la $(2, 3)$).
3. Deplasare verticală de 3 unități (de la $(2, 3)$ la $(2, 6)$).
4. Deplasare orizontală de 2 unități (de la $(2, 6)$ la $(4, 6)$).
5. Deplasare verticală de 3 unități (de la $(4, 6)$ la $(4, 9)$).
6. Deplasare orizontală de 2 unități (de la $(4, 9)$ la $(6, 9)$).
7. Deplasare verticală de 3 unități (de la $(6, 9)$ la $(6, 12)$).
8. Deplasare orizontală de 2 unități (de la $(6, 12)$ la $(8, 12)$).
9. Deplasare verticală de 3 unități (de la $(8, 12)$ la $(8, 15)$).
10. Deplasare orizontală de 2 unități (de la $(8, 15)$ la $(10, 15)$).
Putem continua acest proces până când ajungem la punctul $(2013, 0)$. Deoarece ciclul de deplasare poate fi repetat de 2013 ori, puricele poate ajunge în punctul $(2013, 0)$ după efectuarea ciclului de deplasare de 2013 ori.

Table 11: Qualitative examples of incorrect zero-shot predictions for RoMath-Baccalaureate and RoMath-Competitions.