LILA: A Unified Benchmark for Mathematical Reasoning

Swaroop Mishra∗†
Arizona State University

Matthew Finlayson∗‡
The Allen Institute for AI

Pan Lu†
UCLA

Leonard Tang
Harvard University

Chitta Baral
Arizona State University

Oyvind Tafjord
The Allen Institute for AI

Tanmay Rajpurohit
Georgia Institute of Technology

Leonard Tang
Harvard University

Sean Welleck
The Allen Institute for AI

Chitta Baral
Arizona State University

Peter Clark
The Allen Institute for AI

Ashish Sabharwal
The Allen Institute for AI

Ashwin Kalyan‡
The Allen Institute for AI

Abstract

Mathematical reasoning skills are essential for general-purpose intelligent systems to perform tasks from grocery shopping to climate modeling. Towards evaluating and improving AI systems in this domain, we propose LILA, a unified mathematical reasoning benchmark consisting of 23 diverse tasks along four dimensions: (i) mathematical abilities e.g., arithmetic, calculus (ii) language format e.g., question-answering, fill-in-the-blanks (iii) language diversity e.g., no language, simple language (iv) external knowledge e.g., commonsense, physics. We construct our benchmark by extending 20 datasets benchmark by collecting task instructions and solutions in the form of Python programs, thereby obtaining explainable solutions in addition to the correct answer. We additionally introduce two evaluation datasets to measure out-of-distribution performance and robustness to language perturbation. Finally, we introduce BHASKARA, a general-purpose mathematical reasoning model trained on LILA. Importantly, we find that multi-tasking leads to significant improvements (average relative improvement of 21.83% F1 score vs. single-task models), while the best performing model only obtains 60.40%, indicating the room for improvement in general mathematical reasoning and understanding.1

1 Introduction

Mathematical reasoning is required in all aspects of life, from buying ingredients for a recipe to controlling the world economy. Given the fundamental nature of mathematical reasoning, a number of works propose datasets to evaluate specific mathematical reasoning abilities of AI agents, e.g., Kushman et al. (2014) (algebra word problems), Mishra et al. (2022c) (arithmetic reasoning), Saxton et al. (2019) (templated math reasoning spanning algebra, calculus, probability, etc.) Since evaluating high-capacity models on narrowly scoped mathematical reasoning datasets risks overestimating the reasoning abilities of these AI systems, creating the need for a unified benchmark for systematic evaluation over diverse topics and problem styles.

Figure 1: A data example with two Python programs in LILA. One program annotation uses function construct whereas the other one is a plain script without function. The instruction for each task and categories across four dimensions are annotated for developing LILA.
To this end, we introduce LILA², a unified mathematical reasoning benchmark that consists of 23 mathematical reasoning tasks. LILA is constructed by extending 20 existing datasets spanning a wide range of topics in mathematics, varying degrees of linguistic complexity, and diverse question formats and background knowledge requirements. Importantly, LILA extends all of these datasets to include a solution program as opposed to only an answer, and instruction annotations to enable instruction-based learning (Sanh et al., 2021; Wei et al., 2021; Mishra et al., 2022b).

In order to accurately assess the mathematical reasoning ability of models, evaluating the chain of reasoning that leads to the correct solution is equally important (if not more important) to evaluating the final answer or expression. We therefore collect Python programs that serve as reasoning chains for each question in the benchmark. We achieve this by automatically converting domain-specific language (DSL) annotations into Python programs and by manually collecting expert annotations when no DSL annotations are available. By incorporating program annotations, LILA unifies various mathematical reasoning datasets under a single problem formulation i.e., given an input problem in natural language, generate a Python program that upon execution returns the desired answer. This formulation allows neural approaches to focus on the high-level aspects of mathematical problem solving (e.g., identifying potential solution strategies, decomposing the problem into simpler sub-problems), while leveraging external solvers (e.g., Python builtins, Sympy) to perform precise operations like adding huge numbers or simplifying expressions. Figure 1 illustrates a sample from our LILA benchmark that illustrates the question, answer, program, instructions, and category tags.

In addition to evaluating high-level problem solving, we also facilitate two other key ways to make a fair assessment of models on mathematical reasoning tasks. In line with Bras et al. (2020), Ribeiro et al. (2020) and Welleck et al. (2022), we evaluate generalization e.g., alternate formulations of a problem (“2+2=” vs. “What is two plus two?”) using an out-of-distribution evaluation set (LILA-OOD) containing datasets requiring the same underlying mathematical reasoning skills, but were collected independently of the training datasets. Further, we collect a robustness split LILA-ROBUST, that introduces linguistic perturbations (e.g., active vs. passive voice) via crowd-sourcing. The evaluation scheme is a combination of the performance on all three sets: LILA-TEST, LILA-OOD and LILA-ROBUST.

Contributions

1. We present LILA, a holistic benchmark for mathematical reasoning. LILA extends 20 existing datasets with solutions in the form of Python programs and instruction annotations, and categorizes questions into 23 tasks based on their language complexity, question format and need for external knowledge. Our benchmark measures performance on out-of-distribution examples and robustness to language perturbations in addition to standard test-set.

2. We introduce BHASKARA, a multi-task model fine-tuned on our dataset. Our best-performing model achieves comparable performance to a 66× larger model pre-trained on both code and language.

3. We provide an analysis of our models’ performance and find that (1) multitasking improves considerably over task-specific learning both in in-distribution and out-of-distribution evaluation (2) program synthesis substantially outperforms answer prediction, (3) few-shot prompting with codex has the strongest performance. We also identify areas for improvement for future work, e.g., data gaps in LILA categories.

2 Related Work

Mathematical Reasoning Datasets. Our work builds on an existing body of mathematical reasoning literature. Early work in this areas focuses on small-scale datasets testing addition-subtraction (Hosseini et al., 2014), templated questions with equations as parameters (Kushman et al., 2014) and other forms of arithmetic reasoning (Koncel-Kedziorski et al., 2015; Roy and Roth, 2016; Upadhyay et al., 2016; Roy and Roth, 2017, 2018; Ling et al., 2017). Later datasets increase in complexity and scale, incorporating reading comprehension (Dua et al., 2019b), algebra (Saxton et al., 2019), and multi-modal contexts (Lu et al., 2021a, 2022). Still other numerical reason-
ing datasets focus on diversity (Miao et al., 2020a) with multiple categories of numerical reasoning tasks (e.g., Amini et al., 2019). Most recently, new datasets have focused on increasing difficulty, e.g., olympiad problems (Hendrycks et al., 2021b) and adversarial problems (Patel et al., 2021), as well as increasing the knowledge requirements to solve tasks, with a growing focus on commonsense reasoning (Zhou et al., 2019; Zhang et al.; Lu et al., 2021b; Mishra et al., 2022c).

A separate line of work in mathematical reasoning includes datasets testing mathematical theorem proving (e.g., Li et al., 2021; Wu et al., 2021; Welleck et al., 2021; Zheng et al., 2021; Han et al., 2021). We do not, however, consider theorem proving in our work, choosing instead to focus on numerical reasoning.

Task Hierarchy and Multi-tasking in Numerical Reasoning. We take inspiration from the success of multi-task learning in NLP (Weston et al., 2015), including benchmarks (e.g., Wang et al., 2018, 2019; Dua et al., 2019a) and multitasking models (e.g., McCann et al., 2018; Khashabi et al., 2020; Lourie et al., 2021; Aghajanyan et al., 2021). NumGLUE (Mishra et al., 2022c) has been proposed as a multi-tasking numerical reasoning benchmark that contains 8 different tasks. LILA expands NumGLUE to provide wider coverage of mathematical abilities, along with evaluation that captures out-of-domain, robustness, and instruction-following performance. Our introduction of mathematical reasoning categories and the evaluation setup is inspired by task hierarchies in other domains such as vision (Zamir et al., 2018) and NLP (Rogers et al., 2021) which appear in large scale benchmarks (e.g., Srivastava et al., 2022; Wang et al., 2022).

3 LILA

LILA is composed of 23 tasks across 4 dimensions, curated from 44 sub-datasets across 20 dataset sources. Here we discuss the construction and composition of the benchmark and provide descriptive statistics of the datasets.

3.1 Dataset Construction

Data Sources. LILA incorporates 20 existing datasets from the mathematical reasoning literature (Table 22 gives a detailed list), where inputs are natural language or templated text and outputs are numerical or expressions, e.g., we exclude theorem proving (Welleck et al., 2021; Han et al., 2021), where the output is not a number or expression. We leave the incorporation of formats like theorem proving to future work.

Unified format. We normalize all datasets to a unified format with the following fields:

1. The source dataset. Category tags for each of the four dimensions (math ability, language complexity, format, and external knowledge; see §3.2).
2. The question, in English.
3. The answer to the question, as a string containing a number, expression, list, or other data format. A set of Python strings that print the answer.
4. A task-level instruction in natural language.

We also retain meta-data from the original dataset.

Automatic program annotation. Most of the annotations in the source datasets do not contain output in the form of a Python program. We automatically annotate most datasets by generating Python programs using the annotations (answer, explanation, etc.) provided in the source datasets. Where possible, we generate multiple Python programs for a single question. This is to account for variation in the program space such as the choice of data structure, language construct, variable name, and programming style (e.g., declarative vs procedural). For example, Figure 1 gives multiple Python programs solving the same question; in this case one program directly calculates the answer, whereas the other defines a function to solve the problem more generally.

Some datasets contain program annotations that can be captured by a domain-specific language (DSL) in which case we write rules to convert them into Python programs, e.g., volume(sphere, 3) to the Python expression \(4/3*\text{math.pi}*3^3\). In some cases where a DSL annotation is not provided, we use pattern matching to convert highly templated datasets like the AMPS dataset (Hendrycks et al., 2021b) to our unified format. In other cases, instead of converting the existing dataset, we modify the data generation code to reproduce the dataset with program annotations. For the DeepMind mathematics dataset (Saxton et al., 2019), this allows us to create diverse, compositional math problems with program annotations using a sophisticated grammar.
Expert program annotation. For many datasets, it is not possible to obtain Python program annotations via automated methods described above; either the original dataset contains only the final answer or contains solutions expressed in free-form natural language. For such datasets, we obtain annotations from experts who are proficient in basic programming and high-school level mathematics. See Appendix B.1 for details.

Instruction annotation. Given the effectiveness of instruction learning (Mishra et al., 2022b; Wei et al., 2021; Mishra et al., 2022a; Sanh et al., 2021) for effective generalization, we collect instruction annotation for each task. Each instruction contains a definition that clearly defines the task and provides guidelines, a prompt that provides a short and straightforward instruction, and examples that facilitate learning by demonstration (Brown et al., 2020). Figure 1 shows an example instruction for the basic math task (§3.2).

3.2 Categories and Tasks
We create 4 views\(^3\) or categories of LILA along the dimensions of mathematical area, language complexity, external knowledge, and question format. Altogether, these views classify the data into 23 tasks (Table 1). By creating multiple views of the benchmark, we are able to systematically characterize the strengths and weaknesses of existing models at a granular level.

The first category, math ability, partitions the datasets into common pedagogical subjects: arithmetic, algebra, geometry, calculus, etc.

Our second category, language complexity, separates math problems by the complexity of the language used to represent them. This ranges from formal representations only (e.g., $1+1=?$) to natural language (e.g., “Mariella has 3 pears...”).

We next partition datasets based on the type of background knowledge, required to solve the problem. For instance, commonsense questions like “How many legs to 3 people have?” or science questions like “Will water boil at 200 degrees Celsius?” require different sets of knowledge to answer.

Lastly, we categorize based on question format, putting e.g., multiple choice questions under one task and natural language inference under another. Examples of each task and the datasets included are in Appendix B.

3.3 LILA-OOD
In order to measure if the model has truly learned the underlying mathematical reasoning skill, we evaluate both in-distribution (IID, i.e., standard train-test splits) and out-of-distribution (OOD) performance for each task, i.e., we evaluate on examples requiring the same underlying mathematical reasoning skill but from a different dataset. To construct LILA-OOD, we follow Bras et al. (2020) and Hendrycks et al. (2020) by randomly assigning the datasets for each task into IID and an OOD sets, using the IID set for training and standard evaluation and the OOD set to evaluate generalization. We do not include tasks in LILA-OOD for tasks containing only one dataset.

3.4 LILA-ROBUST
In light of recent work demonstrating the brittleness of language models at solving math problems (Patel et al., 2021), we create a high-quality evaluation dataset, LILA-ROBUST, to evaluate performance on mathematical reasoning tasks when linguistic perturbations are introduced. Specifically, we define and apply a set of carefully chosen augmentation templates, summarized in Table 19, on each task, yielding a set of challenging problems that are consistent answer-wise but stylistically different question-wise. Overall, we define a total of 9 templates for such question perturbations: 3 from Patel et al. (2021) and 6 of our own. From each constituent dataset, we sample 20 questions and obtain perturbed question annotations via Amazon.

| Category       | Tasks                                                                 |
|----------------|------------------------------------------------------------------------|
| Math ability   | Basic math, multiplication/division, number theory, algebra, geometry, counting and statistics, calculus, linear algebra, advanced math |
| Language       | No language, simple language, complex language                         |
| Knowledge      | No background knowledge, commonsense, math, science, computer science, real world knowledge |
| Format         | Fill-in-the-blank, generative question answering, multiple-choice, natural language inference, reading comprehension |

Table 1: Categories and their associated tasks.
### 3.5 Statistics

Table 2 shows key statistics of our proposed benchmark, LILA. LILA contains \( \approx 134\)K examples with significant diversity across question, answer, program and instruction length (see detailed statistics in Appendix C). Figure 2 shows the diversity of questions in LILA. Note that we downsample (via random selection) some datasets like AMPS (Hendrycks et al., 2021b) which contains numerous templated questions that can get over-represented in the distribution of examples across categories in LILA.

### 4 Experiments

In this section, we introduce our modeling contributions for the LILA benchmark and discuss the overall experimental setup.

---

**Data partition and evaluation.** For the IID setup, we randomly partition the data in each task into training (70%), development (10%) and test (20%) sets. Additionally, we also evaluate on LILA-OOD and LILA-ROBUST settings; thus, the final evaluation scheme is a combination of the performance on all three evaluation setups.

**Fine-tuning** We fine-tune a series of GPT-Neo-2.7B causal language models (Black et al., 2021) on LILA. We choose GPT-Neo because it was pre-trained on both natural language and code (Gao et al., 2020), as opposed to solely on natural language. To assess the capabilities of GPT-Neo on various aspects of the dataset, we fine-tune single-task models on each of the 23 tasks in LILA. We also evaluate the benefit of transfer learning by fine-tuning a single multi-task GPT-Neo baseline on all the tasks simultaneously.

**Prompting.** We also use few-shot prompting to evaluate GPT-3 and Codex (Brown et al., 2020; Chen et al., 2021). For the IID setting, we prompt the model with a random input-output examples from the same dataset as the input. In the OOD setting, we take examples from other datasets (Table 15-18) within the same task. We repeat this evaluation with increasing numbers of examples (up to the token size of models) to study the effect on performance.

**Evaluation.** We evaluate our models under two regimes—directly outputting the answer i.e., program induction and outputting a Python program that is then executed to obtain the final answer i.e., program synthesis. In the case of our fine-tuned models, we train them to output both the final answer and the Python program conditioned on the input question. To evaluate our models under direct question answering, we use F1-score to compare the model output and the gold answer. To evaluate program synthesis, we execute the model’s output within a Python interpreter and compare the program output with the output of the gold program, again using F1. We evaluate based on the program output, rather than the program itself, to account for diversity in solving techniques and programming styles.

---

**Table 2: Key statistics of LILA.**

| Statistic            | Number   |
|----------------------|----------|
| # Total tasks        | 23       |
| # Total datasets     | 44       |
| # Total instructions | 44       |
| # Total questions    | 133,815  |
| # Total programs     | 358,769  |
| Unique questions     | 132,239  |
| Unique programs      | 325,597  |
| Unique answers       | 271,264  |
| Average length of instructions | 31.18 |
| Average length of questions | 47.72 |
| Average length of programs | 47.85 |

**Figure 2:** Question n-gram distribution in LILA.

---

Mechanical Turk (AMT). Refer to Appendix B.1 for additional details on the construction of LILA-ROBUST.

---

4text-davinci-002, code-davinci-002

5Henceforth we refer to the max example model unless otherwise specified.

6This is a soft version of exact match accuracy assigning partial credit when common words are present in the output and gold answer.
Table 3: Evaluations of different baselines across 23 tasks in LiLa. On most tasks, Codex outperforms all baselines while Neo-Multi-P outperforms all fine-tuned baselines. A model usually performs worse on the OOD data set. The **bold** score refers to the best score among models with the same supervision/size method; the underlined score refers to the best score among all models. GPT-3 and Codex performance is computed on 100 uniformly distributed examples owing to their cost and usage limit. Fine-tuned model performance is calculated on the full test set.

5 Results and Analysis

A summary of all key results on our LiLa benchmark are shown in Table 3. In this section, we will discuss the performance of fine-tuned 2.7B GPT-Neo models (§5.1), performance of models along the 4 categories of tasks (§5.2) and finally, the few-shot performance of much larger (~175B parameters) models (§5.3).

### 5.1 Results: Fine-tuned Models

Multitasking improves IID performance, robustness, and OOD generalization. The multitasking model (Neo-Multi) substantially improves upon the single task models (Neo). Neo-Multi achieves better average in-domain performance than the 23 individual per-task models (0.480 vs. 0.394 average score), suggesting that it leverages cross-task structure not present in a single task’s training set. We also find that our multi-task model is robust to the linguistic perturbations we test in LiLa-ROBUST. That is to say, we did not find any degradation in performance when testing on perturbed IID test examples.

Additionally, multi-task training substantially improves out-of-domain generalization (0.448 vs. 0.238). The gap between IID and OOD performance is much smaller for Neo-Multi than for the single task models (Table 3), and in one case (for-form program output (Neo-A) always outperforms number output (Neo-P)).
Program synthesis substantially outperforms answer prediction. Synthesizing the program and evaluating it to get an answer substantially outperforms directly predicting the answer. For instance, multi-task program synthesis (Neo-Multi-P) has an average score of 0.480 while multi-task answer prediction (Neo-Multi-A) scores 0.252. This means models are often able to generate a program that evaluates to the correct answer, even when the model cannot directly compute the answer.

Program synthesis improves over answer prediction in all math categories except Geometry, with the largest improvements in Statistics and Linear Algebra; see Table 8 for examples. We even see benefits of program synthesis in NLI, a classification-based task. LILA’s unified problem format decouples synthesis from computation, while opening directions for further study on either aspect.

Models leverage symbolic execution and libraries. The gap between program synthesis and answer prediction suggests that the neural language model offloads computations to the symbolic Python runtime that are otherwise difficult to compute directly. We identify two common cases. First, the model leverages standard Python as a calculator. For instance, this pattern is common in the basic_math and mul_div categories, which involve evaluating arithmetic expressions; Table 7 shows examples. Second, the model is able to call external libraries that perform sophisticated computations. For instance, in statistics the model uses scipy.stats.entropy or np.linalg.det in linear algebra while solving problems (Table 8).

Models occasionally generate non-executable code. Roughly 10% of Neo-Multi’s IID programs fail to execute. 86% of these are SyntaxErrors, which often occur because decoding terminates before finishing the program or the model generates a program of the form ‘2+3=5’, which is invalid Python. The remaining 14% of execution failures are less trivial, including NameErrors (7%) and TypeErrors (1%) (see Table 9).

Our Neo-Multi model is a good starting point for further fine-tuning Table 5 shows that our Neo-Multi model is a better starting point for downstream fine-tuning than the vanilla pre-trained GPT-Neo-2.7B. When comparing fine-tuning for direct question answering with T5-3B, we see an almost 8% absolute improvement in F1 (30.1% to 37.6%).

| Data    | Answer (% F1) | Program (% F1) |
|---------|---------------|----------------|
|         | Neo  | Multi | Δ     | Neo  | Multi | Δ     |
| 100%    | 28.4 | 32.3  | +4.0 | 80.0 | 82.4  | +2.5 |
| 40%     | 20.0 | 21.1  | +1.2 | 75.2 | 70.3  | -4.9 |
| 20%     | 15.8 | 18.4  | +2.6 | 66.3 | 67.1  | +0.8 |

Table 5: Here we show the results of fine-tuning both GPT-Neo-2.7B (Neo) and our own multi-task model (Multi) on 100%, 40%, and 20% of the held-out data from LILA-OOD. The Multi almost always outperforms Neo (the Δ column shows the margin).

These findings establish Neo-Multi as a strong starting point for further fine-tuning on new tasks. For this reason, we release our multi-task model for public use under the name BHÅSKARA, with the hope that it will be useful for future research into math reasoning models.

5.2 Results: Category-wise Analysis

In this section we discuss the trends among the tasks within each category. For brevity, we primarily consider the GPT-Neo multi-task model in the program-synthesis setting.

Math ability. Among the tasks in the math category, Neo-Multi excels in basic math, linear algebra, and in-domain statistics. On these tasks, it performs equal or better to Codex. On the other hand, Neo-Multi struggles in advanced math and geometry, with mediocre performance in multiplication-division, number theory, and calculus. Codex shows analogous trends, except for performing very well on calculus (0.930).

Language complexity. Models generally show lower performance on program synthesis as language complexity increases. Fine-tuned GPT-Neo gets mean F1 over 0.5 only for datasets with the least linguistic complexity where it achieves an F1 of 0.7.

Question format. Among the format tasks in the dataset, Neo-Multi does exceptionally well on multiple-choice and natural-language inference, getting performance close to 0.9 on the latter, and outperforming Codex on both. On the other hand, the model performs close to 0.25 for reading comprehension and fill-in-the-blank, though with 0.5 F1 on out-of-domain fill-in-the-blank.

Background knowledge. Neo-Multi performs above 0.5 F1 only for problems requiring common-
Figure 3: Average F1 scores of GPT-3 and Codex with different numbers of few-shot examples in LILA.

Table 6: The IID scores for GPT-3 models with and without instruction prompting (Inst). Instruction helps slightly in zero-shot setting, but not in few-shot setting.

| Dimension | Zero-shot | Few-shot (3) |
|-----------|-----------|--------------|
|           | w/o Inst  | w/ Inst      | w/o Inst  | w/ Inst      |
| Math ability | 0.120 | 0.123 | 0.311 | 0.306 |
| Language  | 0.124 | 0.131 | 0.352 | 0.350 |
| Format  | 0.241 | 0.257 | 0.555 | 0.540 |
| Knowledge | 0.108 | 0.112 | 0.367 | 0.363 |
| Average  | 0.148 | 0.156 | 0.396 | 0.390 |

5.3 Results: Few-shot Prompting

Finally, we study the few-shot performance of much larger models (≈175B), to better understand the performance of the smaller trained models (≈2.7B) and to provide a benchmark for evaluating other large language models. Overall, we find that few-shot prompted models generally outperform their much smaller but fine-tuned counterparts.

Instructions and more examples improve performance. We find that the number of few-shot examples greatly impacts prompt models’ performance. Figure 3 shows that GPT-3 answer prediction beats Codex program synthesis in zero- to one-shot settings, but Codex overtakes with more examples. Table 6 shows that prompting with instructions improves performance only in the zero-shot setting, meaning that in the limited contexts of the prompt models, examples are more important than instructions for mathematical reasoning. This is consistent with the findings of Puri et al. (2022) on instruction-example equivalence.

Few-shot GPT-3 answer prediction underperforms Neo-Multi. While prompt-based models outperform our fine-tuned models in general when comparing within direct-answering and program-synthesis, when comparing GPT-Neo program-synthesis to GPT-3 direct answering we find that the much smaller fine-tuned GPT-Neo consistently outperforms GPT-3.

Few-shot Codex performance is relatively strong. Relative to the 2.7B trained models, Codex demonstrates strong few-shot IID and OOD performance. Some notable exceptions to this pattern are the statistics, linear algebra, multiple-choice question answering, and NLI tasks. Generally, OOD few-shot performs much better than OOD for the fine-tuned models.

Few-shot Codex fails on some tasks. Despite strong performance relative to fine-tuned GPT-Neo, Codex obtains less that 0.5 F1 on several tasks, with especially poor performance on geometry, number theory, advanced math, complex language, computer science problems, science formulas, and real world knowledge.

6 Conclusion

In this work, we introduce LILA, a unified mathematical reasoning benchmark for a holistic evaluation of AI agents. LILA consists of 23 tasks across 4 dimensions (i) mathematical abilities, (ii) language format, (iii) language complexity, (iv) external knowledge. It builds on 20 existing mathematical reasoning datasets to collect instructions and Python programs. Further, it also supports measuring out-of-distribution performance and robustness to language perturbations via LILA-OOD and LILA-ROBUST respectively. We also introduce BHĀSKARA, a 2.7B-parameter fine-tuned multi-task model. We find that multi-tasking improves over single-task performance by 21.83% F1 score on average, and that our model is a strong starting point for further fine-tuning on new math reasoning tasks. The best performing model we evaluate achieves only 60.40% F1 indicating the potential for improvement on the proposed benchmark.

6.1 Limitations

One drawback of our unified format is the difficulty of evaluating models. In our work we use F1 for lack of a better alternative. F1 likely overestimates performance, e.g., given the gold answer
“2 apples”, the predicted answers “2” and “apples” receive the same score, though the former is better.

LILA contains 23 tasks which are created from 20 datasets and 44 sub-datasets. There is scope to add more mathematical reasoning datasets (e.g., theorem proving.) The flexible unified format of LILA allows for future extensions. Additionally, our categorization provides a way to identify areas for extension. For instance, we only have 1 dataset for linear algebra, which happens to not use natural language, and takes the form of generative QA. Our benchmark will benefit from future linear algebra additions, perhaps with word problems formatted as fill-in-the-blank questions.

References

Gilles Adda, Benoît Sagot, Karén Fort, and Joseph Mariani. 2011. Crowdsourcing for language resource development: Critical analysis of amazon mechanical turk overpowering use. In 5th Language and Technology Conference.

Armen Aghajanyan, Anchant Gupta, Akshat Shrivas-tava, Xiliun Chen, Luke Zettlemoyer, and Sonal Gupta. 2021. Muppet: Massive multi-task representations with pre-finetuning. arXiv preprint arXiv:2101.11038.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel- Kedziorski, Yejin Choi, and Hannan Toudaheishiri. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. arXiv preprint arXiv:1905.13319.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. arXiv preprint arXiv:2108.07732.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. Gpt-neo: Large scale autore-gressive language modeling with mesh-tensorflow.

Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew E Peters, Ashish Sabharwal, and Yejin Choi. 2020. Adversarial filters of dataset biases. arXiv preprint arXiv:2002.04108.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In Advances in Neural Information Processing Systems, volume 33, pages 1877–1901. Curran Associates, Inc.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168.

Henry T Colebrooke. 1817. Arithmetick and mensuration of brahmegupta and bhaskara.

Dheeru Dua, Ananth Gottumukkala, Alon Talmor, Sameer Singh, and Matt Gardner. 2019a. Orb: An open reading benchmark for comprehensive evaluation of machine reading comprehension. arXiv preprint arXiv:1912.12598.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019b. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 2368–2378.

Karén Fort, Gilles Adda, and Kevin Bretonnel Cohen. 2011. Amazon mechanical turk: Gold mine or coal mine? Computational Linguistics, pages 413–420.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Pang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800gb dataset of diverse text for language modeling. arXiv preprint arXiv:2101.00027.

Jesse Michael Han, Jason M. Rute, Yuhuai Wu, Edward W. Ayers, and Stanislas Polu. 2021. Proof arti-fact co-training for theorem proving with language models. ArXiv, abs/2102.06203.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, et al. 2021a. Measuring coding challenge competence with apps. arXiv preprint arXiv:2105.09938.

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.

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. 2020. Pretrained transformers improve out-of-distribution
robustness. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2744–2751.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Danqing Huang, Shuming Shi, Chien-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 887–896.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. Unifiedqa: Crossing format boundaries with a single qa system. *arXiv preprint arXiv:2005.00700*.

Aditya Kolachana, K Mahesh, and K Ramasubramanian. 2019. Use of calculus in hindu mathematics. In *Studies in Indian Mathematics and Astronomy*, pages 345–355. Springer.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. Mawps: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281.

Wenda Li, Lei Yu, Yuhuai Wu, and Lawrence C. Paulson. 2021. Isarstep: a benchmark for high-level mathematical reasoning. In *International Conference on Learning Representations*.

Bill Yuchen Lin, Seyeon Lee, Rahul Khanna, and Xiang Ren. 2020. Birds have four legs?! numbersense: Probing numerical commonsense knowledge of pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6862–6868.

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*.

Nicholas Lourie, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Unicorn on rainbow: A universal commonsense reasoning model on a new multitask benchmark. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13480–13488.

Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. 2021a. Inter-gps: Interpretable geometry problem solving with formal language and symbolic reasoning. In *The 59th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurush, Peter Clark, and Ashwin Kalyan. 2022. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. *arXiv preprint arXiv:2209.14610*.

Pan Lu, Liang Qiu, Jiaqi Chen, Tony Xia, Yizhou Zhao, Wei Zhang, Zhou Yu, Xiaodan Liang, and Song-Chun Zhu. 2021b. Iconqa: A new benchmark for abstract diagram understanding and visual language reasoning. In *The 35th Conference on Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS 2021)*.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020a. A diverse corpus for evaluating and developing English math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984, Online. Association for Computational Linguistics.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020b. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, Yejin Choi, and Hannaneh Hajishirzi. 2022a. Reframing instructional prompts to GPTk’s language. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 589–612, Dublin, Ireland. Association for Computational Linguistics.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022b. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487.

Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Sachdeva, Peter Clark, Chitta Baral, and Ashwin Kalyan. 2022c. Nunglue: A suite of fundamental yet challenging mathematical reasoning tasks. In *Proceedings of the 60th Annual Meeting of the
Association for Computational Linguistics (Volume 1: Long Papers), pages 3505–3523.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2080–2094, Online. Association for Computational Linguistics.

Ravsehaj Singh Puri, Swaroop Mishra, Mihir Parmar, and Chitta Baral. 2022. How many data samples is an additional instruction worth? arXiv preprint arXiv:2203.09161.

Abhilasha Ravichander, Aakanksha Naik, Carolyn Rose, and Eduard Hovy. 2019. Equate: A benchmark evaluation framework for quantitative reasoning in natural language inference. arXiv preprint arXiv:1901.03735.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 4902–4912.

Anna Rogers, Matt Gardner, and Isabelle Augenstein. 2021. Qa dataset explosion: A taxonomy of nlp resources for question answering and reading comprehension. arXiv preprint arXiv:2107.12708.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1743–1752.

Subhro Roy and Dan Roth. 2016. Solving general arithmetic word problems. arXiv preprint arXiv:1608.01413.

Subhro Roy and Dan Roth. 2017. Unit dependency graph and its application to arithmetic word problem solving. In Thirty-First AAAI Conference on Artificial Intelligence.

Subhro Roy and Dan Roth. 2018. Mapping to declarative knowledge for word problem solving. Transactions of the Association for Computational Linguistics, 6:159–172.

Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language. Transactions of the Association for Computational Linguistics, 3:1–13.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. arXiv preprint arXiv:2110.08207.

Benoy Kumar Sarkar. 1918. Hindu Achievements in Exact Science: A Study in the History of Scientific Development. Longmans, Green and Company.

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

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. arXiv preprint arXiv:2206.04615.

Oyvind Tafjord, Peter Clark, Matt Gardner, Wen-tau Yih, and Ashish Sabharwal. 2019. Quarel: A dataset and models for answering questions about qualitative relationships. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 7063–7071.

Shyam Upadhyay and Ming-Wei Chang. 2015. Draw: A challenging and diverse algebra word problem set. Technical report, Citeseer.

Shyam Upadhyay, Ming-Wei Chang, Kai-Wei Chang, and Wen-tau Yih. 2016. Learning from explicit and implicit supervision jointly for algebra word problems. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 297–306.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In Advances in Neural Information Processing Systems, pages 3261–3275.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. arXiv preprint arXiv:1804.07461.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022. Benchmarking generalization via in-context instructions on 1,600+ language tasks. arXiv preprint arXiv:2204.07705.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. arXiv preprint arXiv:2109.01652.

Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hannaneh Hajishirzi, Yejin Choi, and Kyunghyun Cho. 2021. Naturalproofs: Mathematical theorem proving in natural language. In Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1).
Sean Welleck, Peter West, Jize Cao, and Yejin Choi. 2022. Symbolic brittleness in sequence models: on systematic generalization in symbolic mathematics. In AAAI.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. arXiv preprint arXiv:1502.05698.

Yuhuai Wu, Albert Jiang, Jimmy Ba, and Roger Baker Grosse. 2021. {INT}: An inequality benchmark for evaluating generalization in theorem proving. In International Conference on Learning Representations.

Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. 2018. Learning to mine aligned code and natural language pairs from stack overflow. In International Conference on Mining Software Repositories, MSR, pages 476–486. ACM.

Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. 2018. Taskonomy: Disentangling task transfer learning. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3712–3722.

Xikun Zhang, Deepak Ramachandran, Ian Tenney, Yanai Elazar, and Dan Roth. Do language embeddings capture scales?

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2021. Minif2f: a cross-system benchmark for formal olympiad-level mathematics. arXiv preprint arXiv:2109.00110.

Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. 2019. “going on a vacation” takes longer than “going for a walk”: A study of temporal commonsense understanding. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 3363–3369.
A Qualitative Examples

Figures 7 and 8 give examples of input-output behavior of GPT-Neo fine-tuned on all the tasks. Figure 9 gives an example of a non-compiling output program.

B Dataset Collection

Tables 15-18 give examples and datasets from each task for each category.

| Category    | Examples | Datasets |
|-------------|----------|----------|
| Math        | Table 11 | Table 15 |
| Language    | Table 12 | Table 16 |
| Format      | Table 13 | Table 17 |
| Knowledge   | Table 14 | Table 18 |

Table 10: Examples and datasets meta-table.

B.1 Expert annotation

In the worker qualification process, we ask each worker to annotate 30 questions. We manually verify each annotation and qualify those whose Python annotations are satisfactory. We also provide feedback such as "write simpler programs, use representative variable names instead of just letters, add comments wherever possible" to annotators after the worker qualification process. We instruct annotators to use a minimal set of Python libraries, and we ask them to record the Python libraries they use in a common document. We find that the annotators could get the task done just by using the sympy and the datetime libraries. We also ask annotators to report any bugs in answer annotation, which they report for a small number of questions; we subsequently fix those.

We give 10 sample question annotations to annotators as illustrative examples which vary in structure, length, format, underlying reasoning skill, etc. We pay 20 dollars per hour up to 20 hours per week as compensation for the data annotation work.

LILA-ROBUST To create the LILA-ROBUST dataset, we first define a set of 9 templates, consisting of 3 variation styles defined in SVAMP (Patel et al., 2021) as well as 6 novel templates of our own. We refer to the SVAMP templates as SVAMP-COO, SVAMP-COP, and SVAMP-IU, which correspond to changing the order of objects, changing the order of phrases, and adding irrelevant, unhelpful information to the problem statement, respectively. Our novel templates are named ROBUST-IR, ROBUST-AP, ROBUST-ADJ, ROBUST-Q, ROBUST-RQ, and ROBUST-RM. ROBUST-IR refers to adding information that is unhelpful for solving the question but may be related to the context of the problem. ROBUST-AP refers to increasing problem verbosity by turning active speech to passive speech. ROBUST-ADJ refers to increasing problem verbosity by adding adjectives or adverbs. ROBUST-Q indicates turning a problem statement into a question, in the style of a conversation with a student. ROBUST-RQ indicates removing question words in a problem and turning it into a statement; it is roughly the inverse of ROBUST-Q. Finally, ROBUST-RM refers to the removal of mathematics terms that are implicitly defined. Examples of each template are found in Table 19.

For our crowdsourcing pipeline, we provide each Amazon Mechanical Turk worker with 10 questions split from 20 questions sampled from each dataset. We run a separate job for each of our 9 templates. In particular, each HIT contains the 10 split questions from the original datasets, alongside the problem solution. Workers are asked to submit an augmentation for each question according to the style of the template assigned to each job. Thus, we run 9 separate jobs to obtain augmentations for all templates across all datasets. To familiarize workers with the intended style of each template, we provide 3 demonstrative augmentations within the instructions of each HIT, as summarized in Table 19. We restrict our crowdsourcing pipeline to workers that had above a 98% acceptance rate with over 1000 completed HITs. We provide workers with an upper bound of 1 hour to complete each HIT but specify in the instructions that each HIT should feasible be completed in 10 minutes. Based on minimum wage policies and under the assumption that workers follow the 10-minute completion guideline, we accordingly compensate $3 per HIT. Finally, to ensure dataset quality of generations via the Amazon Mechanical Turk (Fort et al., 2011; Adda et al., 2011), we manually assess the worker augmentations produced for each template.

C Dataset Statistics

Figure 5 gives relatives sizes of tasks within each category. Figure 6 illustrates the unigram frequencies in LILA, where larger words indicate higher frequency. Table 20 gives comprehensive statistics on each task. Table 22 cites each component
Task Basic Math

Problem Before December, customers buy 1346 ear muffs from the mall. During December, they buy 6444, and there are none. In all, how many ear muffs do the customers buy?

Answer 7790.0

Predicted Answer 1346.0 X

Generated Program

```python
from sympy import Poly
p = Poly(9*x**2 + 3*x + 7 + 3*x**2 + 7*x**5 + 2)
answer = p.as_expr()
print(answer)
```

Executed Answer 7790.0 ✓

Task Muldiv

Problem Tickets to the school play cost 6 for students and 8 for adults. If 20 students and 12 adults bought tickets, how many dollars’ worth of tickets were sold?

Answer 216

Predicted Answer 48 X

Generated Program

```python
from sympy import Poly
p = Poly(9*x**2 + 3*x + 7 + 3*x**2 + 7*x**5 + 2)
answer = p.as_expr()
print(answer)
```

Executed Answer 216 ✓

Table 7: Examples using Neo-Multi.
**Task Statistics**

**Problem** Determine the KL Divergence between the empirical distributions
0.149, 0.039, 0.099, 0.09, 0.203, 0.077, 0.039, 0.19 and
0.158, 0.256, 0.042, 0.008, 0.064, 0.051, 0.05, 0.072.

**Answer** 0.5122...

**Predicted Answer** 0.36 ✗

**Generated Program**

```python
from sympy import Poly

p = Poly(9*x**2 + 3*x + 7 + 3*x**2 + 7*x**5 + 2)

answer = p.as_expr()

print(answer)
```

**Executed Answer** 0.5122... ✓

---

**Task Linear Algebra**

**Problem** Find the determinant of the matrix
\[
\begin{pmatrix}
0 & -2 & -3 \\
0 & 5 & 0 \\
1 & 3 & 2
\end{pmatrix}
\]

**Answer** 15

**Predicted Answer** -8 ✗

**Generated Program**

```python
from sympy import Poly

p = Poly(9*x**2 + 3*x + 7 + 3*x**2 + 7*x**5 + 2)

answer = p.as_expr()

print(answer)
```

**Executed Answer** 15.0 ✓

---

Table 8: Examples using Neo-Multi.
Model Neo-Multi
Task Muldiv
Problem Jenny collects cans and bottles to take down to the recycling center. Each bottle weighs 6 ounces and each can weighs 2 ounces. Jenny can carry a total of 100 ounces. She collects 20 cans and as many bottles as she can carry. If she gets paid 10 cents per bottle and 3 cents per can, how much money does she make (in cents)?

Generated Program

```python
from sympy import Poly

p = Poly(9*x**2 + 3*x + 7 + 3*x**2 + 7*x**5 + 2)

answer = p.as_expr()

print(answer)

Error: NameError (g is not defined)
```

Table 9: NameErrors in Neo-Multi and Codex.

---

Model Codex
Task Advanced Math
Problem Simplify the expression \((9x^2 + 3x + 7) + (3x^2 + 7x^5 + 2)\). Express your answer as a polynomial with terms arranged in decreasing order of degree.

Generated Program

```python
from sympy import Poly

p = Poly(9*x**2 + 3*x + 7 + 3*x**2 + 7*x**5 + 2)

answer = p.as_expr()

print(answer)

Error: NameError (x is not defined)
```
Table 11: Example of each task in the math ability category of the LILA benchmark.

**Question:** A gardener is going to plant 2 red rosebushes and 2 white rosebushes. If the gardener is to select each of the bushes at random, one at a time, and plant them in a row, what is the probability that the 2 rosebushes in the middle of the row will be the red rosebushes?

**Options:** (A: 1/12, B: 1/6, C: 1/5, D: 1/3, E: 1/2)

**Answer:** B

**Explanation:** We are asked to find the probability of one particular pattern: wrrw. Total # of ways a gardener can plant these four bushes is the # of permutations of 4 letters wwww, out of which 2 w’s and 2 r’s are identical, so 4! / 2! 2! = 6; so p = 1 / 6. Answer: B.

**Program:**

```python
import scipy

n0 = 2.0
n1 = 2.0
n2 = 2.0
t0 = n0 + n0
t1 = scipy.special.comb(t0, n0)
answer = 1.0 / t1
```

Figure 4: An example of instruction annotation.

dataset of LILA.

### D Additional Results

Table 21 gives the unaggregated performance of each model on each dataset in LILA (some datasets are split across tasks).
| Task | Question category | Example |
|------|------------------|---------|
| TASK 10 | No language | Compute the median of $\sqrt{2} \cdot 3e - 6, -6, -6, \frac{11}{12}, 6$. Answer: 3 |
| TASK 11 | Simple language | **Question:** Joan had 9 blue balloons, but Sally popped 5 of them. Jessica has 2 blue balloons. They have __ blue balloons now. **Answer:** 6 |
| TASK 12 | Complex language: involving co-reference resolution etc., multi-sentence language, adversarial language: containing tricky words etc., often created adversarially | **Question:** Passage: According to the 2011 National Household Survey, 89.3% of Markhams residents are Canadian citizens, and about 14.5% of residents are recent immigrants (from 2001 to 2011). The racial make up of Markham is; East Asian (39.7%), White Canadian (27.5%), South Asian Canadian (19.1%), Southeast Asian (3.9%), Black Canadians (3.2%), West Asian & Arab Canadians (3.2%), Latin American Canadian (0.5%), Aboriginal peoples in Canada (0.2%), and 1.9% of the population is multiracial while the rest of the population (0.7%) is of another group. Markham has the highest visible minority population of any major Canadian city (over 100,000 residents) at 72.3%, and is one of eight major cities with no majority racial group. Question: How many percent of people were not white? **Answer:** 72.5 |

Table 12: Example of each task in the *language complexity* category of the LILA benchmark.

| Task | Question category | Example |
|------|------------------|---------|
| TASK 13 | Fill in the blank | **Question:** Delphinium has _ florets or they are full of holes. **Answer:** no |
| TASK 14 | Generative question answering | **Question:** Calculate the remainder when 160 is divided by 125. **Answer:** 35 |
| TASK 15 | Multiple choice question answering (MCQ) | **Question:** The fish glided with a speed of 8 m/s through the water and 5 m/s through the jello because the ___ is smoother? "Option 1": jello, "Option 2": water. **Answer:** Option 2 |
| TASK 16 | Natural language inference (NLI) | **Question:** "statement 1": Alyssa picked 42.0 pears from the pear tree and Nancy sold 17.0 of the pears, "statement 2":25.0 pears were left , "options: " Entailment or contradiction? **Answer:** Entailment |
| TASK 17 | Reading comprehension (RC) | **Question:** Passage: A late game rally by Washington led them to the Eagles’ 26 yard line. A shot to the end zone by Robert Griffin III would be intercepted by Brandon Boykin, clinching an Eagles win. The Eagles would move to 6-5. This is the Eagles first win at Lincoln Financial Field since Week 4 of the 2012 season, because prior to this game, the Eagles had never won a game in their home stadium in 414 days since that same week, snapping a 10-game losing streak at home with this win. Question: How many more wins than losses did the Eagles have after this game? **Answer:** 1 |

Table 13: Example of each task in the *question format* category of the LILA benchmark.

| Task | Question category | Example |
|------|------------------|---------|
| TASK 18 | No external knowledge: only mathematical commonsense knowledge required | **Question:** If there are 7 bottle caps in a box and Linda puts 7 more bottle caps inside, how many bottle caps are in the box? **Answer:** 14 |
| TASK 19 | Commonsense: temporal commonsense knowledge (e.g., people usually play basketball for a few hours and not days), numerical commonsense knowledge (e.g. birds has 2 legs) | **Question:** Outside temple, there is a shop which charges 12 dollars for each object. Please note that one shoe is counted as an object. Same is true for socks and mobiles. Paisley went to temple with both parents. All of them kept their shoes, socks and mobiles in the shop. How much they have to pay? **Answer:** 180 |
| TASK 20 | Math formulas: algebra, geometry, probability etc. | **Question:** Simplify $-3*(\sqrt{1700}) - (\sqrt{1700} + (3 + \sqrt{1700})*-6) + -3$. **Answer:** $-180*\sqrt{17} - 57$ |
| TASK 21 | Science formulas: physics, chemistry etc. | **Question:** Find the number of moles of H2O formed on combining 2 moles of NaOH and 2 moles of HCl. **Answer:** 2 |
| TASK 22 | Computer science knowledge: data structure algorithms like merge sort etc. | **Question:** Calculate the median of $4\sqrt{2} \cdot 3e - 6, -6, -6, \frac{11}{12}, 6$. **Answer:** 3 |
| TASK 23 | Real-world knowledge: COVID modelling, climate modelling etc. | **Question:** Our physics club has 20 members, among which we have 3 officers: President, Vice President, and Treasurer. However, one member, Alex, hates another member, Bob. How many ways can we fill the offices if Alex refuses to serve as an officer if Bob is also an officer? (No person is allowed to hold more than one office.) **Answer:** 6732 |

Table 14: Example of each task in the *background knowledge* category of the LILA benchmark.
| Task | Math category | IID | OOD |
|------|---------------|-----|-----|
| TASK 1 | Basic math | addsub.json, Numerense_structured.json, MCTaco_stationarity_structured.json, MCTaco_frequency_structured.json, MCTaco_event_typical_time_structured.json, MCTaco_event_ordering_structured.json, NumGLUE_Task3.json | MCTaco_event_duration_structured.json, NumGLUE_Task1.json |
| TASK 2 | Muldiv | singleop.json, multiarith.json, asdiv.json, GSM8k_structured.json, NumGLUE_Task1.json, NumGLUE_Task2.json, deepmind_mathematics_muldiv.json | svamp_structured.json, NumGLUE_Task4.json |
| TASK 8 | Number theory | mathqa_physics.json, APPS_structured.json, mathqa_gain.json, mathqa_numbertheory.json, mathqa_general.json, conala_structured.json, NumGLUE_Task5.json, deepmind_mathematics_numbertheory.json | mbpp_structured.json, mathqa_other.json |
| TASK 4 | Algebra | singleq.json, simuleq.json, amps_algebra.json, NumGLUE_Task8.json, deepmind_mathematics_algebra.json | draw_structured.json, dolphin_structured.json |
| TASK 5 | Geometry | amps_geometry.json | mathqa_geometry.json |
| TASK 6 | Statistics | amps_counting_and_stats.json | mathqa_probability.json |
| TASK 7 | Calculus | amps_calculus.json, deepmind_mathematics_calculus.json | |
| TASK 8 | Linear algebra | amps_linear_algebra.json | |
| TASK 9 | Advanced math | MATH_crowdsourced.json | |

Table 15: Raw datasets used to create different tasks in LILA across different math categories.
| ID | Language category | IID | OOD |
|----|------------------|-----|-----|
|    |                  |     |     |
| 10 | No language      | amps_number_theory.json | amps_algebra.json |
|    |                  | amps_counting_and_stats.json | deepmind_mathematics_calculus.json |
|    |                  | amps_calculus.json | |
|    |                  | amps_linear_algebra.json | |
|    |                  | deepmind_mathematics_multdiv.json | |
|    |                  | deepmind_mathematics_numbertheory.json | |
|    |                  | deepmind_mathematics_algebra.json | |
|    |                  | deepmind_mathematics_basicmath.json | |
| 11 | Simple language  | addsub.json | MCTaco_frequency_structured.json |
|    |                  | Numbersense_structured.json | NumGLUE_Task1.json |
|    |                  | MCTaco_stationarity_structured.json | mathqa_general.json |
|    |                  | MCTaco_event_typical_time_structured.json | NumGLUE_Task4.json |
|    |                  | MCTaco_event_ordering_structured.json | |
|    |                  | MCTaco_event_duration_structured.json | |
|    |                  | singleop.json | |
|    |                  | multiarith.json | |
|    |                  | asdiv.json | |
| 12 | Complex language | GSM8k_structured.json | MCTaco_frequency_structured.json |
|    |                  | APPS_structured.json | NumGLUE_Task1.json |
|    |                  | mathqa_gain.json | mathqa_general.json |
|    |                  | mathqa_other.json | NumGLUE_Task4.json |
|    |                  | singleq.json | |
|    |                  | simulq.json | |
|    |                  | NumGLUE_Task8.json | |
|    |                  | draw_structured.json | |
|    |                  | dolphin_structured.json | |
|    |                  | mathqa_probability.json | |
|    |                  | mathqa_physics.json | mbpp_structured.json |
|    |                  | APPS_structured.json | mathqa_other.json |
|    |                  | mathqa_gain.json | |
|    |                  | amps_number_theory.json | |
| 13 | Complex language | mathqa_general.json | |
|    |                  | conala_structured.json | |
|    |                  | NumGLUE_Task5.json | |
|    |                  | deepmind_mathematics_numbertheory.json | |

Table 16: Raw datasets used to create different tasks in LiLa across different language categories.
| ID     | Format category | ID                      | OOD                                      |
|--------|-----------------|-------------------------|------------------------------------------|
| TASK 13| Fill in the blank | NumGLUE_Task4.json | Numersense_structured.json              |
|        |                 |                         | svamp_structured.json                   |
|        |                 |                         | mathqa_geometry.json                    |
|        |                 |                         | amps_calculus.json                      |
|        |                 |                         | singleeq.json                           |
|        |                 |                         | NumGLUE_Task2.json                      |
|        |                 |                         | mbpp_structured.json                    |
|        |                 |                         | deepmind_mathematics_numbertheory.json  |
| TASK 14| Generative QA   |                         |                                         |
|        |                 |                         |                                         |
|        |                 |                         |                                         |
| TASK 15| MCQ             | NumGLUE_Task3.json      | MCTaco_event_typical_time_structured.json |
|        |                 | MCTaco_stationarity_structured.json |                                         |
|        |                 | MCTaco_event_ordering_structured.json |                                         |
|        |                 | MCTaco_event_duration_structured.json |                                         |
| TASK 16| NLI             | NumGLUE_Task5.json      |                                         |
| TASK 17| RC              | mathqa_physics.json    | mbpp_structured.json                    |

Table 17: Raw datasets used to create different tasks in LILA across different format categories.
Table 18: Raw datasets used to create different tasks in LILA across different knowledge categories.

| ID | Knowledge category | ID | OOD |
|----|--------------------|----|-----|
| TASK 18 | No external knowledge | addsub.json | NumGLUE_Task4.json |
| | | singleop.json | GSM8k_structured.json |
| | | multiarith.json | svamp_structured.json |
| | | asdiv.json | NumGLUE_Task7.json |
| TASK 19 | Commonsense | simuleq.json | NumGLUE_Task8.json |
| | | draw_structured.json | draw_structured.json |
| | | dolphin_structured.json | dolphin_structured.json |
| | | NumGLUE_Task5.json | NumGLUE_Task5.json |
| | | deepmind_mathematics_muldiv.json | deepmind_mathematics_muldiv.json |
| TASK 20 | Math formulas | Numbersense_structured.json | NumGLUE_Task1.json |
| | | MCTaco_frequency_structured.json | MCTaco_event_ordering_structured.json |
| | | MCTaco_stationarity_structured.json | MCTaco_stationarity_structured.json |
| | | MCTaco_event_duration_structured.json | MCTaco_event_duration_structured.json |
| | | MCTaco_event_typical_time_structured.json | MCTaco_event_typical_time_structured.json |
| TASK 21 | Science formulas | amps_number_theory.json | amps_counting_and_stats.json |
| | | amps_linear_algebra.json | mathqa_general.json |
| | | amps_algebra.json | amps_calculus.json |
| | | deepmind_mathematics_calculus.json | deepmind_mathematics_calculus.json |
| | | mathqa_probability.json | mathqa_probability.json |
| | | singleeq.json | singleeq.json |
| | | mathqa_gain.json | mathqa_gain.json |
| | | mathqa_other.json | mathqa_other.json |
| | | deepmind_mathematics_algebra.json | deepmind_mathematics_algebra.json |
| | | deepmind_mathematics_basicmath.jsom | deepmind_mathematics_basicmath.jsom |
| | | deepmind_mathematics_calculus.jsom | deepmind_mathematics_calculus.jsom |
| | | deepmind_mathematics_numbertheory.jsom | deepmind_mathematics_numbertheory.jsom |
| TASK 22 | Computer science knowledge | APPS_structured.json | mathqa_geometry.jsom |
| | | mathqa_physics.jsom | mathqa_physics.jsom |
| TASK 23 | Real-world knowledge | MATH_crowdsourced.jsom | mbpp_structured.jsom |

(a) Math ability categories.
(b) Language categories.
(c) Format categories.
(d) Knowledge categories.

Figure 5: Task diversity in LILA across math, language, format, and knowledge categories.
| Template Name | Variation | Example |
|--------------|-----------|---------|
| SVAMP-COO    | Change the order of objects | **Question:** Allen bought 20 stamps at the post office in 37 cents and 20 cents denominations. If the total cost of the stamps was $7.06, how many 37 cents stamps did Allen buy? **Variation:** Allen bought 20 stamps at the post office in 20 cents and 37 cents denominations. If the total cost of the stamps was $7.06, how many 37 cents stamps did Allen buy? |
| SVAMP-COP    | Change the order of phrases | **Question:** One pipe can fill a tank in 5 hours and another pipe can fill the same tank in 4 hours. A drainpipe can empty the full content of the tank in 20 hours. With all the three pipes open, how long will it take to fill the tank? **Variation:** A drainpipe can empty the full content of a tank in 20 hours. One pipe can fill the tank in 4 hours and another pipe can fill the same tank in 5 hours. With all the three pipes open, how long will it take to fill the tank with all the three pipes open? |
| SVAMP-IU     | Add irrelevant, unhelpful information | **Question:** the area of an isosceles trapezoid with sides of length 5 and bases of length 7 and 13 is? **Variation:** monkeys and apes are both primates, which means they’re both part of the human family tree. the area of an isosceles trapezoid with sides of length 5 and bases of length 7 and 13 is? |
| ROBUST-IR    | Add unhelpful, but contextually related information | **Question:** Tom is 15 years younger than alice. Ten years ago, Alice was 4 times as old as Tom was then. How old is each now? **Variation:** Tom is 15 years younger than alice. Ten years ago, Alice was 4 times as old as Tom was then. Alice really likes pineapple pizza. How old is each now? |
| ROBUST-AP    | Turn active into passive speech to increase problem verbosity | **Question:** Hay’s Linens sells hand towels in sets of 17 and bath towels in sets of 6. If the store sold the same number of each this morning, what is the smallest number of each type of towel that the store must have sold? **Variation:** Hand towels are sold by Hay’s Linens in sets of 17 and bath towels are sold in sets of 6. If the same number of each were sold by the store this morning, what is the smallest number of each type of towel that the store must have sold? |
| ROBUST-ADJ   | Add adjectives and adverbs to increase problem verbosity | **Question:** Tea leaves exposed to oxygen for up to _ hours become black tea. **Variation:** Black tea leaves continuously exposed to oxygen for up to _ hours become a very rich black tea. |
| ROBUST-Q     | Turn a task statement into a question | **Question:** Product of -7 and -1469.125. **Variation:** What is the product of -7 and -1469.125? |
| ROBUST-RQ    | Turn a question into a task statement | **Question:** Problem: If the product of 5 and a number is increased by 4, the result is 19. What is the number? **Variation:** Increasing the product of 5 and a number by 4 results is 19. Find the number. |
| ROBUST-RM    | Remove explicitly mathematical terms that are implicitly defined | **Problem:** Find the arclength of the function \( f(x) = 2\sqrt{x} \) on the interval \( x = 2 \) to \( x = 8 \) \( \sqrt{\frac{4}{x}} \) on \([2, 8]\). **Variation:** Find the arclength of \( f(x) = 2\sqrt{x} \) on \([2, 8]\). |

Table 19: Example for each template provided to MTurk workers to produce LILA-ROBUST

5829
| ID | Category               | Questions | Unique questions | Question length | Programs | Unique programs | Program length |
|----|------------------------|-----------|------------------|-----------------|----------|-----------------|----------------|
| 1  | Basic math             | 31,052    | 31,032           | 43.1            | 31,052   | 7,066           | 13.3           |
| 2  | Multiv                 | 16,021    | 15,936           | 26.9            | 16,021   | 15,279          | 8.2            |
| 3  | Number theory          | 44,760    | 44,183           | 41.3            | 269,232  | 261,865         | 33.2           |
| 4  | Algebra                | 15,882    | 15,615           | 19.3            | 16,364   | 15,986          | 12.7           |
| 5  | Geometry               | 3,190     | 3,149            | 36.1            | 3,190    | 3,035           | 28.7           |
| 6  | Counting and statistics| 6,423     | 6,384            | 39.7            | 6,423    | 6,335           | 31.5           |
| 7  | Calculus               | 4,493     | 4,202            | 21.2            | 4,493    | 4,170           | 40.6           |
| 8  | Linear algebra         | 11,248    | 11,204           | 32.4            | 11,248   | 11,204          | 23.0           |
| 9  | Advanced math          | 746       | 746              | 21.2            | 746      | 745             | 27.3           |
| 10 | No language            | 41,191    | 40,551           | 21.2            | 42,466   | 41,794          | 40.6           |
| 11 | Simple language        | 66,505    | 66,172           | 26.9            | 290,184  | 258,839         | 8.2            |
| 12 | Complex language       | 26,119    | 25,728           | 36.1            | 26,119   | 25,052          | 28.7           |
| 13 | Fill in the blank      | 11,634    | 11,615           | 11.0            | 11,634   | 997             | 3.0            |
| 14 | Generative QA          | 102,493   | 101,239          | 14.7            | 327,447  | 314,652         | 16.0           |
| 15 | MCQ                    | 9,989     | 9,989            | 28.3            | 9,989    | 470             | 3.0            |
| 16 | NLI                    | 6,326     | 6,325            | 50.8            | 6,326    | 6,243           | 25.8           |
| 17 | RC                     | 3,642     | 3,552            | 182.5           | 3,642    | 3,592           | 10.4           |
| 18 | No external knowledge  | 28,115    | 27,964           | 50.8            | 28,115   | 27,117          | 25.8           |
| 19 | Commonsense            | 24,677    | 24,658           | 30.9            | 24,677   | 823             | 3.0            |
| 20 | Math formulas          | 57,841    | 56,947           | 19.1            | 59,116   | 57,019          | 25.5           |
| 21 | Science formulas        | 10,505    | 10,319           | 36.1            | 10,505   | 9,764           | 28.7           |
| 22 | Complex knowledge      | 12,200    | 12,086           | 14.5            | 235,879  | 230,486         | 24.2           |
| 23 | Real-world knowledge   | 746       | 746              | 21.2            | 746      | 745             | 27.3           |

Table 20: Main statistics of LiLa across the total of 23 tasks.

![Word cloud distribution of annotated programs in the LiLa dataset.](figure6.png)

Figure 6: The word cloud distribution of annotated programs in the LiLa dataset.
Table 21: Evaluation results of baselines across different single datasets. On most datasets, **Codex** performs best.

| ID  | Dataset                        | GPT-3 | Neo-A | Neo-P | Codex |
|-----|--------------------------------|-------|-------|-------|-------|
| 1   | addsub                         | 0.910 | 0.116 | 0.797 | 0.950 |
| 2   | amps_algebra                   | 0.116 | 0.100 | **0.902** | 0.655 |
| 3   | amps_calculus                  | 0.192 | 0.168 | **0.922** | 0.860 |
| 4   | amps_counting_and_stats        | 0.183 | 0.117 | **0.958** | 0.650 |
| 5   | amps_geometey                  | **0.283** | 0.263 | 0.074 | 0.000 |
| 6   | amps_linear_algebra            | 0.127 | 0.235 | **0.815** | 0.692 |
| 7   | amps_number_theory             | 0.273 | 0.026 | 0.875 | **1.000** |
| 8   | APPS_structured                | 0.167 | 0.154 | 0.134 | **0.459** |
| 9   | asdiv                          | 0.737 | 0.166 | 0.092 | 0.022 |
| 10  | conala_structured              | 0.356 | 0.329 | 0.329 | **0.391** |
| 11  | deepmind_mathematics_algebra   | 0.202 | 0.258 | 0.847 | **0.910** |
| 12  | deepmind_mathematics_basicmath | 0.270 | 0.125 | 0.614 | **1.000** |
| 13  | deepmind_mathematics_calculus  | 0.208 | 0.026 | 0.152 | **0.884** |
| 14  | deepmind_mathematics_muldiv    | 0.160 | 0.034 | 0.909 | **1.000** |
| 15  | deepmind_mathematics_numbertheory | 0.296 | 0.462 | 0.538 | **0.710** |
| 16  | dolphin_t2_final               | 0.170 | 0.027 | 0.006 | **0.812** |
| 17  | draw_structured                | 0.090 | 0.034 | 0.005 | **0.210** |
| 18  | GSM8k_structured               | 0.110 | 0.060 | 0.139 | **0.350** |
| 19  | MATH_crowdsourced              | 0.150 | 0.013 | 0.074 | **0.472** |
| 20  | mathqa_gain                    | 0.134 | 0.054 | **0.339** | 0.270 |
| 21  | mathqa_general                 | 0.110 | 0.073 | **0.193** | 0.120 |
| 22  | mathqa_geometry                | 0.120 | 0.002 | 0.000 | **0.250** |
| 23  | mathqa_other                   | 0.180 | 0.043 | 0.011 | **0.280** |
| 24  | mathqa_probability             | **0.210** | 0.003 | 0.000 | 0.200 |
| 25  | mbpp_structured                | 0.128 | 0.175 | 0.164 | **0.408** |
| 26  | MCTaco_event_duration_structured | **0.800** | 0.773 | 0.773 | 0.710 |
| 27  | MCTaco_event_ordering_structured | 0.860 | 0.831 | 0.831 | **0.890** |
| 28  | MCTaco_event_typical_time_structured | 0.870 | **0.881** | **0.881** | 0.870 |
| 29  | MCTaco_frequency_structured    | **0.890** | 0.862 | 0.862 | 0.790 |
| 30  | MCTaco_stationarity_structured | 0.710 | **0.758** | **0.758** | 0.670 |
| 31  | multiarith                     | 0.360 | 0.143 | 0.921 | **0.990** |
| 32  | Numersense_structured          | 0.620 | 0.495 | 0.495 | **0.660** |
| 33  | NumGLUE_Type_1                 | 0.535 | 0.108 | 0.083 | **0.740** |
| 34  | NumGLUE_Type_2                 | 0.512 | 0.285 | 0.646 | **0.735** |
| 35  | NumGLUE_Type_3                 | **0.835** | 0.004 | 0.001 | 0.815 |
| 36  | NumGLUE_Type_4                 | 0.710 | 0.076 | 0.208 | **0.790** |
| 37  | NumGLUE_Type_5                 | 0.460 | 0.200 | 0.305 | **0.615** |
| 38  | NumGLUE_Type_7                 | 0.500 | 0.516 | **0.854** | 0.710 |
| 39  | NumGLUE_Type_8                 | 0.420 | 0.082 | 0.257 | **0.610** |
| 40  | simuleq                        | 0.120 | 0.074 | 0.010 | **0.170** |
| 41  | singleloop                     | 0.940 | 0.347 | 0.611 | **1.000** |
| 42  | singleq                        | **0.830** | 0.143 | 0.474 | 0.670 |
| 43  | svamp_structured               | 0.620 | 0.085 | 0.060 | **0.790** |

Average F1 score: 0.400, 0.223, 0.440, **0.613**
| ID | Dataset   | References                                                                 |
|----|-----------|-----------------------------------------------------------------------------|
| 1  | addsub    | (Hosseini et al., 2014)                                                    |
| 2  | amps      | (Hendrycks et al., 2021b)                                                  |
| 3  | APPS      | (Hendrycks et al., 2021a)                                                  |
| 4  | asdiv     | (Miao et al., 2020b)                                                       |
| 5  | conala    | (Yin et al., 2018)                                                         |
| 6  | mathematics | (Saxton et al., 2019)                                                        |
| 7  | dolphin   | (Huang et al., 2016)                                                       |
| 8  | draw      | (Upadhyay and Chang, 2015)                                                  |
| 9  | GSM8k     | (Cobbe et al., 2021)                                                       |
| 10 | MATH      | (Hendrycks et al., 2021b)                                                  |
| 11 | mathqa    | (Amini et al., 2019)                                                       |
| 12 | mbpp      | (Austin et al., 2021)                                                      |
| 13 | MCTaco    | (Zhou et al., 2019)                                                        |
| 14 | multiarith| (Roy and Roth, 2015)                                                       |
| 15 | Numersense| (Lin et al., 2020)                                                         |
| 16 | NumGLUE   | (Mishra et al., 2022c; Dua et al., 2019b; Ravichander et al., 2019; Kushman et al., 2014; Tafjord et al., 2019; Roy and Roth, 2018, 2017; Koncel-Kedziorski et al., 2016, 2015) |
| 17 | simuleq   | (Kushman et al., 2014)                                                      |
| 18 | singleop  | (Roy et al., 2015)                                                         |
| 19 | singleq   | (Koncel-Kedziorski et al., 2015)                                            |
| 20 | svamp     | (Patel et al., 2021)                                                       |

Table 22: List of source datasets and corresponding references used in constructing LIL.A.