LĪLA: A Unified Benchmark for Mathematical Reasoning

Swaroop Mishra[2*†] **Matthew Finlayson**[1*‡] **Pan Lu**[3†] **Leonard Tang**[4], **Sean Welleck**[1], **Chitta Baral**[2], **Tanmay Rajpurohit**[5], **Oyvind Tafjord**[1], **Ashish Sabharwal**[1], **Peter Clark**[1], **Ashwin Kalyan**[1‡]

[1]The Allen Institute for AI, [2]Arizona State University, [3]UCLA,
[4]Harvard University, [5]Georgia Institute of Technology

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 LĪLA, 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 BHĀSKARA and its variants, a family of mathematical reasoning models fine-tuned on LĪLA. 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.[4]

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., [21] (algebra word problems), [34] (arithmetic reasoning), [46] (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 formats.

To this end, we introduce LĪLA, a unified mathematical reasoning benchmark that consists of 23 mathematical reasoning tasks. LĪLA 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, LĪLA extends all of these datasets to include a solution program as opposed to only an answer, and instruction annotations to enable instruction-based learning [45, 54, 33, 32].

*Equal first authors
†Work done while at the Allen Institute for AI
‡Corresponding authors: `matthewf@allenai.org`, `ashwink@allenai.org`

[4]Data: `https://github.com/allenai/Lila`. Model: `https://huggingface.co/allenai/bhaskara`

36th Conference on Neural Information Processing Systems (NeurIPS 2022) Workshop on MATH-AI.

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.

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, LĪLA 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 2 illustrates a sample from our LĪLA 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 [5], [38] and [56], 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 (LĪLA-OOD) containing datasets requiring the same underlying mathematical reasoning skill, but collected independently of the training datasets. further, we collect a robustness split LĪLA-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: LĪLA-TEST, LĪLA-OOD and LĪLA-ROBUST.

Contributions

1. We present LĪLA, a holistic benchmark for mathematical reasoning. LĪLA 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 BHĀSKARA, a family of language models 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 (i) multitasking improves considerably over task-specific learning both in IID and OOD evaluation (ii) program synthesis substantially outperforms answer prediction, (iii) few-shot prompting with codex performs best.

2 LĪLA

LĪLA contains 23 tasks across 4 dimensions, curated from 44 sub-datasets across 20 dataset sources. Here we discuss the construction and composition of the benchmark

2.1 Dataset Construction

Data Sources. We incorporate 20 existing datasets from the mathematical reasoning literature (Table 21 gives a detailed list), where inputs are natural language or templated text and outputs are numerical or expressions, e.g., we exclude theorem proving [55, 12], 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.
2. Category tags for each of the four dimensions (math ability, langauge complexity, format, and external knowledge; see §B.1).
3. The question, in English.
4. The answer to the question, as a string containing a number, expression, list, or other data format.
5. A set of Python strings that `print` the answer.
6. A task-level instruction in natural language.

We also retain meta-data from the original dataset. More details are in Appendix B.

3 Experiments

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

Data partition and evaluation. For the standard in-distribution (IID) setup, we randomly partition the data in *each* task into training (70%), development (10%) and test (20%) sets. Additionally, we also evaluate on LĪLA-OOD and LĪLA-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 models (a pre-trained causal language model [4]) on LĪLA. We choose GPT-Neo because it was pre-trained on both natural language and code [11], as opposed to a model trained 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 LĪLA. 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[5] [6, 7]. 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[6].

Evaluation. We evaluate 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[7] 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 programming techniques.

4 Results and Analysis

A summary results of all key results on our LĪLA benchmark are shown in Table 2. In this section, we wull discuss the performance of fine-tuned 2.7B GPT-Neo models (§4.1), performance of models along the 4 categories of tasks (§C.1) and finally, the few-shot performance of much larger (∼175B parameters) models (§C.2).

4.1 Results: Fine-tuned Models

Multitasking improves IID and OOD generalization. The multi-tasking model (Neo-Multi) substantially improves upon the single task models (Neo). Neo-Multi achieves better average

[5]`text-davinci-002, code-davinci-002`

[6]Henceforth we refer to the max example model unless otherwise specified.

[7]This soft version of exact match accuracy allows assigning partial credit for word overlap with gold answers.

	→ **Supervision/Size**	Few-shot, 175B		Few-shot, 175B		Fine-tuned, 2.7B		Fine-tuned, 2.7B		Fine-tuned, 2.7B		Fine-tuned, 2.7B	
↓ **Task**	**Category**	**GPT3**		**Codex**		**Neo-A**		**Neo-P**		**Neo-Multi-A**		**Neo-Multi-P**	
		IID	OOD	IID	OOD	IID	OOD	IID	OOD	IID	OOD	IID	OOD
1	Basic math	0.766	**0.818**	**0.791**	0.762	0.533	0.523	0.611	0.555	0.693	0.657	**0.790**	**0.787**
2	Muldiv	0.479	0.665	**0.691**	**0.790**	0.136	0.089	0.388	0.194	0.155	0.083	**0.448**	**0.395**
3	Number theory	0.240	0.154	**0.472**	**0.344**	0.108	0.095	0.328	0.107	0.129	0.190	**0.358**	**0.293**
4	Algebra	0.338	0.130	**0.603**	**0.511**	0.164	0.031	0.348	0.051	0.203	**0.054**	**0.473**	0.007
5	Geometry	**0.283**	0.120	0.000	**0.250**	0.288	0.025	0.077	0.021	**0.297**	0.105	0.079	**0.250**
6	Statistics	0.183	**0.210**	**0.650**	0.200	0.107	0.008	0.839	0.034	0.115	**0.179**	**0.947**	0.164
7	Calculus	0.231	0.208	**0.930**	**0.884**	0.138	0.119	0.486	0.334	0.102	0.167	**0.495**	**0.805**
8	Linear algebra	0.127	-	**0.692**	-	0.229	-	**0.809**	-	0.240	-	0.808	-
9	Advanced math	0.150	-	**0.472**	-	0.012	-	0.100	-	0.019	-	**0.160**	-
10	No language	0.213	0.162	**0.853**	**0.770**	0.143	0.083	0.698	0.330	0.140	0.138	**0.703**	**0.850**
11	Simple language	0.486	0.561	**0.568**	**0.610**	0.269	0.243	0.363	0.292	0.332	0.269	**0.433**	**0.384**
12	Complex language	0.356	0.413	**0.456**	**0.583**	0.147	0.113	0.216	0.106	0.215	0.259	**0.288**	**0.557**
13	Fill in the blank	0.710	0.620	**0.790**	**0.660**	0.086	0.193	**0.304**	0.193	0.059	**0.519**	0.262	**0.519**
14	Generative QA	0.305	0.385	**0.566**	**0.632**	0.142	0.135	0.376	0.199	0.178	0.160	**0.476**	**0.235**
15	MCQ	**0.801**	**0.870**	0.771	**0.870**	0.636	0.818	0.652	0.818	0.752	**0.888**	**0.817**	**0.888**
16	NLI	0.500	-	**0.710**	-	0.221	-	0.212	-	0.566	-	**0.893**	-
17	RC	0.460	-	**0.615**	-	0.135	-	**0.295**	-	0.132	-	0.264	-
18	No external k.	0.437	0.485	**0.638**	**0.660**	0.138	0.110	0.387	0.159	0.167	0.199	**0.400**	**0.465**
19	Commonsense	**0.788**	0.698	0.752	**0.815**	0.613	0.364	0.624	0.356	0.735	0.470	**0.778**	**0.526**
20	Math formulas	0.259	0.162	**0.661**	**0.544**	0.137	0.074	0.454	0.382	0.170	0.077	**0.599**	**0.404**
21	Science formulas	0.305	0.120	**0.315**	**0.250**	0.158	0.025	**0.239**	0.021	0.157	0.105	0.181	**0.250**
22	Computer science k.	0.262	0.128	**0.425**	**0.408**	0.151	0.137	0.147	0.134	**0.232**	**0.304**	0.220	0.278
23	Real-world k.	0.150	-	**0.472**	-	0.012	-	0.100	-	0.019	-	**0.160**	-
	Average score	0.384	0.384	**0.604**	**0.586**	0.204	0.177	0.394	0.238	0.252	0.268	**0.480**	**0.448**

Table 2: Evaluations of different baselines across 23 tasks in LĪLA. 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* method; the underlined score refers to the best score among *all* models. GPT3 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.

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.

Multi-task training also 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 2), and in one case (format) Neo-Multi's OOD performance on held-out tasks is better than its IID performance (Table 4). LĪLA's multi-task structure opens interesting future directions related to developing improved multitasking techniques, and further understanding its benefits.

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 7 for examples. We even see benefits of program synthesis in NLI, a classification-based task. LĪLA's unified problem format decouples synthesis from computation, while opening directions for further study on either aspect. More analysis is in Appendix C.

5 Conclusion

In this work, we introduce LĪLA, a unified mathematical reasoning benchmark for a holistic evaluation of AI agents. LĪLA 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 langauge perturbations via LĪLA-OOD and LĪLA-ROBUST resepctively. We also introduce BHĀSKARA, a 2.7B-parameter fine-tuned multi-task model, where we find that multi-tasking improves over single-task performance by 21.83% F1 score. The best performing model we evaluate achieves only 60.40% F1 indicating the potential for improvement on the proposed benchmark.

References

[1] Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. Muppet: Massive multi-task representations with pre-finetuning. *arXiv preprint arXiv:2101.11038*, 2021.

[2] Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*, 2019.

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

[4] Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow, March 2021.

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

[6] 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. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[7] 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. Evaluating large language models trained on code. 2021.

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

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

[10] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 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, 2019.

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

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

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

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

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

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

[17] Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 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, 2016.

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

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

[20] Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 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, 2016.

[21] Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 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, 2014.

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

[23] Bill Yuchen Lin, Seyeon Lee, Rahul Khanna, and Xiang Ren. Birds have four legs?! numersense: 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, 2020.

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

[25] Nicholas Lourie, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 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, 2021.

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

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

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

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

[30] Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. 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, July 2020. Association for Computational Linguistics.

[31] Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 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, 2020.

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

[33] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 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, 2022.

[34] Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Sachdeva, Peter Clark, Chitta Baral, and Ashwin Kalyan. Numglue: 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, 2022.

[35] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 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, June 2021. Association for Computational Linguistics.

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

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

[38] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 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, 2020.

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

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

[41] Subhro Roy and Dan Roth. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413*, 2016.

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

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

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

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

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

[47] 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. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

[48] Oyvind Tafjord, Peter Clark, Matt Gardner, Wen-tau Yih, and Ashish Sabharwal. 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, 2019.

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

[50] Shyam Upadhyay, Ming-Wei Chang, Kai-Wei Chang, and Wen-tau Yih. 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, 2016.

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

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

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

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

[55] Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hannaneh Hajishirzi, Yejin Choi, and Kyunghyun Cho. Naturalproofs: Mathematical theorem proving in natural language. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[56] Sean Welleck, Peter West, Jize Cao, and Yejin Choi. Symbolic brittleness in sequence models: on systematic generalization in symbolic mathematics. In *AAAI*, 2022.

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

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

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

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

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

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

[63] Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. "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, 2019.

A 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 [16], templated questions with equations as parameters [21] and other forms of arithmetic reasoning [19, 41, 50, 42, 43, 24]. Later datasets increase in complexity and scale, incorporating reading comprehension [10], algebra [46], and multi-modal contexts [28, 26, 27]. Still other numerical reasoning datasets focus on diversity [30] with multiple categories of numerical reasoning tasks [e.g., 2]. Most recently, new datasets have focused on incrasing difficulty, e.g., olympiad problems [14] and adversarial problems [35], as well as increasing the knowledge requirements to solve tasks, with a growing focus on commonsense reasoning [63, 61, 34].

A separate line of work in mathematical reasoning includes datasets testing mathematical theorem proving [e.g., 22, 58, 55, 62, 12]. 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 [57], including benchmarks [e.g., 52, 51, 9] and multitasking models [e.g., 29, 18, 25, 1]. NumGLUE [34] has been proposed as a multi-tasking numerical reasoning benchmark that contains 8 different tasks. LĪLA 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 [60] and NLP [39] which appear in large scale benchmarks [e.g., 47, 53].

B Data Creation

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 2 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-specifc 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*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 [14] 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 [46], this allows us to create diverse, compositional math problems with program annotations using a sophisticated grammar.

Expert program annotation. For many of 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 D.2.1 for details.

Instruction annotation. Given the effectiveness of instruction learning [33, 54, 45] 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 straight forward instruction, and *examples* that facilitate learning by demonstration [6]. Figure 2 shows an example instruction for the basic math task (§B.1).

B.1 Categories and Tasks

We create 4 *views*[8] or categories of LĪLA 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. Examples of each task and the datasets included are in Table 11 and Table 15 respectively.

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…"). Examples of each task and the datasets included are in Table 12 and Table 16 respectively.

We next partition datasets based on the type of *background knowledge*, required to solve the problem. For instance, commonsense questions like "How may 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 Table 13 and Table 17 respectively.

B.2 LĪLA-OOD

In order to measure if the model has truly learned the underlying mathematical reasoning skill, we evaluate both in-distribution (standard train-test splits) and out-of-distribution performance for each task, i.e., we evaluate on examples requiring the *same* underlying mathematical reasoning skill but from a different dataset. To construct LĪLA-OOD, we follow **(author?)** [5] and **(author?)** [15] by randomly splitting the datasets for each task into an IID and an OOD set, and using the IID set for training and standard evaluation and the OOD set to evaluate generalization. We do not include tasks in LĪLA-OOD for tasks containing only one dataset.

B.3 LĪLA-ROBUST

In light of recent work demonstrating the brittleness of language models at solving math problems [35], we create a high-quality evaluation dataset, LĪLA-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 10, on each task, yielding a set of challenge 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 [35] and 6 of our own. From each constituent dataset, we sample 20 questions and obtain perturbed question annotations via Amazon Mechanical Turk (AMT). Refer to Appendix D.2.1 for additional details on the construction of LĪLA-ROBUST.

B.4 Statistics

Table 3 shows key statistics of our proposed benchmark, LĪLA. LĪLA contains $\sim$135k examples with significant diversity across question, answer, program and instruction length (see detailed statistics in Table 19). Figure 5 shows the distribution of words across the 4 categories (i) math ability, (ii) language, (iii) knowledge and (iv) format (Section B.1). Figure 1 shows the diversity of questions in LĪLA. Note that, we downsample (via random selection) some datasets like AMPS [14] which contains numerous templated questions that can get over-representated in the distribution of examples across categories in LĪLA.

[8] Note that it is *not* a partition of the benchmark as each dimensions divides the constituent examples in different ways

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

Table 3: Key statistics of LĪLA.

Figure 1: Question n-gram distribution in LĪLA.

C Experiment Results

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 6 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 7).

Math ability: basic math
Language complexity: simple language
Format: generative question answering
Knowledge: no external knowledge

Instruction: You are given a question that involves the calculation of numbers. You need to perform either an addition or subtraction operation on the numbers. Generate your answer to the given question.

Question: Sara picked 45 pears and Sally picked 11 pears from the pear tree. How many pears were picked in total?

Program 1:

```
def solution(x, y):
    answer = x + y
    return answer
print(solution(45, 11)) # total pears is the sum of
pears with Sara and Sally
```

Program 2:

```
x = 45
y = 11
answer = x + y # total pears is the sum of pears with
Sara and Sally
print(answer)
```

Answer: 56

Figure 2: A data example with two Python programs in LĪLA. 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 LĪLA.

Dimension	Neo-A IID	Neo-A OOD	Neo-P IID	Neo-P OOD
Math ability	0.191	0.129	**0.445**	**0.188**
Language	0.189	0.147	**0.429**	**0.246**
Format	0.246	0.382	**0.372**	**0.404**
Knowledge	0.206	0.143	**0.331**	**0.213**
Average	0.208	0.200	**0.394**	**0.263**

Table 4: Multi-task models are able to generalize to unseen tasks in some categories. In general, program output (Neo-P) outperforms number output (Neo-A).

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 8).

C.1 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-

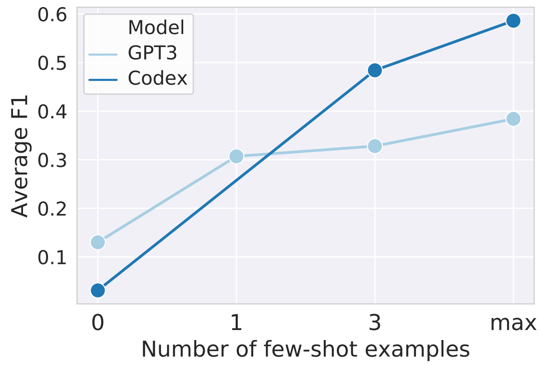


Figure 3: Average F1 scores of GPT-3 and Codex with different numbers of few-shot examples in LĪLA.

Dimension	Zero-shot w/o Inst.	Zero-shot w/ Inst.	Few-shot (3) w/o Inst.	Few-shot (3) 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

Table 5: 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.

division, number theory, and calculus. Codex shows analogous trends, except for performing very well on calculus (0.930)[9].

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-sense and math formulas and fails to do similarly on problems requiring other forms of external knowledge like physics, computer science, or real-world knowledge.

C.2 Results: Few-shot Prompting

Finally, we study the few-shot performance of much larger models ($\approx$175B), to better understand the performance of the smaller trained models ($\approx$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.

More examples and instructions improve prompts. 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 5 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

[9]Note that the training set for Codex is not known

instructions for mathematical reasoning. This is consistent with the findings of **(author?)** [36] on instruction-example equivalence.

Few-shot GPT-3 answer prediction underperforms fine-tuned GPT-Neo. 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 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.

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 ✗
Generated Program	

```
answer = 1346.0 + 6444.0
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 ✗
Generated Program	

```
a=20*6
b=12*8
c=a+b
answer= c
print(answer)
```

Executed Answer	216 ✓

Table 6: Examples using Neo-Multi.

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

Table 9: Dataset partitions (Categories) and associated tasks.

D Appendix

D.1 Qualitative Examples

Figures 6 and 7 give examples of input-output behavior of GPT-Neo fine-tuned on all the tasks. Figure 8 gives an example of a non-compiling output program.

D.2 Dataset Collection

D.2.1 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

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 scipy.stats import entropy

distribution1 = [0.149, 0.039, 0.099, 0.09, 0.203, 0.077, 0.039, 0.19]
distribution2 = [0.158, 0.256, 0.042, 0.008, 0.064, 0.051, 0.05, 0.072]
print(entropy(distribution1, qk=distribution2))
```

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
import numpy as np

a = np.array([
    [0, -2, -3],
    [0, 5, 0],
    [1, 3, 2]])
print(np.linalg.det(a))
```

Executed Answer	15.0 ✓

Table 7: Examples using Neo-Multi.

`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.

LĪLA-ROBUST To create the LĪLA-ROBUST dataset, we first define a set of 9 templates, consisting of 3 variation styles defined in SVAMP [35] 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 10.

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.

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

```
a=20*6
b=a*2
c=b*3
d=c*10
e=d*3
f=e*3
g=f+g
answer= g
print(answer)
```

Error: `NameError` (g is not defined)

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

```
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 8: `NameErrors` in Neo-Multi and Codex.

In particular, each HIT contains the 10 split questions from the original datasets, alongside the problem solution. Workers are asked to submit a 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 10. 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, we manually assess the worker augmentations produced for each template.

D.3 Dataset Statistics

Figure 5 gives relatives sizes of tasks within each category. Figure 6 illustrates the unigram frequencies in LĪLA, where larger words indicate higher frequency. Table 19 gives comprehensive statistics on each task. Table 21 cites each component dataset of LĪLA.

D.4 Additional Results

Table 20 gives the unaggregated performance of each model on each dataset in LĪLA (some datasets are split across tasks).

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 pinapple 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:** ThereTea 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 $x = 8$ **Variation:** Find the arclength of $f(x) = 2\sqrt{x}$ on $[2, 8]$

Table 10: Example for each template provided to MTurk workers to produce LĪLA-ROBUST

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
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
/ 2 ! 2 ! = 6 ; so p = 1 / 6. Answer: B.

Program:
```
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.

Task	Question category	Example
Task 1	Basic math: addition, subtraction, fact based QA etc.	**Original:** If Jimbo is 484 feet away from a beetle and quarter of 827 feet away from a grasshopper, which insect will seem bigger to him?? "Option 1": beetle, "Option 2" :grasshopper **Answer:** Option 2
Task 2	Muldiv: multiplication, division along with addition, subtraction etc.	**Question:** Mrs. Hilt bought 2 pizzas. Each pizza had 8 slices. So, she had __ total slices of pizza. **Answer:** 16
Task 3	Number theory: prime, power, negation, modulus and other operators etc.	**Question:** How many numbers are divisible by both 2 and 3 up to 300? **Answer:** 50
Task 4	Algebra: equations, functions, polynomials, series etc.	**Question:** The sum of the three smallest of four consecutive integers is 30 more than the largest integer. What are the four consecutive integers ? **Answer:** 15.0
Task 5	Geometry: triangles, polygons, 3D structures etc.	**Question:** A hall is 6 meters long and 6 meters wide. If the sum of the areas of the floor and the ceiling is equal to the sum of the areas of four walls, what is the volume of the hall (in cubic meters)? **Answer:** 108
Task 6	Statistics: binomial, divergence, mean, median, mode, variance etc.	**Question:** There are 11 boys and 10 girls in a class. If three students are selected at random, in how many ways that 3 girl and 2 boys are selected? **Answer:** 6600
Task 7	Calculus: differentiation, integration, gradient, series expansion etc.	**Question:** Let g(y) = 9*y**4 + 25*y**2 + 6. Let s(d) = 1 - d**4. Let x(t) = -g(t) + 6*s(t). What is the third derivative of x(f) wrt f? **Answer:** -360*f
Task 8	Linear algebra: vectors, dot products, Eigen vectors, matrices etc.	**Question:** Problem: Convert the following matrix to reduced row echelon form: $\begin{pmatrix} 7 & -2 & -10 & -4 \\ -5 & -10 & 2 & -7 \end{pmatrix}$. **Answer:** $\begin{pmatrix} 1 & 0 & -\frac{13}{10} & -\frac{13}{40} \\ 0 & 1 & \frac{9}{20} & \frac{69}{80} \end{pmatrix}$
Task 9	Advanced math: heuristics required along with probability, statistics, or algebra, Olympiad level problems	**Question:** Let $f(x) = 2^x$. Find $\sqrt{f(f(f(f(1))))}$. **Answer:** 256

Table 11: Example of each task in the *math ability* category of the LĪLA benchmark.

Task	Question category	Example
Task 10	No language	Compute the median of $4\sqrt{2}, -6, 3e, 3, -6, -\frac{14}{\sqrt{\pi}}, 6$. **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 LĪLA 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 LĪLA 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:** Apply functions 'mean' and 'std' to each column in dataframe 'df' **Answer:** df.groupby(lambda idx: 0).agg(['mean', 'std'])
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 LĪLA benchmark.

Task	Math category	IID	OOD

TASK 1	Basic math	`addsub.json`
		`Numersense_structured.json`
		`MCTaco_stationarity_structured.json`
		`MCTaco_frequency_structured.json`
		`MCTaco_event_typical_time_structured.json`
		`MCTaco_event_ordering_structured.json`
		`NumGLUE_Task7.json`
			`MCTaco_event_duration_structured.json`
			`NumGLUE_Task3.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`
		`amps_number_theory.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_basicmath.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 LĪLA across different math categories.

ID	Language category	IID	OOD
Task 10	No language	amps_number_theory.json amps_counting_and_stats.json amps_calculus.json amps_linear_algebra.json deepmind_mathematics_muldiv.json deepmind_mathematics_numbertheory.json deepmind_mathematics_algebra.json deepmind_mathematics_basicmath.json	amps_algebra.json deepmind_mathematics_calculus.json
Task 11	Simple language	addsub.json Numersense_structured.json MCTaco_stationarity_structured.json MCTaco_event_typical_time_structured.json MCTaco_event_ordering_structured.json MCTaco_event_duration_structured.json singleop.json multiarith.json asdiv.json GSM8k_structured.json APPS_structured.json mathqa_gain.json mathqa_other.json singleq.json simuleq.json NumGLUE_Task8.json draw_structured.json dolphin_structured.json mathqa_probability.json	MCTaco_frequency_structured.json NumGLUE_Task1.json mathqa_general.json NumGLUE_Task4.json
Task 12	Complex language	mathqa_physics.json APPS_structured.json mathqa_gain.json amps_number_theory.json mathqa_general.json conala_structured.json NumGLUE_Task5.json deepmind_mathematics_numbertheory.json	mbpp_structured.json mathqa_other.json

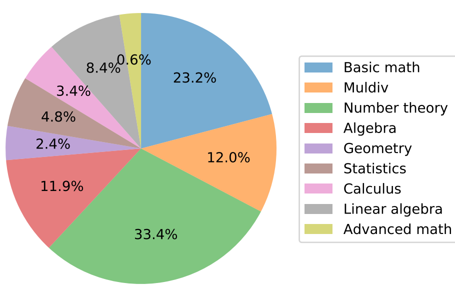
Table 16: Raw datasets used to create different tasks in Līla across different language categories.

ID	Format category	IID	OOD
Task 13	Fill in the blank	NumGLUE_Task4.json	Numersense_structured.json
Task 14	Generative QA	amps_number_theory.json amps_counting_and_stats.json amps_linear_algebra.json amps_algebra.json deepmind_mathematics_calculus.json addsub.json singleop.json multiarith.json asdiv.json GSM8k_structured.json APPS_structured.json mathqa_gain.json mathqa_other.json simuleq.json NumGLUE_Task8.json draw_structured.json dolphin_structured.json mathqa_probability.json MCTaco_frequency_structured.json NumGLUE_Task1.json mathqa_general.json mathqa_physics.json conala_structured.json amps_geometry.json MATH_crowdsourced.json deepmind_mathematics_calculus.json deepmind_mathematics_muldiv.json deepmind_mathematics_algebra.json deepmind_mathematics_basicmath.json	svamp_structured.json mathqa_geometry.json amps_calculus.json singleq.json NumGLUE_Task2.json mbpp_structured.json deepmind_mathematics_numbertheory.json
Task 15	MCQ	NumGLUE_Task3.json MCTaco_stationarity_structured.json MCTaco_event_ordering_structured.json MCTaco_event_duration_structured.json	MCTaco_event_typical_time_structured.json
Task 16	NLI	NumGLUE_Task5.json	
Task 17	RC	mathqa_physics.json	mbpp_structured.json

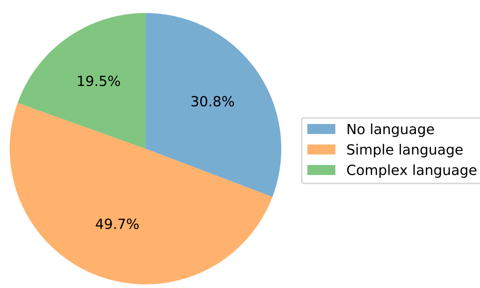
Table 17: Raw datasets used to create different tasks in Līla across different format categories.

ID	Knowledge category	IID	OOD
Task 18	No external knowledge	addsub.json singleop.json multiarith.json asdiv.json simuleq.json NumGLUE_Task8.json draw_structured.json dolphin_structured.json NumGLUE_Task5.json deepmind_mathematics_muldiv.json	NumGLUE_Task4.json GSM8k_structured.json svamp_structured.json NumGLUE_Task7.json
Task 19	Commonsense	Numersense_structured.json MCTaco_frequency_structured.json NumGLUE_Task3.json MCTaco_stationarity_structured.json MCTaco_event_duration_structured.json MCTaco_event_typical_time_structured.json	NumGLUE_Task1.json MCTaco_event_ordering_structured.json
Task 20	Math formulas	amps_number_theory.json amps_linear_algebra.json amps_algebra.json deepmind_mathematics_calculus.json mathqa_probability.json singleq.json mathqa_gain.json mathqa_other.json deepmind_mathematics_algebra.json deepmind_mathematics_basicmath.json deepmind_mathematics_calculus.json deepmind_mathematics_numbertheory.json	amps_counting_and_stats.json mathqa_general.json amps_calculus.json
Task 21	Science formulas	amps_geometry.json NumGLUE_Task2.json mathqa_physics.json	
Task 22	Computer science knowledge	APPS_structured.json conala_structured.json	mathqa_geometry.json
Task 23	Real-world knowledge	MATH_crowdsourced.json	mbpp_structured.json

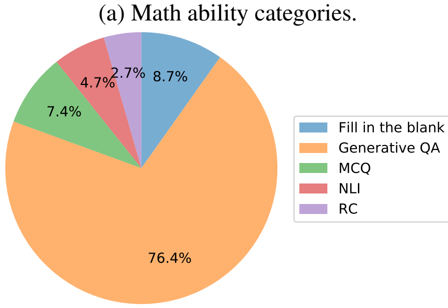
Table 18: Raw datasets used to create different tasks in LĪLA across different knowledge categories.



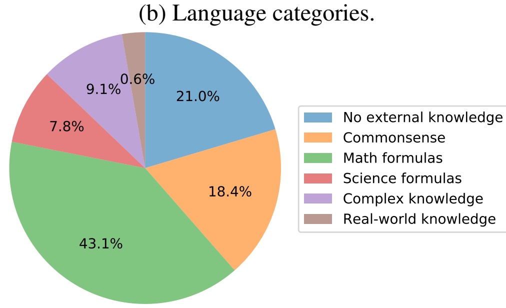
(a) Math ability categories.



(b) Language categories.



(c) Format categories.



(d) Knowledge categories.

Figure 5: Task diversity in LĪLA across math, language, format, and knowledge categories.

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

Table 19: Main statistics of LĪLA across the total of 23 tasks.

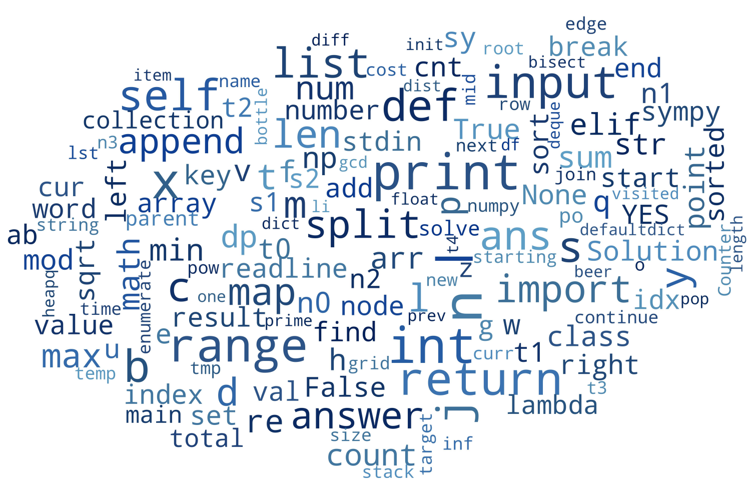


Figure 6: The word cloud distribution of annotated programs in the LĪLA dataset.

ID	Dataset	GPT3	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_geometry	**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_physics	0.120	0.087	**0.429**	0.210
25	mathqa_probability	**0.210**	0.003	0.000	0.200
26	mbpp_structured	0.128	0.175	0.164	**0.408**
27	MCTaco_event_duration_structured	**0.800**	0.773	0.773	0.710
28	MCTaco_event_ordering_structured	0.860	0.831	0.831	**0.890**
29	MCTaco_event_typical_time_structured	0.870	**0.881**	**0.881**	0.870
30	MCTaco_frequency_structured	**0.890**	0.862	0.862	0.790
31	MCTaco_stationarity_structured	0.710	**0.758**	**0.758**	0.670
32	multiarith	0.360	0.143	0.921	**0.990**
33	Numersense_structured	0.620	0.495	0.495	**0.660**
34	NumGLUE_Type_1	0.535	0.108	0.083	**0.740**
35	NumGLUE_Type_2	0.512	0.285	0.646	**0.735**
36	NumGLUE_Type_3	**0.835**	0.004	0.001	0.815
37	NumGLUE_Type_4	0.710	0.076	0.208	**0.790**
38	NumGLUE_Type_5	0.460	0.200	0.305	**0.615**
39	NumGLUE_Type_7	0.500	0.516	**0.854**	0.710
40	NumGLUE_Type_8	0.420	0.082	0.257	**0.610**
41	simuleq	0.120	0.074	0.010	**0.170**
42	singleop	0.940	0.347	0.611	**1.000**
43	singleq	**0.830**	0.143	0.474	0.670
44	svamp_structured	0.620	0.085	0.060	**0.790**
	Average F1 score	0.400	0.223	0.440	**0.613**

Table 20: Evaluation results of baselines across different single datasets. On most datasets, **Codex** performs best. Model names: **GPT3**: the few-shot 175B GPT-3 model; **GPT-Neo-A**: the fine-tuned 2.7B GPT-3 model where the prediction output is an answer; **GPT-Neo-P**: the fine-tuned 2.7B GPT-3 model where the prediction output is a program; **Codex**: the few-shot Codex model where the prediction output is a program.

ID	Dataset	References
1	addsub	[16]
2	amps	[14]
3	APPS	[13]
4	asdiv	[31]
5	conala	[59]
6	mathematics	[46]
7	dolphin	[17]
8	draw	[49]
9	GSM8k	[8]
10	MATH	[14]
11	mathqa	[2]
12	mbpp	[3]
13	MCTaco	[63]
14	multiarith	[40]
15	Numersense	[23]
16	NumGLUE	[34, 10, 37, 21, 48, 43, 42, 20, 19]
17	simuleq	[21]
18	singleop	[44]
19	singleq	[19]
20	svamp	[35]

Table 21: List of source datasets and corresponding references used in constructing LĪLA.