LLEMMA: an open language model for mathematics

Zhangir Azerbayev
Princeton University
za2514@princeton.edu

Hailey Schoelkopf
EleutherAI
hailey@eleuther.ai

Keiran Paster
University of Toronto, Vector Institute
keirp@cs.toronto.edu

Marco Dos Santos
University of Cambridge
mjad3@cam.ac.uk

Stephen McAleer

Albert Q. Jiang
University of Cambridge
qj213@cam.ac.uk

Jia Deng
Princeton University
jiadeng@princeton.edu

Stella Biderman
EleutherAI
stella@eleuther.ai

Sean Welleck
University of Washington
wellecks@uw.edu

Abstract

We present LLEMMA, a large language model for mathematics. We continue pretraining Code Llama on Proof-Pile-2, a mixture of scientific papers, web data containing mathematics, and mathematical code, yielding LLEMMA. On the MATH benchmark LLEMMA outperforms all known openly released models, as well as the unreleased Minerva model suite on an equi-parameter basis. Moreover, LLEMMA is capable of tool use and formal theorem proving without any finetuning. We openly release all artifacts, including 7 billion and 34 billion parameter models, the Proof-Pile-2, and code to replicate our experiments.[1]

1 Introduction

We present an open-source recipe for adapting a language model to mathematics through continued pretraining (Lewkowycz et al., 2022; Rozière et al., 2023) on the Proof-Pile-2, a diverse mixture of math-related text and code. Applying the recipe to Code Llama (Rozière et al., 2023) yields LLEMMA: 7 billion and 34 billion parameter base language models with substantially improved mathematical capabilities.

Specifically, our contributions are as follows:

1. We train the LLEMMA models: 7B and 34B parameter language models specialized for mathematics. The LLEMMA models are initialized with Code Llama (Rozière et al., 2023) weights, then further trained on the Proof-Pile-2, a new mixture of scientific papers, mathematics web pages (ope, 2023), and mathematical code comprising 53B unique tokens.
2. We evaluate LLEMMA on a standard suite of mathematical reasoning benchmarks. On the MATH benchmark (Hendrycks et al., 2021a), LLEMMA outperforms all known open-access language

[1] https://github.com/EleutherAI/math-lm

37th Conference on Neural Information Processing Systems (NeurIPS 2023) Workshop on MATH-AI.

models and is more performant than the unreleased Minerva model (Lewkowycz et al., 2022) per-parameter and per FLOP of math-specific training.

3. We demonstrate that LLEMMA is capable of using computational tools to solve mathematical problems, namely, calculators, computer algebra systems, and formal theorem provers.
4. Unlike prior mathematics language models such as Minerva (Lewkowycz et al., 2022), the LLEMMA model is open access and we open source our training data and code. This allows LLEMMA to serve as a platform for future research in mathematical reasoning.

Our work builds on Minerva, a suite of PaLM models whose pretraining was continued on mathematical text (Lewkowycz et al., 2022), but differs in several ways: (1) LLEMMA's training and evaluation covers a wider range of data and tasks, notably code data (e.g., the AlgebraicStack), tool use, and formal mathematics; (2) our work only depends on publicly accessible tools and data; (3) we provide new analyses related to the continued training data mixture, memorization, and additional supervised finetuning; (4) we make all artifacts publicly available for further research and use.

2 Approach

LLEMMA models are 7 billion and 34 billion parameter language models specialized for mathematics. Our approach is to continue pretraining Code Llama (Rozière et al., 2023) on Proof-Pile-2.

2.1 Data: Proof-Pile-2

We form Proof-Pile-2, a 53B-token dataset of mathematical text. The Proof-Pile-2 contains scientific papers via the RedPajama arXiv subset (Computer, 2023), web scrape data via OpenWebMath (ope, 2023), and code data via the newly-constructed AlgebraicStack. See Appendix A.1 for further details on AlgebraicStack.

2.2 Model and Training

Each model is initialized from Code Llama (Rozière et al., 2023). Code Llama models are decoder-only transformer language models initialized from Llama 2 (Touvron et al., 2023) and further trained on 500B tokens of code. We continue training the Code Llama models on Proof-Pile-2 using a standard autoregressive language modeling objective. We train the 7B model for 200B tokens, and the 34B model for 50B tokens. For a copmlete discussion of training hyperparameters, see Appendix E.

3 Evaluation

Our goal is to evaluate LLEMMA as a base model for mathematical text. To this end, we compare LLEMMA models using few-shot evaluation (Brown et al., 2020), and primarily focus on state-of-the-art models that have not been finetuned on supervised examples for the task. First, we evaluate the model's ability to solve mathematics problems using chain of thought reasoning (Wei et al., 2023) and majority voting (Wang et al., 2023). Our evaluations include MATH (Hendrycks et al., 2021b) and GSM8k (Cobbe et al., 2021), the de-facto standard benchmarks for evaluating quantitative reasoning in language models (Lewkowycz et al., 2022). Second, we explore few-shot tool use and formal theorem proving. Third, we study the effects of memorization and the data mixture. Appendix K contains a preliminary study of supervised finetuning with LLEMMA. Furthermore, in Appendix G we study the ability LLEMMA to leverage computational tools. In Appendix H we study how memorization may have influenced evaluation scores. Finally, in subsection H.1 we run ablations on the data mixture.

3.1 Chain-of-thought mathematical problem solving

We evaluate on a suite of tasks, including MATH (Hendrycks et al., 2021a) and (Cobbe et al., 2021), that involve generating self-contained text solutions to problems expressed in LaTeX or natural language, without using external tools (Lewkowycz et al., 2022). For a full description of this suite see Appendix F. For a qualitative example of a model output, see Figure 4.

We compare with Minerva (Lewkowycz et al., 2022), which continued pretraining the PaLM language model on a dataset of technical content; Code Llama, the initialization of LLEMMA's continued pretraining; and Llama 2, the initialization of Code Llama's continued pretraining on code. For open access models, we report scores computed using our evaluation suite, which is implemented as a fork of the Language Model Evaluation Harness (Gao et al., 2021). For Minerva models, we report benchmark scores from Lewkowycz et al. (2022).

Results. LLEMMA's continued pretraining on Proof-Pile-2 improves few-shot performance on the four mathematical benchmarks. LLEMMA improves over Code Llama by 20 percentage points on GSM8k and 13 points on MATH, and outperforms the proprietary Minerva model. Our approach also outperforms all open-weights language models at the time of writing. We conclude that continued pretraining on Proof-Pile-2 is effective for improving a pretrained model's ability to perform mathematical problem solving.

LLEMMA is pretrained on a diverse distribution of mathematics-related data, and is not tuned for a particular task. Therefore, we expect that LLEMMA can adapt to many other tasks via task-specific finetuning and few-shot prompting.

		GSM8k	**OCW**	**MMLU-STEM**	**SAT**	**MATH**
Llama 2	7B	11.8%	3.7%	29.9%	25.0%	3.2%
Code Llama	7B	10.5%	4.4%	25.1%	9.4%	4.5%
Minerva	8B	16.2%	7.7%	35.6%	-	14.1%
LLEMMA	7B	**36.4%**	**7.7%**	**37.7%**	**53.1%**	**18.0%**
Code Llama	34B	29.6%	7.0%	40.5%	40.6%	12.2%
LLEMMA	34B	**51.5%**	**11.8%**	**49.0%**	**71.9%**	**25.0%**
Minerva	62B	52.4%	12.0%	53.9%	-	27.6%
Minerva	540B	58.8%	17.6%	63.9%	-	33.6%

Table 1: Results on our five chain-of-thought reasoning tasks with samples generated via greedy decoding. Minerva results are quoted from Lewkowycz et al. (2022). Note that CodeLlama 7B performs worse than random guessing (25%) on MMLU and SAT, largely due to failing to conclude its chain of thought with a valid answer.

		GSM8k maj@k	**OCW** maj@k	**MMLU-STEM** maj@k	**SAT** maj@k	**MATH** maj@k
Minerva	8B	28.4%	12.5%	43.4%	-	25.4%
LLEMMA	7B	**54.0%**	**14.3%**	**49.9%**	**78.1%**	**33.5%**
LLEMMA	34B	**69.3%**	**18.4%**	**59.7%**	**81.3%**	**43.1%**
Minerva	62B	68.5%	23.5%	63.5%	-	43.4%
Minerva	540B	78.5%	30.8%	75.0%	-	50.3%

Table 2: Majority voting results for LLEMMA and Minerva. Minerva results are quoted from Lewkowycz et al. (2022). Voting is done with $k = 256$ for MATH, $k = 100$ for GSM8k and OCW, and $k = 16$ for MMLU-STEM and SAT. We sample with temperature $T = 0.6$ for $k = 256$ and $k = 100$ and $T = 0.3$ for $k = 16$, and use nucleus sampling with $p = 0.95$ (Holtzman et al., 2020). Due to compute constraints, we do not calculate majority voting scores for Llama 2 and Code Llama.

3.2 Formal mathematics

Interactive proof assistants such as Lean (de Moura et al., 2015), Isabelle (Wenzel et al., 2008), and Coq (Paulin-Mohring, 1989a,b) express mathematics in programming languages that allow for verification. These languages are data scarce compared to mainstream languages, especially in the context of pretraining. For instance, the Stack dataset used to pretrain language models in the BigCode project (Allal et al., 2023) has over 700 gigabytes of Python, compared to 322 megabytes of Lean. Proof assistants also provide information that is not present in raw source code, such as proof states that contain information about each step of a proof.

Proof-Pile-2's AlgebraicStack contains over 1.5 billion tokens of formal mathematics data, including proof states extracted from Lean and Isabelle formalizations. While a full investigation of formal math is outside the scope of this paper, we evaluate LLEMMA few-shot on two tasks:

- **Informal-to-formal proving** (Jiang et al., 2023), the task of generating a formal proof, given a formal statement, an informal LaTeX statement, and an informal LaTeX proof. The formal proof is checked by the proof assistant. We use the Isabelle proof assistant and evaluate on miniF2F (Zheng et al., 2021), a benchmark consisting of problem statements from Olympiads and undergraduate coursework. For the prompt, we use 11 (formal statement, informal statement, informal proof, formal proof) examples from Jiang et al. (2023), selecting 7 examples for number theory problems, and 6 examples for all others. We generate a single proof with greedy decoding.

- **Formal-to-formal proving** (e.g., Polu & Sutskever (2020)), the task of proving a formal statement by generating a sequence of proof steps (tactics). At each step, the input is a state x_t given by the proof assistant, and the language model's task is to generate a proof step y_t (a sequence of code). The proof step is checked by the proof assistant, yielding a new state x_{t+1} or an error message. The process continues, stopping if a proof is completed or a timeout is reached. We prompt the model using three (x_t, y_t) examples. We evaluate on miniF2F (Zheng et al., 2021) using the Lean 4 proof assistant, and use a standard best first search. See Appendix C for more details.

Results. As seen in Table 3, LLEMMA's continued pretraining on Proof-Pile-2 improved few-shot performance on the two formal theorem proving tasks.

Method	**Informal-to-formal**	
	miniF2F-valid	miniF2F-test
Sledgehammer	14.72%	20.49%
Code Llama 7b	16.31%	17.62%
Code Llama 34b	18.45%	18.03%
LLEMMA-7b	20.60%	22.13%
LLEMMA-34b	21.03%	21.31%

Method	**Formal-to-formal**	
	Search	miniF2F-test
ReProver*	1×64	26.50%
Code Llama 7b	1×32	20.49%
Code Llama 34b	1×32	22.13%
LLEMMA-7b	1×32	26.23%
LLEMMA-34b	1×32	25.82%

Table 3: Formal theorem proving tasks. *Left*: Informal-to-formal proving in Isabelle, showing the percentage of proven theorems with greedy decoding. *Right*: Formal-to-formal proving in Lean, showing the percentage of proven theorems with the given number of attempts × generations-per-iteration of best first search, and a 10-minute timeout. Sledgehammer (Paulson & Nipkow, 2023) is built-in Isabelle automation and ReProver (Yang et al., 2023) is a recent supervised model. ∗uses Lean 3.

On informal-to-formal proving, LLEMMA-7b closes 22.1% of the theorems, improving upon its Code Llama initialization and the Sledgehammer prover. The theorems that LLEMMA proves are often complementary to those proved with Sledgehammer: taking the union of Sledgehammer and LLEMMA proofs results in 26 new validation proofs (a 11 percentage-point increase), and 17 new test proofs (a 7 point increase); see Appendix Table 11. Prior to our work, the only demonstration of few-shot proof autoformalization used the proprietary Codex model (Jiang et al., 2023).

On Lean 4 formal-to-formal proving, LLEMMA-7b improves upon its Code Llama initialization, and performs similar to ReProver (Yang et al., 2023), a retrieval-augmented language model finetuned for tactic prediction. LLEMMA adapts to the task using a 3 example prompt, which to our knowledge is the first demonstration of few-shot tactic prediction for theorem proving.

4 Related Work

Language models for mathematics. Lewkowycz et al. (2022) trained large language models for mathematics tasks. Applying large language models to problems in mathematics is an active subfield of machine learning, including benchmarking mathematical knowledge and reasoning at varying levels (Hendrycks et al., 2021b; Zheng et al., 2021; Welleck et al., 2022; Azerbayev et al., 2023).

A number of recent works focus on supervised finetuning on specific task formats (e.g.,Yu et al. (2023); Yue et al. (2023)). Such work is orthogonal and complementary to ours, as improved base model substantially increased finetuned performance (see Appendix K).

Language models for formal mathematics. An ongoing line of work explores integrating language models with interactive theorem provers (Polu & Sutskever, 2020; Wu et al., 2022; Jiang et al., 2023; Welleck & Saha, 2023; Polu & Sutskever, 2020; Jiang et al., 2021, 2022; Han et al., 2022; Polu et al., 2022; Lample et al., 2022; Yang et al., 2023; First et al., 2023; Jiang et al., 2023). Our work provides a demonstration of few-shot proof autoformalization and tactic prediction, a large collection of formal mathematics data, along with an open access model for further exploring these directions.

5 Conclusion

We introduce LLEMMA and Proof-Pile-2, a novel base model and corpus for language modeling of mathematics. Our models, dataset, and code are openly available. We have shown that LLEMMA achieves state-of-the-art results for open-weights models on mathematical problem solving benchmarks, shown capabilities of using external tools via Python code, and demonstrated few-shot tactic prediction for theorem proving. We hope that LLEMMA and Proof-Pile-2 will be a useful base for future work on understanding language model generalization and dataset composition, investigating the limits of domain-specific language models, using language models as tools for mathematicians, and improving the mathematical capabilities of language models.

References

Openwebmath: An open dataset of high-quality mathematical web text. *arXiv preprint, forthcoming*, 2023.

Loubna Ben Allal, Raymond Li, Denis Kocetkov, Chenghao Mou, Christopher Akiki, Carlos Munoz Ferrandis, Niklas Muennighoff, Mayank Mishra, Alex Gu, Manan Dey, Logesh Kumar Umapathi, Carolyn Jane Anderson, Yangtian Zi, Joel Lamy Poirier, Hailey Schoelkopf, Sergey Troshin, Dmitry Abulkhanov, Manuel Romero, Michael Lappert, Francesco De Toni, Bernardo García del Río, Qian Liu, Shamik Bose, Urvashi Bhattacharyya, Terry Yue Zhuo, Ian Yu, Paulo Villegas, Marco Zocca, Sourab Mangrulkar, David Lansky, Huu Nguyen, Danish Contractor, Luis Villa, Jia Li, Dzmitry Bahdanau, Yacine Jernite, Sean Hughes, Daniel Fried, Arjun Guha, Harm de Vries, and Leandro von Werra. Santacoder: don't reach for the stars! In *Deep Learning for Code (DL4C) Workshop*, 2023.

Alex Andonian, Quentin Anthony, Stella Biderman, Sid Black, Preetham Gali, Leo Gao, Eric Hallahan, Josh Levy-Kramer, Connor Leahy, Lucas Nestler, Kip Parker, Michael Pieler, Jason Phang, Shivanshu Purohit, Hailey Schoelkopf, Dashiell Stander, Tri Songz, Curt Tigges, Benjamin Thérien, Phil Wang, and Samuel Weinbach. GPT-NeoX: Large scale autoregressive language modeling in PyTorch. GitHub Repo, 9 2023. URL `https://www.github.com/eleutherai/gpt-neox`.

Jeremy Avigad. The mechanization of mathematics. *Notices of the AMS*, 65(6):681–90, 2018.

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

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Stella Rose Biderman, Kieran Bicheno, and Leo Gao. Datasheet for the pile. *ArXiv*, abs/2201.07311, 2022.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023.

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

Katherine M. Collins, Albert Q. Jiang, Simon Frieder, Lionel Wong, Miri Zilka, Umang Bhatt, Thomas Lukasiewicz, Yuhuai Wu, Joshua B. Tenenbaum, William Hart, Timothy Gowers, Wenda Li, Adrian Weller, and Mateja Jamnik. Evaluating language models for mathematics through interactions. *arXiv preprint arXiv:2306.01694*, 2023.

Together Computer. Redpajama: An open source recipe to reproduce llama training dataset, April 2023. URL `https://github.com/togethercomputer/RedPajama-Data`.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. 2023.

Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris Van Doorn, and Jakob von Raumer. The lean theorem prover (system description). In *Automated Deduction-CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings 25*, pp. 378–388. Springer, 2015.

Erich Elsen, Curtis Hawthorne, and Arushi Somani. The adventure of the errant hardware, 2023. URL `https://www.adept.ai/blog/sherlock-sdc`.

Emily First, Markus N. Rabe, Talia Ringer, and Yuriy Brun. Baldur: Whole-proof generation and repair with large language models. *arXiv preprint arXiv:2303.04910*, 2023.

Leo Gao, Stella Rose 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*, abs/2101.00027, 2020.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, September 2021. URL `https://doi.org/10.5281/zenodo.5371628`.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*, 2022.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*, 2023.

Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III au2, and Kate Crawford. Datasheets for datasets, 2021.

Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training imagenet in 1 hour. *CoRR*, abs/1706.02677, 2017. URL `http://arxiv.org/abs/1706.02677`.

Kshitij Gupta, Benjamin Thérien, Adam Ibrahim, Mats L Richter, Quentin Anthony, Eugene Belilovsky, Irina Rish, and Timothée Lesort. Continual pre-training of large language models: How to (re) warm your model? *arXiv preprint arXiv:2308.04014*, 2023.

Jesse Michael Han, Jason Rute, Yuhuai Wu, Edward Ayers, and Stanislas Polu. Proof artifact co-training for theorem proving with language models. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=rpxJc9j04U`.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. 2021a.

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. *NeurIPS*, 2021b.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration, 2020.

Albert Q. Jiang, Wenda Li, Jesse Michael Han, and Yuhuai Wu. Lisa: Language models of isabelle proofs. *6th Conference on Artificial Intelligence and Theorem Proving*, 2021.

Albert Q. Jiang, Wenda Li, Szymon Tworkowski, Konrad Czechowski, Tomasz Odrzygóźdź, Piotr Miłoś, Yuhuai Wu, and Mateja Jamnik. Thor: Wielding hammers to integrate language models and automated theorem provers. *arXiv preprint arXiv:2205.10893*, 2022.

Albert Qiaochu Jiang, Sean Welleck, Jin Peng Zhou, Timothee Lacroix, Jiacheng Liu, Wenda Li, Mateja Jamnik, Guillaume Lample, and Yuhuai Wu. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=SMa9EAovKMC.

Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro von Werra, and Harm de Vries. The stack: 3 tb of permissively licensed source code. *Preprint*, 2022.

Guillaume Lample, Marie-Anne Lachaux, Thibaut Lavril, Xavier Martinet, Amaury Hayat, Gabriel Ebner, Aurélien Rodriguez, and Timothée Lacroix. Hypertree proof search for neural theorem proving. *arXiv preprint arXiv:2205.11491*, 2022.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.

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

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.

The mathlib Community. The lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2020, pp. 367–381, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370974. doi: 10.1145/3372885.3373824. URL https://doi.org/10.1145/3372885.3373824.

Scott Morrison. lean-training-data. https://github.com/semorrison/lean-training-data, 2023.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

Christine Paulin-Mohring. Extracting ω's programs from proofs in the calculus of constructions. In *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pp. 89–104, 1989a.

Christine Paulin-Mohring. *Extraction de programmes dans le Calcul des Constructions*. PhD thesis, Université Paris-Diderot-Paris VII, 1989b.

Larry Paulson and Tobias Nipkow. The sledgehammer: Let automatic theorem provers write your isabelle scripts!, 2023. URL https://isabelle.in.tum.de/website-Isabelle2009-1/sledgehammer.html.

Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*, 2020.

Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal mathematics statement curriculum learning. *arXiv preprint arXiv:2202.01344*, 2022.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press, 2020. ISBN 9781728199986. doi: 10.5555/3433701.3433727. URL https://dl.acm.org/doi/10.5555/3433701.3433727.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism. *Computing Research Repository*, 2019. doi: 10.48550/arXiv.1909.08053. URL https://arxiv.org/abs/1909.08053v4. Version 4.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2022.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Vincent Zhao, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Pranesh Srinivasan, Laichee Man, Kathleen Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed Chi, and Quoc Le. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

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*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

Sean Welleck. Neural theorem proving tutorial. https://github.com/wellecks/ntptutorial, 2023.

Sean Welleck and Rahul Saha. llmstep: Llm proofstep suggestions in lean. https://github.com/wellecks/llmstep, 2023.

Sean Welleck, Jiacheng Liu, Ximing Lu, Hannaneh Hajishirzi, and Yejin Choi. Naturalprover: Grounded mathematical proof generation with language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=rhdfTOiXBng.

Makarius Wenzel, Lawrence C Paulson, and Tobias Nipkow. The isabelle framework. In *Theorem Proving in Higher Order Logics: 21st International Conference, TPHOLs 2008, Montreal, Canada, August 18-21, 2008. Proceedings 21*, pp. 33–38. Springer, 2008.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Norman Rabe, Charles E Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=IUikebJ1Bf0.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *arXiv preprint arXiv:2305.10429*, 2023.

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. LeanDojo: Theorem proving with retrieval-augmented language models. In *Neural Information Processing Systems (NeurIPS)*, 2023.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mammoth: Building math generalist models through hybrid instruction tuning. *CoRR*, abs/2309.05653, 2023. doi: 10.48550/arXiv.2309.05653. URL https://doi.org/10.48550/arXiv.2309.05653.

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

Data source	Tokens	Weight
Proof-Pile-2	53B	–
Code (AlgebraicStack)	10B	1.00
Web (OpenWebMath)	14B	4.00
Papers (ArXiv)	29B	2.00
General code (RedPajama)	59B	0.22
General language (Pile)	300B	0.15

Table 4: Proof-Pile-2 data sources (top), general language and code data included during training (bottom), and the mixture weights of each component during training.

Language	AlgebraicStack tokens
Agda	35.2 M
C	25.1 M
C++	954.1 M
Coq	281.9 M
Fortran	724.9 M
GAP	3.6 M
Haskell	9.1 M
Idris	10.9 M
Isabelle	1089.7 M
Julia	531.0 M
Jupyter	199.1 M
Lean	285.6 M
Maple	2.0 M
Matlab	65.8 M
Python	6098.8 M
R	71.3 M
Tex	567.7 M
Total	10955.7 M

Table 5: Tokens in AlgebraicStack, computed with the Llama tokenizer.

A Data: Proof-Pile-2

A.1 Mathematical code: AlgebraicStack

AlgebraicStack contains over 10B tokens of code related to mathematics. We describe its sources, filtering, and content below. Table 5 shows the number of tokens per language in AlgebraicStack.

Model	Adaptation tokens	Open
Minerva-8b	164B	✗
Minerva-62b	109B	✗
LLEMMA-7b (ours)	200B	✓
LLEMMA-34b (ours)	50B	✓

Dataset	Tokens	Open
Minerva Dataset	38.5B	✗
Proof-Pile-2 (ours)	53B	✓
Code (AlgebraicStack)	10B	✓
OpenWebMath (ope, 2023))	14B	✓
ArXiv (Computer, 2023))	29B	✓

Figure 1: Comparison of LLEMMA and Minerva training

A.1.1 Composition

Scientific papers. We use the ArXiv subset of RedPajama (Computer, 2023), an open-access reproduction of the LLaMA training dataset. The ArXiv subset contains 29B tokens.

Web data. We use OpenWebMath (ope, 2023), a 14B-token dataset of high-quality web pages filtered for mathematical content. OpenWebMath filters CommonCrawl web pages based on math-

related keywords and a classifier-based math score, preserves mathematical formatting (e.g., LaTeX, AsciiMath), and includes additional quality filters (e.g., perplexity, domain, length) and near-deduplication. Refer to ope (2023) for a full description of OpenWebMath.

Code. Computational tools such as numerical simulations, computer algebra systems, and formal theorem provers are of ever increasing importance to mathematicians (Avigad, 2018). Motivated by this fact, we create AlgebraicStack, a 10B-token dataset of source code from 17 languages, spanning numerical, symbolic, and formal math. The dataset consists of filtered code from the Stack (Kocetkov et al., 2022), public GitHub repositories, and formal proofstep data. Table 5 shows the number of tokens by language in AlgebraicStack.

General natural language and code data. Following Lewkowycz et al. (2022), our training mixture consists of a small amount of general domain data, which functions as a form of regularization. Since the pretraining dataset for LLaMA 2 is undisclosed, we use the Pile (Gao et al., 2020; Biderman et al., 2022) as a surrogate training dataset.[2] We set 95% of our training mixture to be the Proof-Pile-2, 2% to be from the Pile (with ArXiv removed, as it is separately in Proof-Pile-2), and 3% to be the GitHub subset of RedPajama (Computer, 2023).

A.1.2 GitHub code

The following programming languages were either barely present in the Stack or consisted of largely incorrect filetypes, so we downloaded data for these languages directly via the Github Python API.

- **Coq** : We filter for files with the `.v` extension, and include Coq via including files that match a heuristic filter for the keywords **"Theorem"**, **"Proof"**, **"Qed"**, **"Inductive"**, **"Definition"**, **"Fixpoint"** and exclude Verilog files via the keyword blacklist **"pragma"**, **"endmodule"**, **"posedge"**, **"negedge"**, **"wire"**. We additionally exclude files noted as automatically generated.
- **Isabelle** : We filter for files with the `.thy` extension and include files matching the keyword whitelist **"theorem "**, **"lemma "**. We keep only `isabelle-prover/mirror-afp-devel` and discard all other older copies of the Archive of Formal Proofs. We further remove theorem statements and proofs that have a theorem name in the PISA (Jiang et al., 2021) test set.
- **Lean** : We filter for files with the `.lean` extension, using the keyword whitelist **"theorem "**, **"lemma "**, **"example "**. We remove all dependency files, and in order to avoid known benchmark contamination, we blacklist the `ProofNet` and `MiniF2F` repositories. We further remove theorems or lemmas that share a theorem name with the LeanDojo val or test sets.
- **MATLAB** : We filter for files with the `.m` extension, using the keyword whitelist **"#import"**, **"interface"**, **"implementation"**, **"property"**, and blacklist C files via the keywords **"#include"** and the regex `r' main\(.*{$'`

We implemented a cutoff date for our Github API downloads, and used a cutoff date of April 1, 2023.

For all languages, unless otherwise stated, we additionally filtered out files with a filesize greater than 1048575 bytes or with a numerical density (ratio of digit characters to non-digit characters) of 0.5. We additionally perform document-level exact deduplication by removing documents which contain an overlapping 2048-character chunk as another document.

A.1.3 Lean proofsteps

We extract a dataset of (tactic state, next tactic) pairs from Mathlib 4 (mathlib Community, 2020) using the `lean-training-data` (Morrison, 2023) tool. We use Mathlib 4 commit `c779bd5`, which was created on August 20th 2023.

[2] Gupta et al. (2023) found that continued pretraining with slightly different datasets is non-problematic. Previous papers (Belrose et al., 2023; Chen et al., 2023) have used the Pile as a stand-in for unreleased pretraining datasets.

A.1.4 Isabelle Proofsteps

We construct a dataset of Isabelle proofs, building upon the PISA dataset Jiang et al. (2021). Isabelle Proofsteps comprises proofs from the Archive of Formal Proofs and Isabelle Standard Library, scraped with PISA Jiang et al. (2021). Each entry in the dataset includes the theorem statement, the proof states and the proof steps, separated by specific tags. To maintain the integrity of evaluations using the PISA test set, we decontaminate Isabelle Proofsteps by removing theorems whose names overlap with those in the PISA test set. Although this approach results in a strict filtering – removing more than 10,000 theorems although there are only 3600 in the PISA test set – we consider it acceptable in order to mitigate data contamination. After filtering, Isabelle Proofsteps contains 251,000 theorems.

A.1.5 Stack Filtering

We source the following programming languages from the Stack (Kocetkov et al., 2022) dataset, and describe our filtering process and quality issues we chose to mitigate beyond our default quality heuristics:

- **Agda**: Only standard filters applied.
- **C** : We include documents based on a keyword whitelist, namely: **"#include <fftw.h>"**, **"#include <fftw3.h>"**, **"#include <rfftw.h>"**, **"#include <gsl"**, **"#include <cblas.h>"**, **"#include <blas.h>"**, **"#include <lapacke.h>"**, **"#include <nlopt.h>"**, **"#include <petsc.h>"**.
- **C++** : We include documents based on a keyword whitelist, namely: **"#include <adept_arrays.h>"**, **"#include <adept.h>"**, **"#include <alglib>**, **"#include <boost"**, **"#include <armadillo"**, **"#include <blitz"**, **"#include <Eigen"**, **"#include <deal.II"**, **"#include <dlib"**, **"#include <NTL"**, **"#include <mtl"**.
- **Fortran** : Only standard filters applied.
- **GAP** : Only standard filters applied.
- **Haskell** : We filtered the data to only contain files with the following imports: **Numeric.LinearAlgebra**, **Numeric.SpecFunctions**, **Numeric.Vector**, **Statistics**, **Data.Complex**.
- **Idris** : Only standard filters applied.
- **Julia** : We filtered out mislabeled JSON lines files. We removed files larger than 10,000 characters long which both were not files containing tests and which had a lower numerical density than 0.5, and otherwise ignored numerical density. We additionally only accepted files within a specific keyword whitelist, to attempt to control relevance to scientific computing, namely: **"LinearAlgebra"**, **"DifferentialEquations"**, **"Symbolics"**, **"Distributions"**, **"DataFrames"**, **"DynamicalSystems"**, **"Turing"**, **"Gen"**, **"JuMP"**, **"sqrt"**, **"abs"**, **"zeros"**, **"ones"**, **"sin"**, **"cos"**, **"tan"**, **"log"**, **"exp"**, **"integrate"**, **"likelihood"**, **"Matrix"**, π, **"pi"**, **"rand"**, **"grad"**.
- **Jupyter** : We found that many Jupyter notebook files were large due to containing long cell outputs, such as base64 images, long tracebacks, or other extra JSON cell metadata. We use `nbconvert` to convert notebooks to a markdown format, removing metadata.
- **Maple** : We filtered out files with a size greater than $100,000$ bytes, and found that some files were XML. We filtered all files beginning with an XML declaration.
- **Python** : We filtered notebooks and JSON files out by excluding documents with beginning **"{"** characters, and included only files importing from the following libraries: **numpy**, **scipy**, **sympy**, **sage**, **numba**, **numexpr**. We chose not to incorporate files importing from common deep learning frameworks because of low data diversity.
- **R** : We excluded all files beginning with an XML declaration. We additionally filtered out all notebooks, and filtered all files containing MacOS "Resource Fork" files.
- **Tex** : We used a max file size of 10,000,000 bytes. We excluded tex files found in directories named **"latex/"** because these were often auto-generated files, and excluded documents using **gnuplot**. We included only documents containing one of the keywords **"\chapter{"**, **"\chapter*{"**, **"\section{"**, **"\section*{"**, **"\subsection{"**, **"\subsection*{"**,

"\subsubsection{", **"\subsubsection*{"**, **"\paragraph{"**, **"\subparagraph{"**, and additionally only included documents identified as English by a classifier from the `langid` package.

For all languages we used within the Stack, unless otherwise stated, we additionally filtered out files with a filesize greater than 1048575 bytes or with a numerical density (ratio of digit characters to non-digit characters) of 0.5.

We used v1.2 of the near-deduplicated Stack as a base for processing.

A.2 Papers: Arxiv

We use the entirety of ArXiv, as accessed by Computer (2023) in April 2023. For further information on preprocessing applied to ArXiv, see Computer (2023).

A.3 Web: OpenWebMath

For the web portion of our training dataset, we use OpenWebMath (ope, 2023).

B Evaluation Harness

We implement a variety of math-related tasks and evaluation protocols into a public fork of the Language Model Evaluation Harness (Gao et al., 2021). The Harness provides a model-agnostic framework for standardized, reproducible evaluation of language models.

We add the following tasks for the evaluations in this paper:

- `hendrycks_math_ppl`: Perplexity evaluation on MATH (Hendrycks et al., 2021a) sub-tasks.
- `minif2f_isabelle`: Proof autoformalization in Isabelle on the miniF2F benchmark based on Jiang et al. (2023), with a Portal-to-Isabelle (Jiang et al., 2021) proof checker.
- `minerva_math`: The MATH benchmark with the prompt and Sympy evaluation from Minerva (Lewkowycz et al., 2022).
- `minerva-hendrycksTest`: MMLU-STEM tasks following Lewkowycz et al. (2022).
- `ocw_courses`: The OCW Courses task from Lewkowycz et al. (2022).
- `python_gsm8k`: GSM8k with Python, based on Gao et al. (2022).
- `sympy_math`: MATH with Sympy evaluation.

We include a link to the implementations for these tasks, including full prompts, in our public codebase.

C Evaluation: Experiment Details

C.1 Isabelle Informal-to-Formal Theorem Proving

We follow Jiang et al. (2023), allowing the model to issue a call to built-in Isabelle automation in the output proof by generating `sledgehammer`. This calls Sledgehammer (Paulson & Nipkow, 2023) and the list of heuristics listed in Jiang et al. (2023). Following Jiang et al. (2023), as a baseline we use Sledgehammer and the heuristics executed at the beginning of the proof (referred to as Sledgehammer in the main text for brevity). We use a 30-second timeout for Sledgehammer and implement proof checking via Portal-to-Isabelle (Jiang et al., 2021). Refer to the implementation in the Evaluation Harness for further details.

C.2 Lean Theorem Proving

Theorem proving via tactic prediction involves interacting with a proof assistant after each step of a proof. Implementing these interactions within the evaluation harness is outside the scope of this

Problem (MATH Number theory 185): When a number is divided by 5, the remainder is 3. What is the remainder when twice the number is divided by 5? Show that it is 1.

Human-written informal proof: If our number is n, then $n \equiv 3 \pmod 5$. This tells us that

$$2n = n + n \equiv 3 + 3 \equiv 1 \pmod 5.$$

The remainder is 1 when the number is divided by 5.

Informal-to-formal (Isabelle):
{Problem, human-written informal proof}

```
theorem mathd_numbertheory_185:
  fixes n ::nat
  assumes "n mod 5 = 3"
  shows "(2 * n) mod 5 = 1"
```

```
proof -
have "2 * n = n + n"  <ATP>
also have "... mod 5 =
    (n mod 5 + n mod 5) mod 5" <ATP>
also have "... = (3 + 3) mod 5"
    using assms <ATP>
also have "... = 1" <ATP>
finally show ?thesis <ATP>
qed
```

Formal-to-formal (Lean 4):

```
theorem mathd_numbertheory_185
  (n : ℕ) (h₀ : n % 5 = 3)
  : 2 * n % 5 = 1 := by

-- INPUT (step 1):
--   n: ℕ
--   h₀: n % 5 = 3
--   ⊢ 2 * n % 5 = 1
rw [mul_mod, h₀]

-- INPUT (step 2):
--   n: ℕ
--   h₀: n % 5 = 3
--   ⊢ 2 % 5 * 3 % 5 = 1
simp only [h₀, mul_one]
```

Figure 2: Example formal proofs from LLEMMA-7b. *Left:* The model is given a problem, informal proof, and formal statement, following Jiang et al. (2023). It generates a formal proof (starting with `proof -`) containing Isabelle code and calls to automation (shown as *<ATP>*). *Right:* The model is given a proof state, visualized as a grey comment, and generates the subsequent step (e.g. `rw [..]`).

work. Therefore, for the Lean theorem proving task we use a separate evaluation setup based on an open-source implementation (Welleck, 2023).

Setup. We evaluate on miniF2F (Zheng et al., 2021), which consists of 488 formalized statements from math competitions and undergraduate coursework. Given a formalized statement, the task is to generate a formal proof that is checked by Lean.

We use best first search, commonly used for neural tactic prediction models (e.g., Polu & Sutskever (2020)). Best first search is parameterized by the number of attempts (N), generated tactics per iteration (S), and maximum iterations (T). We define the *search budget* to be the maximum number of generated tactics, $N \times S \times T$. We set our search budget to $N = 1$, $S = 32$, and $T = 100$, less than that of the baseline model. Following Yang et al. (2023), we generate tactics with beam search and use a 10 minute timeout. We adapt the proof search implementation from Welleck (2023), which uses LeanDojo v.1.1.2 (Yang et al., 2023) for interaction. We use Lean 4 miniF2F, using `https://github.com/rah4927/lean-dojo-mew` commit `d00c776260c77de7e70125ef0cd119de6c0ff1de`.

Prompt. We prompt the model with three (state, tactic) examples, shown in Figure 3. For concrete examples of model outputs, see Figure 2.

```
"""Given the Lean 4 tactic state, suggest a next tactic.
Here are some examples:

Tactic state:
---
α : Type u_1
r : α → α → Prop
inst¹ : DecidableEq α
inst : IsIrrefl α r
⊢ CutExpand r ≤ InvImage (Finsupp.Lex (r  fun x x_1 => x ≠ x_1)
    fun x x_1 => x < x_1) ↑toFinsupp
---
Next tactic:
---
rintro s t ⟨u, a, hr, he⟩
---

Tactic state:
---
ι : Type u_1
I J : Box ι
x y : ι → ℝ
I J : WithBot (Box ι)
⊢ ↑I = ↑J ↔ I = J
---
Next tactic:
---
simp only [Subset.antisymm_iff, ← le_antisymm_iff,
    withBotCoe_subset_iff]
---

Tactic state:
---
m n : ℕ
h : Nat.coprime m n
⊢ Nat.gcd m n = 1
---
Next tactic:
---
rw [← h.gcd_eq_one]
---

Tactic state:
---
%s
---
Next tactic:
---"""
```

Figure 3: Prompt for the Lean theorem proving experiments.

D Datasheet

We provide a datasheet for Proof-Pile-2, following the framework in Gebru et al. (2021).

MOTIVATION	
For what purpose was the dataset created?	Proof-Pile-2 was created for the training or finetuning of domain-specific large language models for general mathematics tasks.
Who created the dataset and on behalf of which entity?	The dataset was created by the authors of this paper for the purposes of this research project.
Who funded the creation of the dataset?	The creation of the dataset was funded by the coauthors' grants and employers. [anonymized].
Any other comment?	
COMPOSITION	
What do the instances that comprise the dataset represent?	Instances are text-only documents.
How many instances are there in total?	We detail fine-grained token counts elsewhere in this paper.
Does the dataset contain all possible instances or is it a sample (not necessarily random) of instances from a larger set?	Our dataset is filtered based on our assessments of quality for the language modeling task. More detail on methodology can be found in Appendix A.
What data does each instance consist of?	Each instance is a text-only document, alongside metadata about its originating split and filename or location.
Is there a label or target associated with each instance?	No.
Is any information missing from individual instances?	Yes, we filter undesired noise, such as base64-encoded images, from some documents.
Are relationships between individual instances made explicit?	No.
Are there recommended data splits?	Yes, we release a canonical train, validation, and test split of the dataset, which we follow in this work.
Are there any errors, sources of noise, or redundancies in the dataset?	We make our best efforts to remove errors or sources of noise, but our dataset will naturally contain documents with errors or noise, and may contain near-duplicate documents.
Is the dataset self-contained, or does it link to or otherwise rely on external resources?	The dataset is self-contained, but can also be reconstructed based on external publicly available data sources and datasets following our instructions.
Does the dataset contain data that might be considered confidential?	All documents in Proof-Pile-2 are publicly available online.

Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety?	We estimate toxic content to be less prevalent in our dataset than other more general web-based datasets, due to its technical focus. However, it is likely to contain such content.
COLLECTION	
How was the data associated with each instance acquired?	Data was largely sourced from existing public subsets, such as the RedPajama dataset (Computer, 2023), OpenWebMath dataset (ope, 2023), and via filtering the Stack (Kocetkov et al., 2022). Some data was collected using the Github API.
What mechanisms or procedures were used to collect the data?	See above.
If the dataset is a sample from a larger set, what was the sampling strategy?	We release the entirety of the dataset following the application of our quality filters. We randomly held out validation and test splits from the dataset.
Who was involved in the data collection process and how were they compensated?	The authors of this paper participated in locating, retrieving, and filtering the dataset.
Over what timeframe was the data collected?	This data was collected in 2023, with a cutoff date of April 2023 for all subsets with the exception of our Lean proofstep data.
Were any ethical review processes conducted?	No.
PREPROCESSING	
Was any preprocessing/cleaning/labeling of the data done?	Yes, the authors extensively filtered the dataset subsets in keeping with our expectations for high-quality language modeling data in our domain. See Appendix A for further detail on filtering steps taken.
Was the "raw" data saved in addition to the preprocessed/cleaned/labeled data?	Raw data can be accessed via reuse of our provided codebase.
Is the software that was used to preprocess/clean/label the data available?	Yes. We release our codebase, which can be used to reproduce our dataset and its construction process, at `anonymized`.
USES	
Has the dataset been used for any tasks already?	Yes, this dataset has been used to train the LLEMMA language models as a domain adaptation and continued pretraining corpus.
Is there a repository that links to any or all papers or systems that use the dataset?	No.
What (other) tasks could the dataset be used for?	The dataset was specifically targeted as a high quality language modeling corpus for the mathematics domain, but may be useful for general-purpose language modeling or unforeseen other downstream uses.

Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses?	We filtered the dataset with the intent of creating a model useful for mathematical tasks with solely English text.
Are there tasks for which the dataset should not be used?	The dataset should not be used with the intent to cause harm or for models intended for the purposes of harm.
DISTRIBUTION	
Will the dataset be distributed to third parties outside of the entity on behalf of which the dataset was created?	We intend to make the dataset publicly available for reproducibility, analysis, and other further downstream uses.
How will the dataset will be distributed?	We will provide code to replicate the dataset, and plan to release it via the Huggingface Hub.
When will the dataset be distributed?	The dataset will be available immediately upon publication.
Will the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)?	We do not relicense the dataset's components, and do not impose our own use restrictions.
Have any third parties imposed IP-based or other restrictions on the data associated with the instances?	Not to our knowledge.
Do any export controls or other regulatory restrictions apply to the dataset or to individual instances?	Not to our knowledge.
MAINTENANCE	
Who will be supporting/hosting/maintaining the dataset?	The dataset will be hosted on the HuggingFace Hub and able to be recreated via code at `anonymized`. The dataset will not be updated post-release.
How can the owner/curator/manager of the dataset be contacted?	`anonymized`.
Is there an erratum?	No.
Will the dataset be updated?	No.
If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so?	No.

Table 6: **Datasheet for** Proof-Pile-2, following the framework introduced by Gebru et al. (2021).

E Training Hyperparameters

We train all models in bfloat16 mixed precision using the GPT-NeoX library (Andonian et al., 2023) across 256 A100 40GB GPUs. We use Tensor Parallelism (Shoeybi et al., 2019) with a world size of 2 for LLEMMA-7B , and a world size of 8 for LLEMMA-34B, alongside ZeRO Stage 1 sharded optimizer states (Rajbhandari et al., 2020) across Data Parallel (Goyal et al., 2017) replicas. We use Flash Attention 2 (Dao, 2023) to improve throughput and further reduce memory requirements.

LLEMMA 7B is trained for $42,000$ steps with a global batch size of 4 million tokens and a 4096 token context length. This corresponds to roughly $23,000$ A100-hours. The learning rate is warmed up to $1 \cdot 10^{-4}$ over 500 steps, then set to cosine decay to $1/30$th of the maximum learning rate over $48,000$ steps. The reason for the discrepancy between the number of training steps and the scheduler length is that we planned to train for $48,000$ steps, but encountered NaN losses after step $42,000$, likely caused by unstable optimization or hardware failures (Elsen et al., 2023).

LLEMMA 34B is trained for $12,000$ steps with a global batch size of 4 million tokens and a 4096 context length. This corresponds to roughly $47,000$ A100-hours. The learning rate is warmed up to $5 \cdot 10^{-5}$ over 500 steps, then decayed to $1/30$th the peak learning rate.

Before training LLEMMA 7B, we contract the RoPE (Su et al., 2022) base period of the Code Llama 7B initialization from $\theta = 1,000,000$ to $\theta = 10,000$. This is so that the long context finetuning procedure described in Rozière et al. (2023) can be repeated on the trained LLEMMA 7B (we leave actually doing so to future work). Due to compute constraints, we were unable to verify that training LLEMMA 34B with a contracted RoPE base period did not come with a performance penalty, therefore for that model we left $\theta = 1,000,000$.

F Chain of Thought Evaluation Suite

Input:
Let $f(r) = \sum_{j=2}^{2008} \frac{1}{j^r} = \frac{1}{2^r} + \frac{1}{3^r} + \cdots + \frac{1}{2008^r}$. Find $\sum_{k=2}^{\infty} f(k)$.

LLEMMA 34B solution:
We have

$$\begin{aligned}
\sum_{k=2}^{\infty} f(k) &= \sum_{k=2}^{\infty} \sum_{j=2}^{2008} \frac{1}{j^k} = \sum_{j=2}^{2008} \sum_{k=2}^{\infty} \frac{1}{j^k} = \sum_{j=2}^{2008} \frac{1}{j^2} \sum_{k=0}^{\infty} \frac{1}{j^k} = \sum_{j=2}^{2008} \frac{1}{j^2} \frac{1}{1-\frac{1}{j}} \\
&= \sum_{j=2}^{2008} \frac{1}{j(j-1)} = \sum_{j=2}^{2008} \left(\frac{1}{j-1} - \frac{1}{j} \right) \\
&= \left(\frac{1}{1} - \frac{1}{2} \right) + \left(\frac{1}{2} - \frac{1}{3} \right) + \cdots + \left(\frac{1}{2007} - \frac{1}{2008} \right) \\
&= 1 - \frac{1}{2008} \\
&= \boxed{\frac{2007}{2008}}.
\end{aligned}$$

Final Answer: The final answer is $\frac{2007}{2008}$.

Figure 4: Example of a LLEMMA 34B solution to a MATH (Hendrycks et al., 2021a) problem. This problem is tagged with difficulty level 5, the highest in MATH. The model was conditioned on the 4-shot prompt described in subsection 3.1, and the solution was produced by greedy decoding. Notice that the model had to apply two nontrivial steps to solve this problem: (1) noticing that swapping the order of summation simplifies the problem, and (2) noticing that the resulting sum telescopes.

- **MATH** (Hendrycks et al., 2021b), a dataset with 12.5k problems (5k evaluation) from high-school math competitions. Given a problem statement, the model generates a LaTeXsolution and an answer

that must match a reference answer. We follow Lewkowycz et al. (2022), using their four-example prompt and evaluating answers using `Sympy` equivalence.

- **GSM8k** (Cobbe et al., 2021), a dataset of middle-school level math word problems. We use the 8-shot prompt from Wei et al. (2023), as Lewkowycz et al. (2022) do not specify their evaluation prompt or number of few-shot examples.
- **OCWCourses** (Lewkowycz et al., 2022), a collection of undergraduate-level STEM problems harvested from MIT's OpenCourseWare. We use the four-example prompt provided by (Lewkowycz et al., 2022).
- **MMLU-STEM** (Hendrycks et al., 2021a), a subset of 18 out of 57 subjects in the MMLU benchmark. We follow Lewkowycz et al. (2022) and use their provided four-example chain-of-thought prompt.
- **SAT**, we create a dataset consisting of the 32 math questions that do not contain figures from the May 2023 College Board SAT examination, which is after our model's knowledge cutoff.

G Mathematical problem solving with tool use

We evaluate the model's ability to solve problems given access to computational tools. We evaluate the following:

- **MATH+Python**, the model is prompted to alternately describe a solution step in natural language, then execute that step with code. The final answer is a program that executes to a numeric type or a SymPy object. Our few-shot prompt includes examples that use built-in numeric operations, the `math` module, and SymPy.
- **GSM8k+Python**, solving a GSM8k word problem by writing a Python program that executes to the answer. We use the prompt from Gao et al. (2023).

		GSM8k+Python pass@1	**MATH+Python** pass@1
Code Llama	7B	27.1%	17.2%
LLEMMA	7B	40.1%	21.5%
Code Llama	34B	52.7%	23.5%
LLEMMA	34B	62.6%	27.1%

Table 7: Mathematical problem solving with tool use.

Results. As seen in Table 7, LLEMMA improves over Code Llama on both tasks. Its performance on MATH and GSM8k with tools is also higher than its performance on these datasets without tools.

H Dataset overlap and memorization

Do test problems or solutions appear in the corpus? We check whether any 30-gram in a test sequence (either an input problem or an output solution) occurs in any OpenWebMath or AlgebraicStack document. If so, we say that a *hit* occurred between the sequence and the document. Table 8 shows hits between sequences from MATH and documents from Proof-Pile-2. Using our methodology, around 7% of MATH test problem statements and 0.6% of MATH test solutions have hits. Note that our methodology gives a lower bound on the number of semantically equivalent sequences (e.g., it does not account for alternative phrasing).

We manually inspected 100 uniformly sampled hits between a test problem statement and an OpenWebMath document. 41 of the cases had no solution, which included websites with a list of problems, discussions, or hints. 49 had an alternative solution to the MATH ground-truth solution, but with the same answer. These include solutions that solve the problem differently than the ground-truth, solutions with missing details, and discussions that include the answer. 9 cases had a missing or incorrect answer, and 1 had the same solution as in the ground-truth. In summary, we find that

solutions can appear in a corpus derived from web documents, particularly alternative solutions to those in the evaluation set. We repeated our analysis with 20-gram hits and our findings were similar, though with false positives; see Appendix Figure 5 for examples.

Proof-Pile-2	Test	Problem Example	Problem Docs	Solution Example	Solution Docs
OpenWebMath	MATH	348	664	34	43
AlgebraicStack	MATH	3	3	0	0
OpenWebMath	GSM8k	2	3	0	0
AlgebraicStack	GSM8k	0	0	0	0

Same solution	1
Different solution, same answer	49
Different solution, different answer	9
No solution	41
Different problem	0

Table 8: *Left:* 30-gram hits between MATH test problems or solutions and Proof-Pile-2 documents. *Example* and *Docs* are the numbers of unique test examples and Proof-Pile-2 documents with a hit. *Right:* manual inspection of 100 hits between a problem statement and a Proof-Pile-2 document.

How do problems in the corpus impact performance? Next, we evaluate LLEMMA-34b on the test examples with a 30-gram hit, and the test examples without a 30-gram hit. Table 9 shows the accuracy partitioned by MATH difficulty level. The model's accuracy remains low on difficult problems (e.g., 6.08% on Level 5 problems with a hit, versus 6.39% on problems without a hit), and we observe no clear relationship between 30-gram hits and accuracy across difficulty levels. We conclude that a nontrivial match between a test example and a training document did not imply that the model generated a memorized correct answer. We repeated the analysis with 20-gram hits, and with the 7b model, and our findings were analogous. Figure 6 shows an example.

MATH Level	Hit Accuracy	Nonhit Accuracy	# Hits
Level 1	72.73	61.50	11
Level 2	35.71	40.18	28
Level 3	30.36	26.88	56
Level 4	14.89	16.61	94
Level 5	6.08	6.39	181

Table 9: LLEMMA-34b's accuracy on hits (a 30-gram overlap between a problem or solution and a training sequence) and non-hits by MATH difficulty level.

Finally, we check 30-gram hits between LLEMMA's MATH generations and OpenWebMath. There were 13 hits, which occurred when the model generated a common sequence of numbers (e.g., a list of Fibonacci numbers), plus one instance of factoring a polynomial. Appendix Figure 5 shows an example. We find all of these observations worthy of further study. Using LLEMMA and Proof-Pile-2 to better understand data, memorization, and performance is an interesting future direction.

H.1 Impact of data mixture

When training a language model, it is common to upsample high-quality subsets of the training data according to mixture weights (Brown et al., 2020; Gao et al., 2020; Biderman et al., 2023; Xie et al., 2023). We select mixture weights by doing short training runs on several hand-picked mixture weights, then choosing the one which minimizes perplexity on a set of high-quality held-out text (we use the MATH training set). Table 10 shows the MATH training set perplexity of models trained using different mixtures of arXiv to web to code. Based on these results, we trained LLEMMA with a ratio of $2:4:1$. Note that our methodology uses the MATH training set to determine a training hyperparameter, though we expect that the effect is similar to that of related high-quality texts.

Mixture	MATH training set perplexity							
	Overall	Prealgebra	Algebra	Number Theory	Counting & Probability	Geometry	Intermediate Algebra	Precalculus
2:4:1	**1.4782**	**1.4945**	**1.5150**	**1.5523**	**1.4745**	**1.5187**	**1.4393**	**1.3312**
2:4:2	1.4821	1.4999	1.5190	1.5558	1.4772	1.5235	1.4426	1.3337
4:2:1	1.4871	1.5047	1.5240	1.5605	1.4811	1.5340	1.4466	1.3377
4:2:2	1.4893	1.5077	1.5271	1.5617	1.4825	1.5380	1.4474	1.3385
4:4:1	1.4868	1.5057	1.5253	1.5607	1.4821	1.5293	1.4456	1.3346
4:4:2	1.4850	1.5030	1.5228	1.5589	1.4795	1.5290	1.4443	1.3341

Table 10: MATH training set perplexity of models trained using different data mixtures. Each mixture is represented by its arXiv:Web:Code ratio.

Method	Autoformalization pass@1	
	miniF2F-valid*	miniF2F-test
Sledgehammer	14.72%	20.49%
Code Llama 7b	16.31%	17.62%
LLEMMA-7b	20.60%	22.13%
Code Llama 7b ∪ Sledgehammer	20.17%	25.00%
LLEMMA-7b ∪ Sledgehammer	25.97%	27.46%

Table 11: **Isabelle autoformalization**. *We exclude the 11 examples used in the few-shot prompts. Pass@1 with greedy decoding.

I Additional Results

I.1 Proof autoformalization

Table 11 shows additional results on Isabelle proof autoformalization, including the union of theorems closed by Sledgehammer and the given language model.

J Qualitative Examples

J.1 Dataset overlap

Figure 5 shows example false positives when checking n-gram overlap with OpenWebMath documents for various n.

Figure 6 shows an example OpenWebMath document that has 30-gram overlap with a MATH problem, and LLEMMA-7b's generated solution.

K Supervised Finetuning

A full exploration of finetuning applications for LLEMMA, such as instruction following (Ouyang et al., 2022; Wei et al., 2022), dialogue modeling (Thoppilan et al., 2022; Touvron et al., 2023; Collins et al., 2023), and reward modeling (Cobbe et al., 2021; Lightman et al., 2023) are outside the scope of this work. However, to establish that LLEMMA retains its advantage over other open models when finetuned, we conduct preliminary experiments finetuning LLEMMA-7B on MetaMathQA (Yu et al., 2023), a supervised dataset targeted at the MATH and GSM8k benchmarks. Results are shown in Table 12.

Initialization	Finetune Dataset	MATH	GSM8k
Llama 2 7B	WizardMath (Proprietary)	10.7%	54.9%
Llama 2 7B	MetaMathQA	19.4%	66.4%
LLEMMA 7B	MetaMathQA	**25.2%**	**66.5%**
Llama 2 70B	WizardMath (Proprietary)	22.7%	81.6%
Llama 2 70B	MetaMathQA	**26.6%**	**82.3%**

Table 12: Finetuning of various 7B base models on supervised mathematics datasets. All results with a Llama 2 initialization are copied from the literature (Luo et al., 2023; Yu et al., 2023). The LLEMMA 7B finetune is trained with identical hyperparameters to the models in Yu et al. (2023)

.

OpenWebMath document

```
2D affine transformations can be better represented using 2 by 2 matrices, since they
are simply linear combinations of 2 variables. The advantage of this is that the matrices
are associative under multiplication Also, GPUs and modern toolkits are optimised to work
with this representation. As a result, a scale matrix is \begin{bmatrix} s_x & 0 \\ 0 &
s_y \end{bmatrix}, and a rotation matrix is \begin{bmatrix} \cos \theta & -\sin \theta \\
\sin \theta & \cos \theta \end{bmatrix}.

A translation matrix is simply \begin{bmatrix} 1 & \frac{t_x}{y} \\ \frac{t_y}{x} & 1 ...
```

MATH problem

A rotation centered at the origin takes $\begin{pmatrix} 13 \\ 0 \end{pmatrix}$ to $\begin{pmatrix} 5 \\ -12 \end{pmatrix}$. Which vector does the rotation take $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ to?

MATH solution

The rotation matrix must be of the form $\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$. Thus,...

Hit

```
\cos \theta & -\sin \theta \\ \sin \theta & \cos
```

OpenWebMath document

```
# Basic Probability

A number is selected at random from 1 through 100, inclusive. What is the probability
that the number is a divisor of 50? Express your answer as a common fraction.

Apr 24, 2019

There are a 100 integers between 1-100, inclusive. Since 50 is $$2*5^2$$, it has
$$(1+1)(1+2)=(2)(3)=6$$ factors. Thus, the answer is
$$\frac{6}{100}=\boxed{\frac{3}{50}}.$$
```

MATH problem

A number is selected at random from 1 through 100, inclusive. What is the probability that the number is a perfect square?

Hit

A number is selected at random from 1 through 100, inclusive. What is the probability that the number is a

OpenWebMath document

```
Fig. 2.

Use values of the most used medicinal plants in the Safi Province (Morocco).

It is also important to note that for the abovementioned medicinal plants, many other
folk uses have been reported in different regions of Morocco. Furthermore,
literature-based proof revealed that these species have proven a wide variety of
biological and pharmacological activities (Table 4, Ref. [14, 17, 19, 20, 21, 23,
24, 26, 28, 30, 31, 34, 35, 36, 38, 39, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75,
76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96,
97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113,
114, 115, 116]), which may confirm the different popular applications of ...
```

Generated solution (LLEMMA 7b):

The first 100 positive integers are 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20, 21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49, 50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78, 79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100. We can see that...

Figure 5: **Data overlap:** Example false positives using 10-gram match between MATH solutions and OpenWebMath documents (*top*), 20-gram match between MATH problems and OpenWebMath documents (*middle*), and 30-gram match between LLEMMA-7b's generated solutions and OpenWebMath documents (*bottom*).

OpenWebMath document

```
A triangle is formed with edges along the line $y=\frac{2}{3}x+5$,
the $x$-axis, and the line $x=k$. If the area of the triangle is less than $20$,
find the sum of all possible integral values of $k$.

Feb 28, 2018

Look at the graph, here...two triangles are possible :
https://www.desmos.com/calculator/m6wnjpgldq
The  height of the triangles at any point will be formed by

[ (2/3)x + 5 ]

And the bases will be  [ x - (- 7.5)]   = [ x + 7.5]
So....we want to solve this
(1/2) [ (2/3)x + 5 ] [ x + 7.5 ]  =  20
[ (2/3)x + 5 ] [ x + 7.5]  =  40
(2/3)x^2 + 5x + 5x  + 37.5  = 0
(2/3)x^2 + 10x - 2.5  =  0

Using a little technology.....the max x  value for the
triangle formed above the x axis will be = .246

And the min x value for the triangle formed below the x axis will be = -15.246

With the given boundaries, the  integer sums  of all possible x values of k giving
triangles with an area < 20 units^2  =

[ (-15) + (-14 ) + (-13) + ...+ ( -2) + ( - 1 ) + 0 ]  =
-  (15) (16) / 2   =
-120

Feb 28, 2018
```

MATH Problem: A triangle is formed with edges along the line $y = \frac{2}{3}x + 5$, the x-axis, and the line $x = k$. If the area of the triangle is less than 20, find the sum of all possible integral values of k.
Answer: -120

Generated solution (LLEMMA 7b):
The triangle has vertices $(k, 0)$, $(0, 5)$, and $(k, \frac{2}{3}k + 5)$. The base of the triangle has length k, and the height of the triangle has length $\frac{2}{3}k + 5$. The area of the triangle is therefore

$$A = \frac{1}{2}bh = \frac{1}{2}k(\frac{2}{3}k + 5) = \frac{1}{3}k^2 + \frac{5}{2}k.$$

Since the area of the triangle is less than 20, we have

$$\frac{1}{3}k^2 + \frac{5}{2}k < 20 \Rightarrow \frac{1}{6}k^2 + \frac{5}{4}k - 10 < 0.$$

This inequality is true if and only if $\frac{1}{6}k^2 + \frac{5}{4}k - 10$ is positive. Since $\frac{1}{6}k^2 + \frac{5}{4}k - 10$ is a quadratic with positive leading coefficient, it is positive if and only if its vertex is less than 0. The vertex of $\frac{1}{6}k^2 + \frac{5}{4}k - 10$ is $(-\frac{10}{3}, -\frac{125}{18})$. Since $-\frac{10}{3} < 0$, we have that $\frac{1}{6}k^2 + \frac{5}{4}k - 10 > 0$ if and only if $k < -\frac{10}{3}$. Therefore, the only possible integral values of k are $k = 0, 1, 2, 3, 4$. The sum of these values is $\boxed{10}$. Final Answer: The final answer is 10.

Figure 6: **Data overlap:** Example OpenWebMath document that has a 30-gram overlap with the given MATH problem, and LLEMMA-7b's generated solution.