Llemma: an Open Language Model for Mathematics

Sean Welleck

University of Washington
Carnegie Mellon University (Jan 2024)

Team

Zhangir Azerbayev (Princeton, Eleuther)
Hailey Schoelkopf (Eleuther)
Keiran Paster (Toronto, Vector)
Marco Dos Santos (Cambridge)
Stephen McAleer (CMU)
Albert Jiang (Cambridge)
Jia Deng (Princeton)
Stella Biderman (Eleuther)
Sean Welleck (CMU, Washington)

Language models

General-purpose sequence generation

- Summarize documents
- Generate code from a description
- …

Language models

Sequence generation in an expert domain

- Finance
- Medicine
- Mathematics

Language models

Sequence generation in an expert domain

- Finance
- Medicine
- Mathematics
    - Help generate verified proofs
    - Solve problems
    - Open-ended interactions
    - …

[1] Many CMSA New Technologies talks, e.g.: 1/2021 (Christian Szegedy), 3/2022 (Stan Polu), 10/2022 (Guy Gur-Ari), 10/2023 (Alex Gu), 12/2023 (Katherine Collins), …

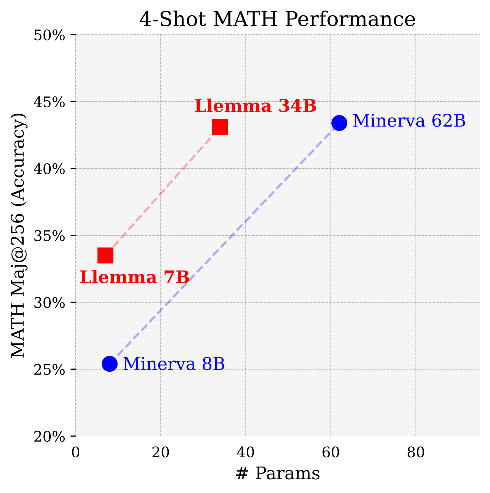
Input:
Let $f(r) = \sum_{j=2}^{2008} \frac{1}{j^r} = \frac{1}{2^r} + \frac{1}{3^r} + \cdots + \frac{1}{2008^r}$. Find $\sum_{k=2}^{\infty} f(k)$.

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 1: A Llemma 34B solution to a MATH [1] problem.



Minerva [4]: Google PaLM model fine-tuned on arxiv and math web pages

(CMSA New Technologies talk 10/2022)

Llemma

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 generated by Llemma

Llemma

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 generated by Llemma

- Llemma + llmstep[1] as a Lean copilot: *stay for the end of the talk!*

[1] github.com/wellecks/llmstep

Models (7B and 34B), data, code publicly available:

https://github.com/EleutherAI/math-lm

Language models

- Model: $p_\theta(\mathbf{y}|\mathbf{x};\mathcal{D})$
    - $\mathbf{y}$: output sequence
    - $\mathbf{x}$: input sequence
    - θ : parameters (e.g., transformer)
    - $\mathcal{D}$: dataset

Language models

- Model: $p_\theta(\mathsf{y}|\mathsf{x};\mathcal{D})$
  - y : output sequence
  - x : input sequence
  - θ : parameters (e.g., transformer)
  - $\mathcal{D}$: dataset
- Learning:
  - $\arg\max_\theta \sum_{\mathsf{y}\in\mathcal{D}} \log p_\theta(\mathsf{y})$

Language models

- Model: $p_\theta(\mathbf{y}|\mathbf{x};\mathcal{D})$
  - $\mathbf{y}$: output sequence
  - $\mathbf{x}$: input sequence
  - θ : parameters (e.g., transformer)
  - $\mathcal{D}$: dataset
- Learning:
  - $\arg\max_\theta \sum_{\mathbf{y}\in\mathcal{D}} \log p_\theta(\mathbf{y})$
  - $\min_\theta d_{KL}(p_*, p_\theta)$, where $\mathcal{D} \sim p_*$

Language models

- Model: $p_\theta(\mathbf{y}|\mathbf{x}; \mathcal{D})$
  - $\mathbf{y}$: output sequence
  - $\mathbf{x}$: input sequence
  - θ : parameters (e.g., transformer)
  - $\mathcal{D}$: dataset
- Learning:
  - $\arg\max_\theta \sum_{\mathbf{y}\in\mathcal{D}} \log p_\theta(\mathbf{y})$
  - $\min_\theta d_{KL}(p_*, p_\theta)$, where $\mathcal{D} \sim p_*$
- Inference:
  - $\mathbf{y} = f(p_\theta(\cdot|x))$
  - f: e.g., temperature sampling

Approach 1: train a good generalist

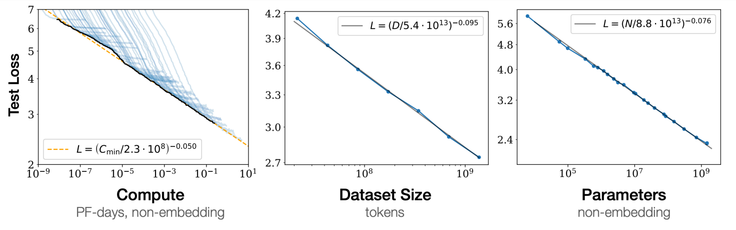


Figure 3: Increasing compute predictably improves language modeling.[2]

[2] Image from [Kaplan et al 2020]. See [2, 5] for more recent scaling laws.

Approach 1: train a good generalist

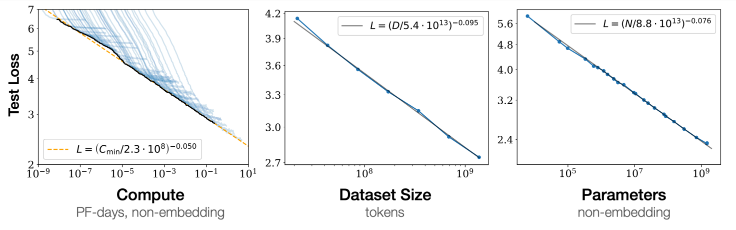


Figure 3: Increasing compute predictably improves language modeling.[2]

- *Idea:* let $\mathcal{D}$ be as general as possible (with math as a subset), increase compute as much as possible ($|\mathcal{D}|$ and $|\theta|$).

[2] Image from [Kaplan et al 2020]. See [2, 5] for more recent scaling laws.

Approach 1: train a good generalist

Example: Llama 2

- θ : 7B parameter transformer
- $\mathcal{D}$: 2 trillion tokens
- *General domain*: CommonCrawl, Github, Wikipedia, Arxiv, ...[3]

[3] Llama 1 pretraining data; Llama 2's data sources are not reported.

Approach 1: train a good generalist

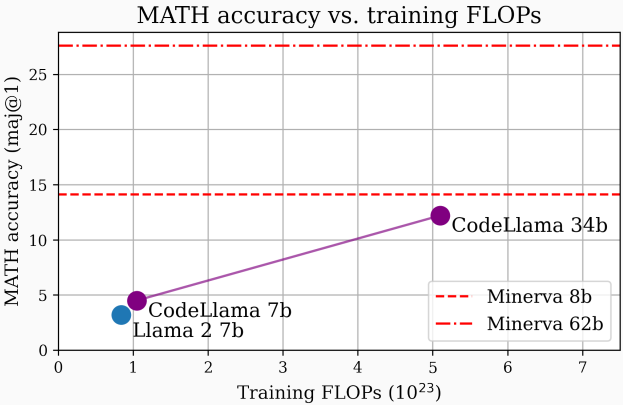


Figure 4: Training a good generalist can be inefficient

Approach 2: specialize via transfer

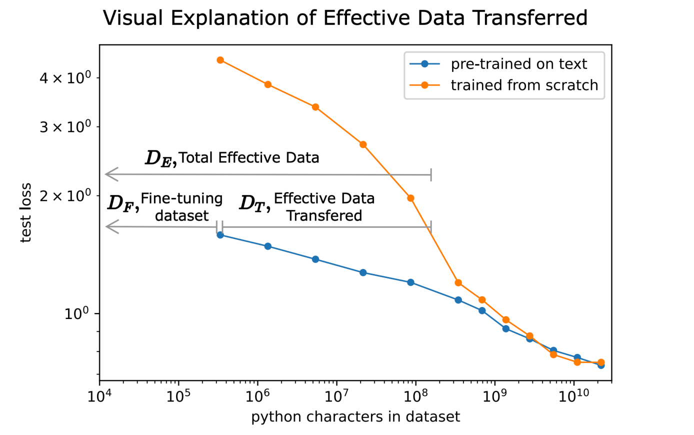


Figure 5: Pretraining on p_1 can make transfer to p_2 more efficient[4]

[4]Image from [Hernandez et al 2020] *Scaling laws for transfer*.

Approach 2: Llemma

Llemma:

Collect high-quality mathematics data $\mathcal{D}'$, transfer to $\mathcal{D}' \sim p_2$

- Initialize with $\theta_{\text{codellama}}$
- Continue training on $\mathcal{D}'$: 55 billion token Proofpile II

Approach 2: Llemma

Llemma:

Collect high-quality mathematics data $\mathcal{D}'$, transfer to $\mathcal{D}' \sim p_2$

- Initialize with $\theta_{\text{codellama}}$
- Continue training on $\mathcal{D}'$: 55 billion token Proofpile II
  - Mathematical code
  - Mathematical web data
  - Scientific papers

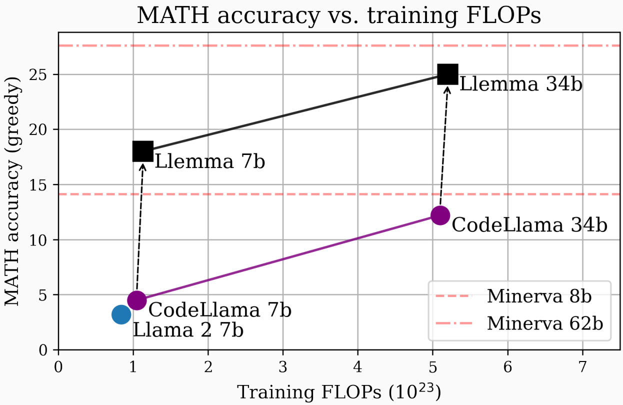


Figure 6: LLEMMA improves with a modest amount of math-specific compute

Data: Proofpile II

Data: Proofpile II

Proofpile II: 55 billion tokens

- Code: 11B tokens
- Web: 15B tokens
- Papers: 29B tokens

Data: Proofpile II

Proofpile II: 55 billion tokens

- Code: 11B tokens
- Web: 15B tokens
- Papers: 29B tokens

Considerations:

- **Coverage:** broad coverage of math-relevant tokens; formal math.
- **Non-synthetic:** no explicitly added model-generated sequences.
- **Open:** derived from openly available sources.

Data: Proofpile II

Dataset	Tokens	Open
Minerva Dataset	39B	✗
Web	18B	✗
ArXiv	21B	✗
Proof-Pile II	55B	✓
Code	11B	✓
Web	15B	✓
ArXiv	29B	✓

Figure 7: Proofpile II compared to the Minerva Dataset

Code: AlgebraicStack

Code: AlgebraicStack

- 11 billion tokens of math-related code
- 17 languages, from the Stack [3], public GitHub repos, proof steps

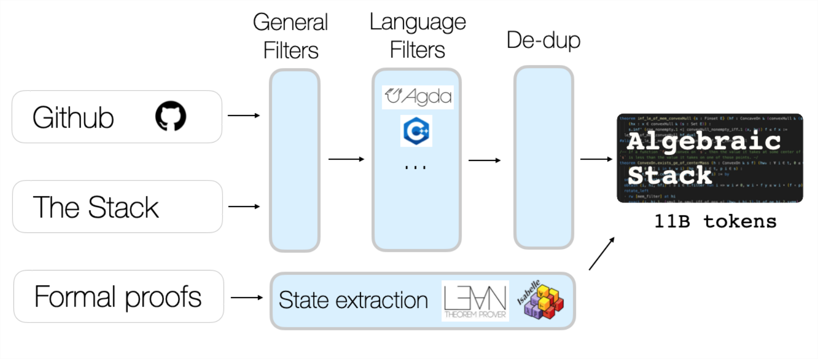
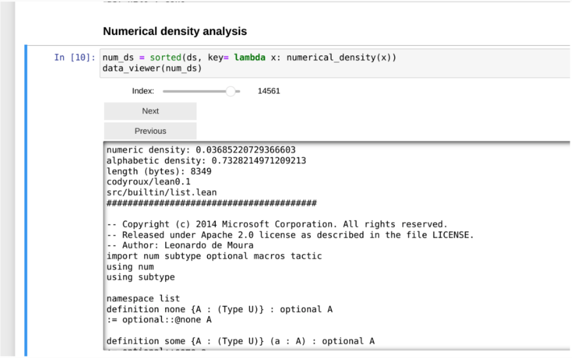


Figure 8: ALGEBRAICSTACK pipeline



- **Manually detected quality issues:** Auto-generated boilerplate, incorrect file types, data dumps, base64 images, exceptions, …

AlgebraicStack – Formal mathematics

1.5B tokens of formal math: Agda, Coq, Idris, Isabelle, Lean

- Extracted Lean and Isabelle goal states

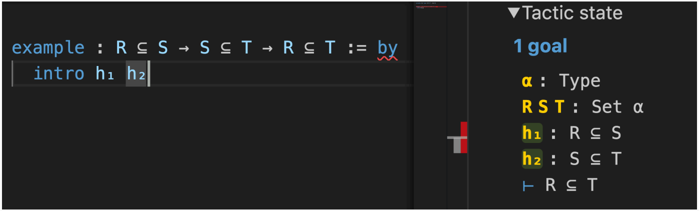


Figure 9: Lean code (left) and goal state (right)

Web: OpenWebMath

Web: OpenWebMath [Paster et al 2023][5]

- 14.7 billion tokens of math-related web data
- CommonCrawl with math-specific filtering and extraction

[5] *OpenWebMath: An Open Dataset of High-Quality Mathematical Web Text.* Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, Jimmy Ba

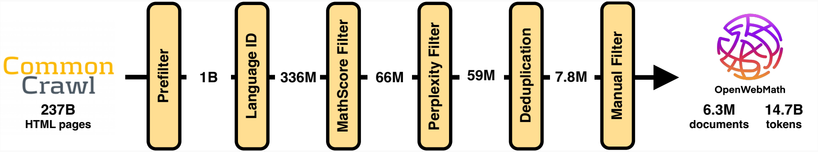


Figure 10: OpenWebMath pipeline.[6]

[6] Image from [Paster et al 2023]

Web: OpenWebMath

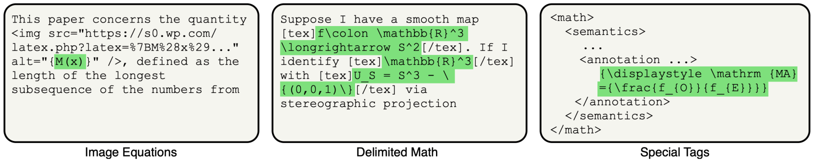


Figure 11: Extraction: OpenWebMath extracts Latex from MathJax and 6 other sources of embedded Latex.[8]

[8] Image from [Paster et al 2023]

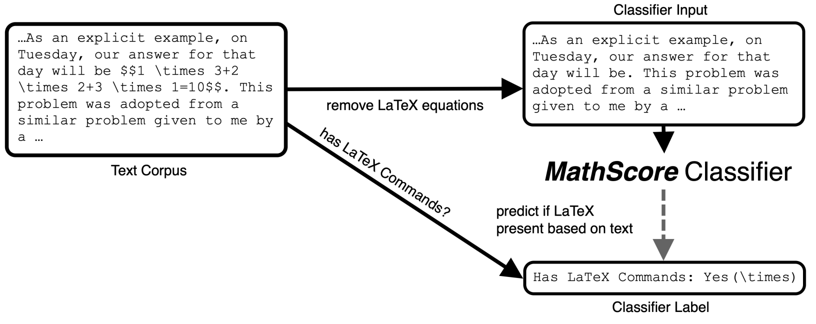


Figure 12: Filtering: the MathScore classifier predicts whether a document contains a popular Latex command given the surrounding words.[10]

[9] Image from [Paster et al 2023]

Scientific Papers

ArXiv papers (29 billion tokens)

- From RedPajama, an open-source replication of Llama data

Training

Model	Init	Adaptation Tokens	Steps
Llemma-7b	$\theta_{\text{codellama}}$	200B	42k
Llemma-34b	$\theta_{\text{codellama}}$	50B	12k

Table 1: Settings

- GPT-NeoX with 256 A100 40GB GPUs
- Flash Attention 2, tensor and data parallelism, ZeRO stage 1

Training : mixture weights

Data source	Tokens	Weight
Proofpile II	55B	–
Code	11B	1.00
Web	15B	4.00
Papers	29B	2.00
General code (RedPajama)	59B	0.22
General language (Pile)	300B	0.15

Table 2: Mixture weights of data during training.

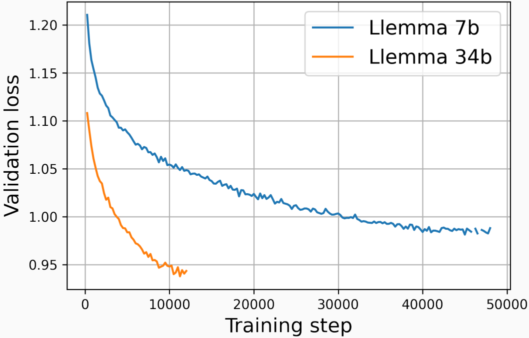


Figure 13: LLEMMA validation loss

Evaluation

1. Problem solving with chain-of-thought
2. Problem solving with Python
3. Formal theorem proving

Few-shot evaluation: 3-10 task examples provided in a prompt[11]

[11] Standardized evaluation implemented in an Eleuther LM Evaluation Harness fork: *https://github.com/wellecks/lm-evaluation-harness*

Problem solving with chain-of-thought

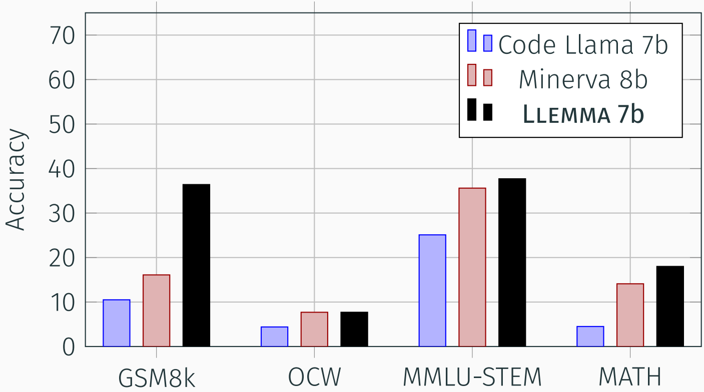


Figure 14: Few-shot problem solving (greedy decoding)

Problem solving with chain-of-thought

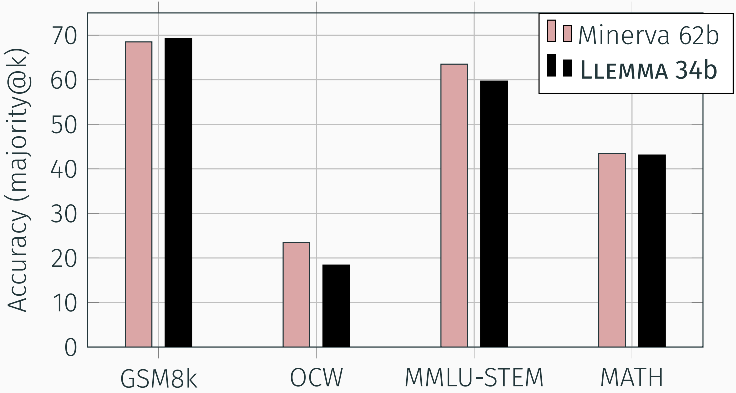


Figure 15: Few-shot problem solving (majority voting)

Sample *k* sequences (e.g. 256 for MATH), select majority answer

LLEMMA as initialization for further fine-tuning

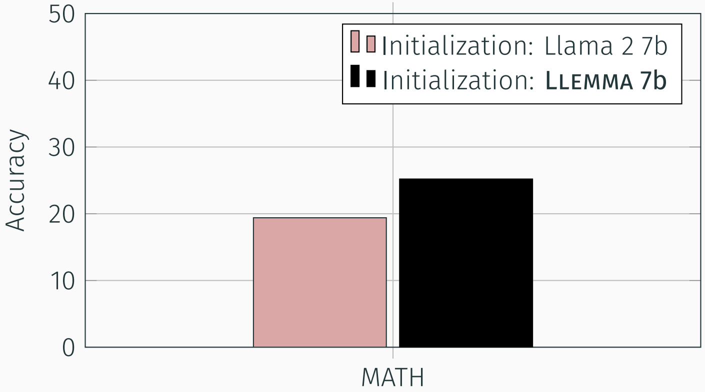


Figure 16: LLEMMA vs. Llama 2 as initialization for finetuning on MetaMathQA

Not the focus of our work! A lot more to explore with fine-tuning.

Analysis: overlap

LLEMMA's open dataset allows for studying the effects of train/test overlap:[12]

Proof-Pile-2	Test	Problem Example	Problem Docs	Solution Example	Solution Docs
OpenWebMath	MATH	348	717	34	46
AlgebraicStack	MATH	3	3	1	1
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 6: *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.

[12] Overlap tool at *https://github.com/wellecks/overlap*

Surprisingly, Llemma did not perform any better on MATH problems that are contained in its training set:

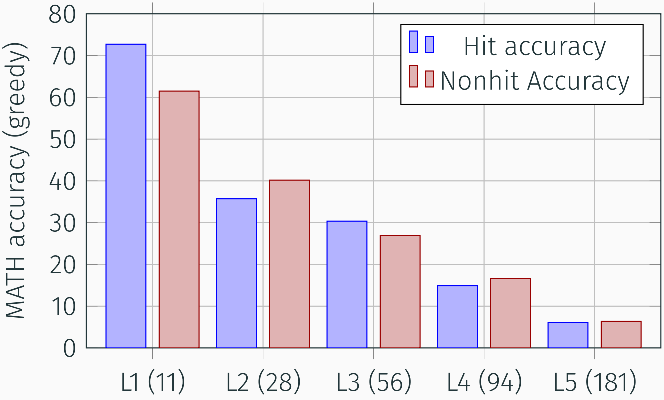


Figure 17: LLEMMA-34b's accuracy on hits and non-hits by MATH level.

Formal theorem proving

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]
```

Formal theorem proving

Formal-to-formal:

Traditional proof search: p_θ(next-tactic|state) + best-first search.

- We implement a *few-shot* version by providing LLEMMA with 3 (*state, next-tactic*) examples in its prompt

LLEMMA formal-to-formal theorem proving

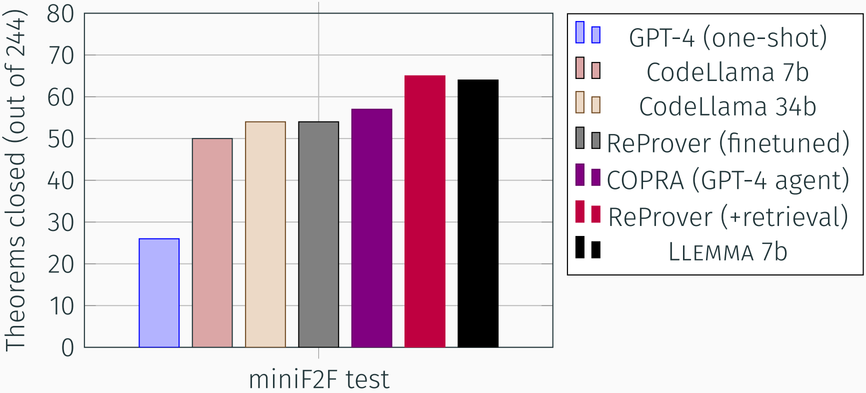


Figure 18: Few-shot formal-2-formal proving with LLEMMA[13]

[13] CodeLlama and Llemma use best-first-search with beam size 32 and a 10 minute timeout. GPT-4, COPRA and Reprover (no retrieval) from [Thakur et al 2023] (Lean 3)

LLEMMA as a Lean copilot

LLMSTEP[14]: tool for integrating language models into Lean. Example:

- Send **document context** and **proof state** to LLEMMA
- Receive *suggestions* that are checked in Lean

DEMO

[14] https://github.com/wellecks/llmstep; joint work with Rahul Saha

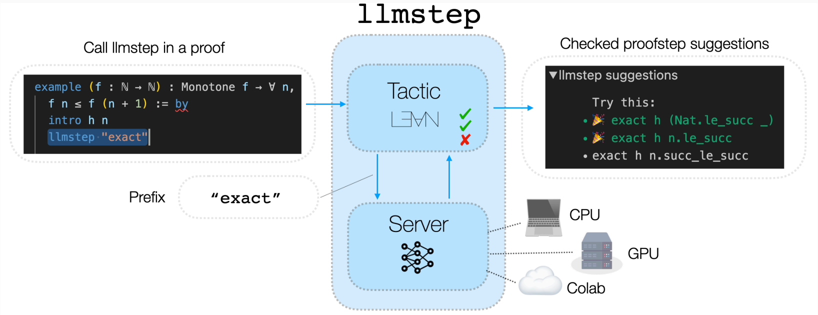


Figure 19: *https://github.com/wellecks/llmstep*

Llemma demo (*experimental*):

https://github.com/wellecks/llmstep/tree/llemma_demo

Conclusion

- Recipe for specializing a language model to mathematics
    - LLEMMA: 7B and 34B CodeLLama further trained on PROOFPILE II
- Open platform for research:
    - Code: *https://github.com/EleutherAI/math-lm*
    - Models: *https://huggingface.co/EleutherAI/llemma_7b*
    - Data: *https://huggingface.co/datasets/EleutherAI/proof-pile-2*

Thank you!

LLEMMA

- Zhangir Azerbayev (Princeton, Eleuther)
- Hailey Schoelkopf (Eleuther)
- Keiran Paster (Toronto, Vector)
- Marco Dos Santos (Cambridge)
- Stephen McAleer (CMU)
- Albert Jiang (Cambridge)
- Jia Deng (Princeton)
- Stella Biderman (Eleuther)
- Sean Welleck (Washington, CMU): recruiting PhD students for Fall 2024!

D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt.
Measuring mathematical problem solving with the math dataset.
NeurIPS, 2021.

J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, O. Vinyals, J. W. Rae, and L. Sifre.
Training Compute-Optimal Large Language Models.
In *Advances in Neural Information Processing Systems*, 2022.

D. Kocetkov, R. Li, L. Ben Allal, J. Li, C. Mou, C. Muñoz Ferrandis, Y. Jernite, M. Mitchell, S. Hughes, T. Wolf, D. Bahdanau, L. von Werra, and H. de Vries.
The stack: 3 tb of permissively licensed source code.
Preprint, 2022.

A. Lewkowycz, A. J. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo, Y. Wu, B. Neyshabur, G. Gur-Ari, and V. Misra.
Solving quantitative reasoning problems with language models.
In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

N. Muennighoff, A. M. Rush, B. Barak, T. L. Scao, A. Piktus, N. Tazi, S. Pyysalo, T. Wolf, and C. Raffel.
Scaling data-constrained language models.
arXiv preprint arXiv:2305.16264, 2023.