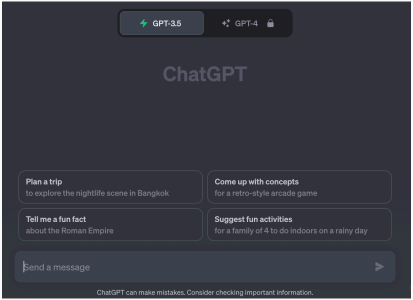
Language models and formal mathematics

Sean Welleck

University of Washington
Carnegie Mellon University (Jan 2024)

General sequence generation

- Summarize documents
- Generate code from a description
- ...



Language models

Sequence generation in mathematics

- Solve problems
- Open-ended dialogue
- Help generate verified proofs
- ...

[1]See, e.g. Harvard New Technologies in Mathematics: 1/2021 (Christian Szegedy), 3/2022 (Stan Polu), 10/2022 (Guy Gur-Ari), 10/2023 (Alex Gu), 12/2023 (Katherine Collins), ...;

Input:
Let r be a root of $x^2 + 5x + 7 = 0$. Compute $(r-1)(r+2)(r+6)(r+3)$.

Llemma 34B solution:
We have

$$
\begin{aligned}
(r-1)(r+2)(r+6)(r+3) &= (r^2 + r - 2)(r^2 + 9r + 18) \\
&= r^4 + 10r^3 + 25r^2 + 19r - 36 \\
&= r^4 + 5r^3 + 7r^2 + 5r + 7 + 5r^3 + 20r^2 + 14r - 36 \\
&= r^4 + 5r^3 + 25r^2 + 19r - 29 \\
&= (r^2 + 5r + 7)(r^2 + 4r - 5) \\
&= (0)(r^2 + 4r - 5) \\
&= \boxed{0}.
\end{aligned}
$$

Final Answer: The final answer is 0.

Figure 1: A Llemma 34B solution to a MATH [4] problem.

"Informal" mathematics

Input:
Let r be a root of $x^2 + 5x + 7 = 0$. Compute $(r-1)(r+2)(r+6)(r+3)$.

Llemma 34B solution:
We have

$$\begin{aligned}(r-1)(r+2)(r+6)(r+3) &= (r^2+r-2)(r^2+9r+18)\\ &= r^4+10r^3+25r^2+19r-36\\ &= r^4+5r^3+7r^2+5r+7+5r^3+20r^2+14r-36\\ &= r^4+5r^3+25r^2+19r-29\\ &= (r^2+5r+7)(r^2+4r-5)\\ &= (0)(r^2+4r-5)\\ &= \boxed{0}.\end{aligned}$$

Final Answer: The final answer is 0.

Figure 2: No correctness guarantees, errors can be difficult to detect.

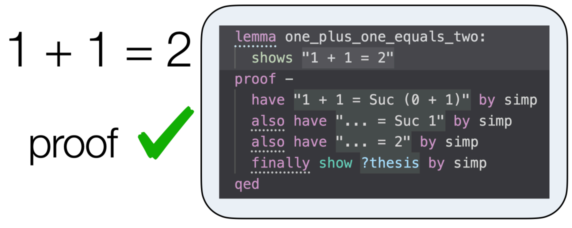


Figure 3: Mathematics as verifiable source code

Formal mathematics (Demo)

If $R \subseteq S$ and $S \subseteq T$ then $R \subseteq T$

LEAN THEOREM PROVER

Formal mathematics

Lean Mathlib

- 1+ million lines of code
- > 300 contributors
- Algebra, Linear Algebra, Topology, Analysis, Probability, Geometry, Combinatorics, . . .

LEAN THEOREM PROVER

Formal mathematics

- **Liquid tensor project**: Lean formalization with Peter Scholze[1]
- **Courses** at CMU, Imperial College London, Fordham, JHU, …[2]
- **eXperimental Lean Lab** at the University of Washington![3]

---

[1] *https://www.nature.com/articles/d41586-021-01627-2*
[2] *https://leanprover-community.github.io/teaching/courses.html*
[3] *https://sites.math.washington.edu/~jarod/xll.html*

Formal mathematics

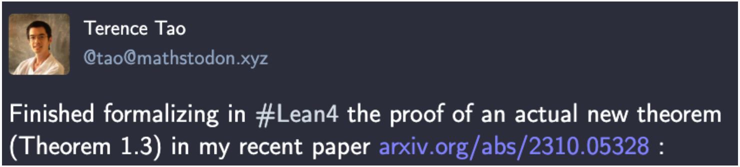


Figure 4: Terence Tao's Lean formalization project (October 2023)

Language models and formal mathematics

Generative Language Modeling for Automated Theorem Proving

Stanislas Polu
OpenAI
spolu@openai.com

Ilya Sutskever
OpenAI
ilyasu@openai.com

Figure 5: *gpt-f* (2020)

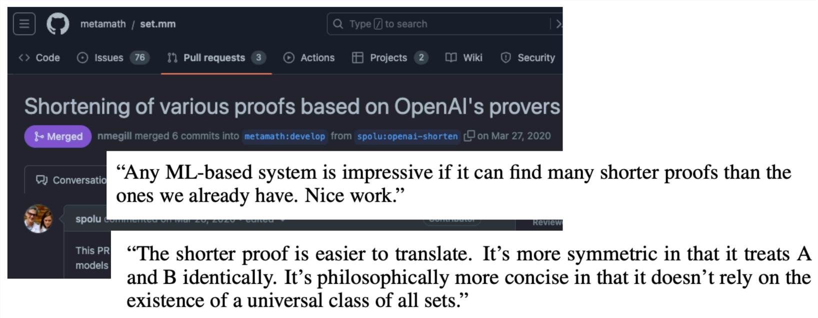


Figure 6: *gpt-f* (2020)

Language models and formal mathematics

Terence Tao
@tao@mathstodon.xyz

Finished formalizing in #Lean4 the proof of an actual new theorem (Theorem 1.3) in my recent paper arxiv.org/abs/2310.05328 :

The ability of Github copilot to correctly anticipate multiple lines of code for various routine verifications, and inferring the direction I want to go in from clues such as the names I am giving the theorems, continues to be uncanny.

Figure 7: Terence Tao's Lean formalization project (October 2023)

This talk: “build your own Lean copilot”

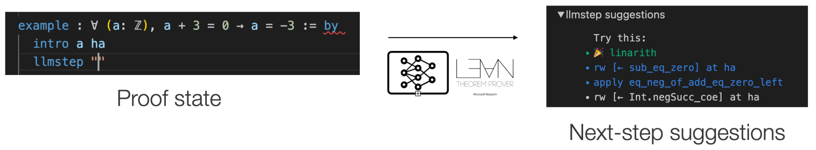
- Part 1: Small models trained to predict the next step of a proof
- Part 2: LLEMMA: foundation model for mathematics

[1] Llemma [2], LLMstep [9]

This talk: “build your own Lean copilot”

- Part 1: Small models trained to predict the next step of a proof
- Part 2: LLEMMA: foundation model for mathematics

LLMSTEP: tool for receiving verified language model suggestions



[1] Llemma [2], LLMstep [9]

Part I:
Next-step prediction

Next-step prediction

Topic	Notebook
0. Intro	notebook
1. Data	notebook
2. Learning	notebook
3. Proof Search	notebook
4. Evaluation	notebook
5. `llmsuggest`	notebook

Interactive notebooks and code:
github.com/wellecks/ntptutorial[4]

[4] From *A tutorial on neural theorem proving*, IJCAI 2023

Next-step prediction

- Language model suggests next-proof-steps
- Generate a full proof via tree search

[1] E.g., [Polu & Sutskever 2020], [Han et al 2021], [Jiang et al 2022], [Yang et al 2023]

Language models

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

Problem setup

Proof: sequence of (state, next step)

- $(x_0, y_0), \ldots, (x_t, y_t), \ldots, (x_T, y_T)$
- x_t: proof state from Lean
- y_t: proof step ("tactic")
- x_T: proof complete

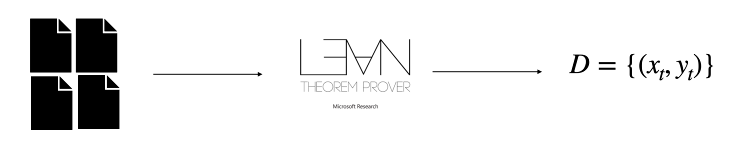
Problem setup

Proof: sequence of (state, next step)

- $(x_0, y_0), \ldots, (x_t, y_t), \ldots, (x_T, y_T)$
- x_t: proof state from Lean
- y_t: proof step ("tactic")
- x_T: proof complete

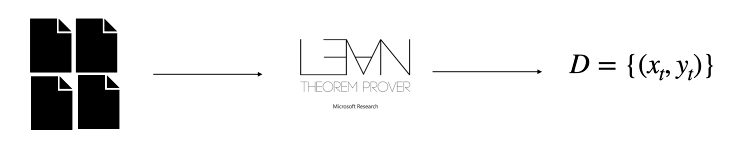
Idea:

- Collect a dataset $\mathcal{D}$ of (state, next step) examples
- Train a language model $p_\theta(y_t|x_t)$ using $\mathcal{D}$



- Extract (state, next step) pairs, e.g. from Mathlib
  - Open-source tooling available[5]

[5] E.g., Lean Dojo [10] and github.com/semorrison/lean-training-data



- Extract (state, next step) pairs, e.g. from Mathlib
  - Open-source tooling available[5]
- Mathlib yields a dataset $\mathcal{D}$ with 170,000 pairs

[5]E.g., Lean Dojo [10] and github.com/semorrison/lean-training-data

Learning

- Standard supervised learning on $\mathcal{D}$:

$$\arg\max_{\theta} \sum_{(x_t, y_t) \in \mathcal{D}} \log p_{\theta}(y_t | x_t)$$

Learning

x_t

```
Input:
[GOAL]ι : Type u_1
I✝ J✝ : Box ι
x y : ι → ℝ
I J : WithBot (Box ι)
⊢ ↑I = ↑J ↔ I = J[PROOFSTEP]
```

y_t

```
Output:
simp only [Subset.antisymm_iff, ← le_antisymm_iff, withBotCoe_subset_iff]<|endoftext|>
```

Proof search

- Use generator $p_\theta(y_t|x_t)$ to generate a full proof $y_1, \ldots, y_T$
- Standard approach: *Best-first search*

Best-first search

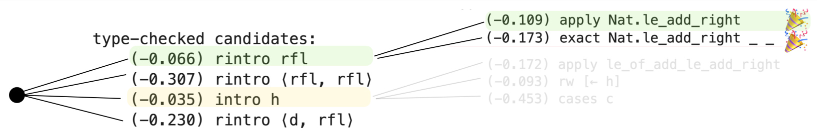
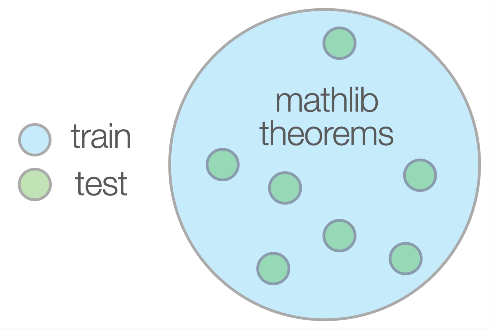


Figure 8: Best-first search[6]

[6] Example scoring function $\frac{1}{Z}\sum_t \log p_\theta(y_t|x_t)$

Evaluation

- Proof search on held-out theorems from the training distribution



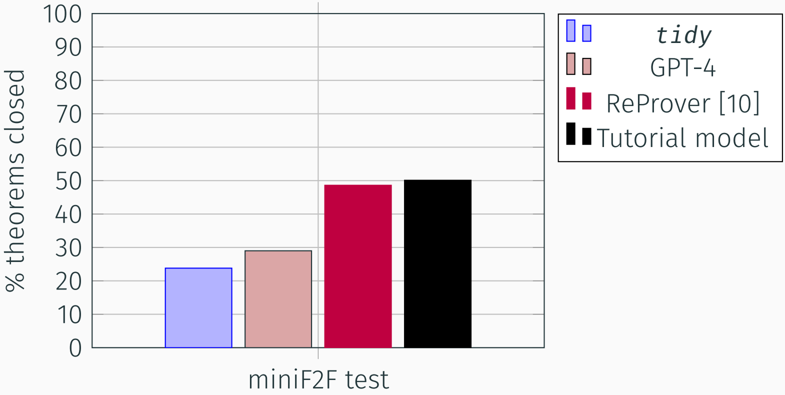


Figure 9: Proof search performance on held-out Mathlib theorems.[7]

[7] *`tidy`* and GPT-4 (Lean 3) from [10]. Tutorial model: llmstep Pythia 2.8b, best-first-search with beam size 32 and a 10 minute timeout.

Evaluation (out of domain)

Benchmarks evaluate problems drawn from a different distribution:

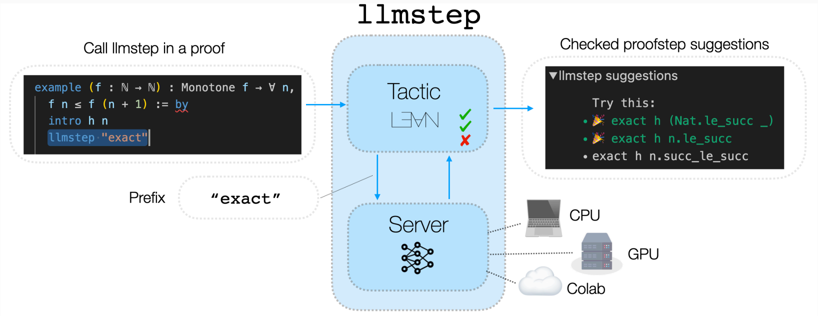
- miniF2F [11]: competition problems (AMC, AIME, IMO)
- ProofNet [1]: undergraduate textbooks (e.g. Analysis, Topology)

```
-- from mathlib:
theorem prod_mono
 {|s₁ s₂ : Subsemiring R|} (hs : s₁ ≤ s₂)
 {|t₁ t₂ : Subsemiring S|} (ht : t₁ ≤ t₂) :
 s₁.prod t₁ ≤ s₂.prod t₂ := by
    intro x hx
    simp_rw [Subsemiring.mem_prod]
    cases' x with x_fst x_snd
    exact ⟨hs hx.1, ht hx.2⟩

-- from miniF2F:
theorem mathd_algebra_159 (b : ℝ) (f : ℝ → ℝ)
 (h₀ : ∀ x, f x = 3*x^4 - 7*x^3 + 2*x^2 - b*x + 1)
 (h₁ : f 1 = 1) : b = -2 := by
    apply eq_neg_of_add_eq_zero_left
    rw [h₀] at h₁
    norm_num at h₁
    linarith
```

Figure 10: Generated proofs on mathlib and miniF2F

Verified Lean copilot: LLMSTEP



LLMSTEP: tool for receiving verified language model suggestions[8]

[8] https://arxiv.org/abs/2310.18457. github.com/wellecks/llmstep.

Verified Lean copilot: LLMSTEP

LLMSTEP: tool for receiving verified language model suggestions

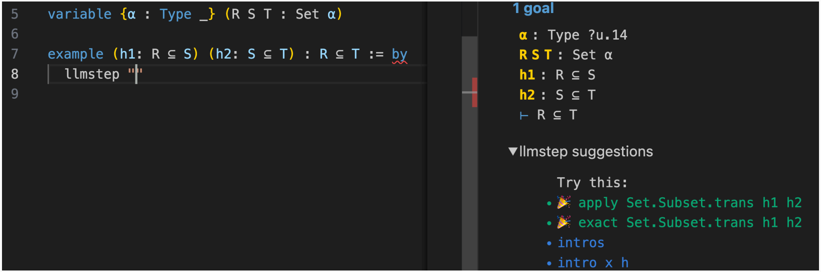


Figure 11: DEMO

Verified Lean copilot: LLMSTEP

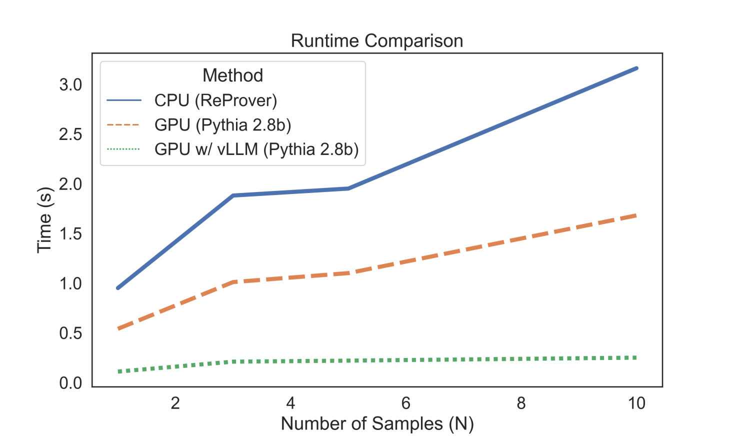


Figure 12: LLMSTEP suggestion latency on CPU and GPU

Try it at *`https://github.com/wellecks/llmstep`*

Part II:
Llemma: foundation model for mathematics

Modeling the distribution of mathematics

- *Previous*: train a model specifically for tactic prediction

Modeling the distribution of mathematics

- *Previous*: train a model specifically for tactic prediction
- *Next*: model a diverse distribution of math-related sequences

Modeling the distribution of mathematics

- *Previous*: train a model specifically for tactic prediction
- *Next*: model a diverse distribution of math-related sequences
    - Perform a task by prompting with a few (input, output) examples

Modeling the distribution of mathematics

- *Previous*: train a model specifically for tactic prediction
- *Next*: model a diverse distribution of math-related sequences
    - Perform a task by prompting with a few (input, output) examples
    - "*Foundation model*" [3]: train on large quantity of data, adapt to tasks via prompting or further training

Foundation model for mathematics

- Language model learning:

$$\arg\max_{\theta} \sum_{y \in \mathcal{D}} \log p_{\theta}(y)$$

- Equivalent to:

$$\arg\min_{\theta} \mathrm{KL}\left(p_{*}, p_{\theta}\right),$$

$$\text{where } \mathcal{D} \sim p_{*}$$

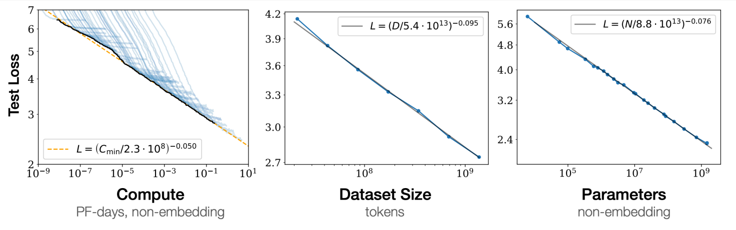


Figure 13: Increasing compute predictably improves language modeling.[9]

[9] Image from [Kaplan et al 2020]. See [5, 8] for more recent scaling laws.

Approach 1: train a good generalist

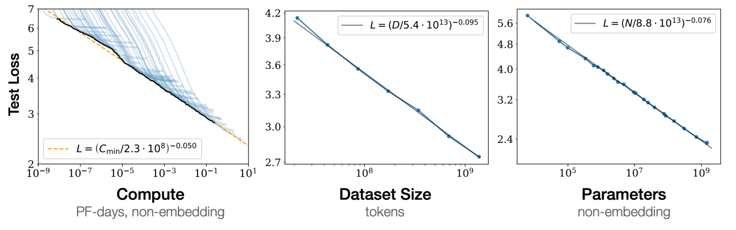


Figure 13: Increasing compute predictably improves language modeling.[9]

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

[9] Image from [Kaplan et al 2020]. See [5, 8] 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, ...

Approach 1: train a good generalist

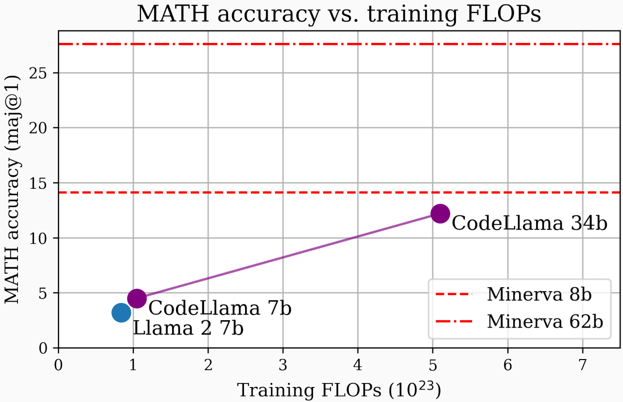


Figure 14: Training a generalist can be inefficient.[10]

[10] Minerva [7]: general language model finetuned on a large mathematical corpus

Approach 2: specialize via transfer

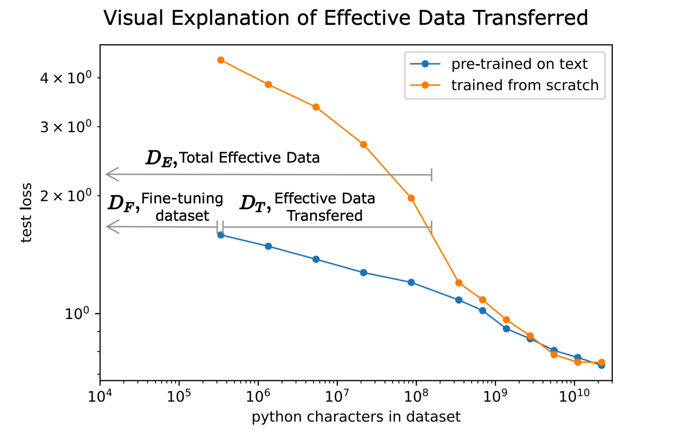


Figure 15: Pretraining on p_1 can make transfer to p_2 more efficient[11]

[11] 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

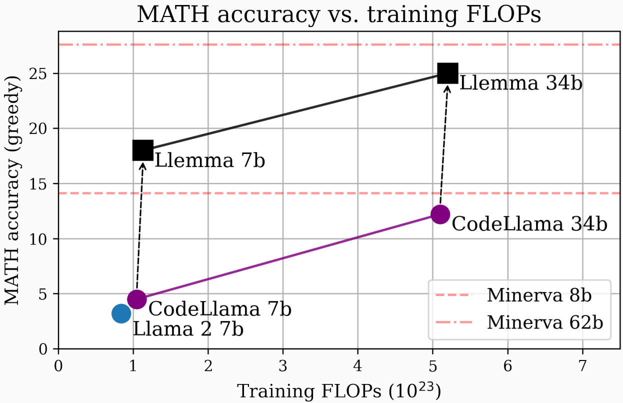


Figure 16: Llemma improves with a modest amount of math-specific compute

Data: Proofpile II

Data: Proofpile II

Proofpile II: 55 billion tokens

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

Code: AlgebraicStack

- 11 billion tokens of math-related code
- 17 languages, from the Stack [6], public GitHub repos, proof steps

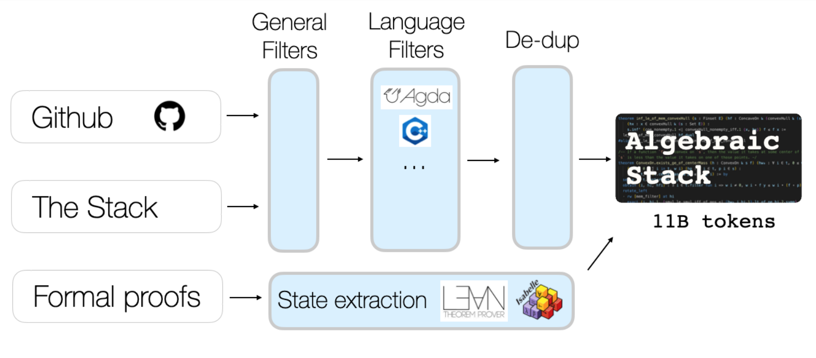


Figure 17: AlgebraicStack pipeline

AlgebraicStack – Formal mathematics

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

- Extracted Lean and Isabelle goal states

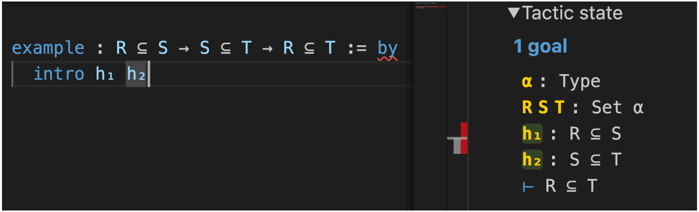


Figure 18: Lean code (left) and goal state (right)

Web: OpenWebMath [Paster et al 2023][12]

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

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

Training

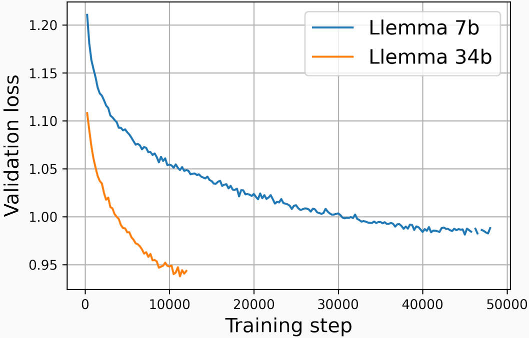


Figure 19: LLEMMA validation loss

Formal theorem proving

Traditional proof search: $p_\theta(\text{next-tactic}|\text{state})$ + 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

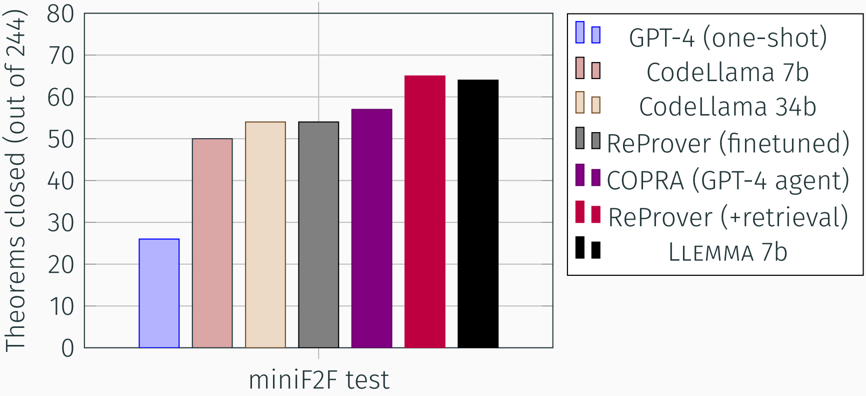


Figure 20: Few-shot proving in Lean with Llemma[13]

[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 + LLEMMA:

- *Part I:* send **proof state** to a tactic prediction model
    - $y_t \sim p_\theta(\cdot|x_t)$
- *Now:* send **file content** and **proof state** to LLEMMA
    - $y_t \sim p_\theta(\cdot|\text{preceding file content})$

Verified Lean copilot: LLMSTEP

Key difference: use new definitions and theorems

```
structure my_object (Ω : Type*)[Fintype Ω] :=
  (f : Ω → ℝ)
  (cool_property : ∀ x : Ω, 0 ≤ f x)

theorem my_object_sum_nonneg (o1 o2: my_object Ω) :
  o1.f + o2.f ≥ 0 := by
  apply add_nonneg
  · apply o1.cool_property
  · llmstep ""
```

```
▼llmstep suggestions

     Try this:
   • 🎉 exact o2.cool_property
   • 🎉 apply o2.cool_property
   • intro hb
   • intro h
```

DEMO

Verified Lean copilot: LLMSTEP

Key difference: use new definitions and theorems

```
-- Probability of any outcome is at most one.
theorem px_le_one (p : pmf Ω) (x : Ω) : p x ≤ 1 := by
  refine' hasSum_le _ (hasSum_ite_eq x (p x)) (hasSum
  intro x
  split_ifs with h
  rw [h]
  llmstep ""
```

```
▼llmstep suggestions
  Try this:
  • 🎉 exact p.my_non_neg _
  • 🎉 exact p.my_non_neg x
  • 🎉 apply p.my_non_neg
  • simp
  • apply le_of_lt
```

DEMO

Beyond formal-to-formal proving

Problem: If $3a + b + c = -3, a + 3b + c = 9, a + b + 3c = 19$, then find abc. Show that it is -56.

Informal Proof (Human-written): Summing all three equations yields that $5a + 5b + 5c = -3 + 9 + 19 = 25$. Thus, $a + b + c = 5$. Subtracting this from each of the given equations, we obtain that $2a = -8, 2b = 4, 2c = 14$. Thus, $a = -4, b = 2, c = 7$, and their product is $abc = -4 \times 2 \times 7 = -56$.

Formal Statement and Proof:

```
theorem mathd_algebra_338:
  fixes a b c :: real
  assumes "3 * a + b + c = -3" and "a + 3 * b + c = 9" and "a + b + 3 * c = 19"
  shows "a * b * c = -56"
proof -
    (* Summing all three equations yields that 5a + 5b + 5c = -3 + 9 + 19 = 25.
    Thus, a + b + c = 5. *)
    have "5 * a + 5 * b + 5 * c = -3 + 9 + 19" using assms <ATP>
    then have "5 * (a + b + c) = 25" <ATP>
    then have "a + b + c = 5" <ATP>
    (* Subtracting this from each of the given equations, we obtain that  2a =
    -8, 2b = 4, 2c = 14. Thus, a = -4, b = 2, c =7, and their product is  abc =
    -4 \times 2 \times 7 = -56. *)
    then have "2 * a = -8" "2 * b = 4" "2 * c = 14" using assms <ATP>
    then have "a = -4" "b = 2" "c = 7" <ATP>
    then show ?thesis <ATP>
qed
```

Few-shot informal-to-formal proving

Setting based on *Draft, Sketch, Prove*; Jiang*, Welleck*, Zhou* et al ICLR 2023

Part II conclusion

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

Conclusion

Neural theorem proving: language models $\cap$ formal proof assistants

- Next-step predictors: *small/fast*, narrow
- Foundation models: *larger*, flexible
- Both enable new practical tools

Conclusion

Neural theorem proving: language models $\cap$ formal proof assistants

- Next-step predictors: *small/fast*, narrow
- Foundation models: *larger*, flexible
- Both enable new practical tools

Claim: mathematical foundation models (e.g. LLEMMA) open a new frontier of methods, capabilities, and applications to explore!

Thank you!

- 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)

- Tutorial: *github.com/wellecks/ntptutorial*
- LLMstep: *github.com/wellecks/llmstep*, arXiv:2310.18457
- LLEMMA: *github.com/EleutherAI/math-lm*, arXiv:2310.10631

LLEMMA∎

Appendix

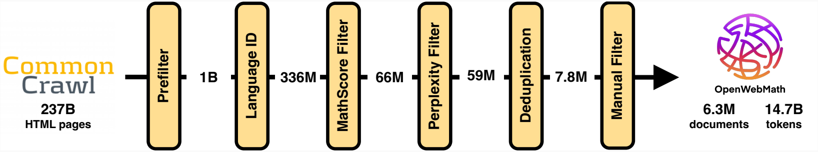


Figure 21: OpenWebMath pipeline.[14]

[14] Image from [Paster et al 2023]

Web: OpenWebMath

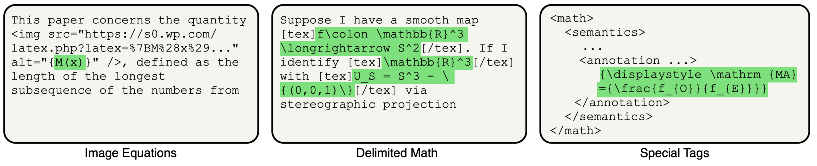


Figure 22: Extraction: OpenWebMath extracts Latex from MathJax and 6 other sources of embedded Latex.[16]

[8] Image from [Paster et al 2023]

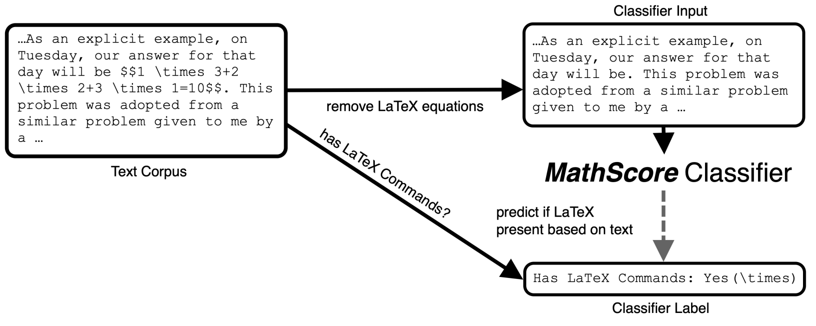


Figure 23: Filtering: the MathScore classifier predicts whether a document contains a popular Latex command given the surrounding words.[18]

[9] Image from [Paster et al 2023]

Scientific Papers

ArXiv papers (29 billion tokens)

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

Llemma as initialization for further fine-tuning

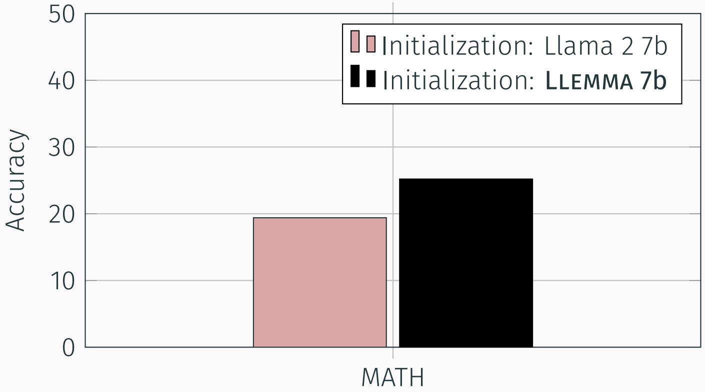


Figure 24: Llemma vs. Llama 2 as initialization for finetuning on MetaMathQA

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

LLEMMA's open dataset allows for studying the effects of train/test overlap:[19]

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.

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

Analysis: memorization

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

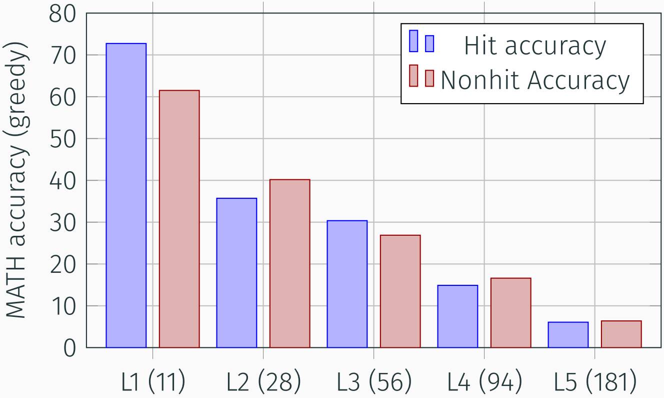


Figure 25: LLEMMA-34b's accuracy on hits and non-hits by MATH level.

References i

Z. Azerbayev, B. Piotrowski, H. Schoelkopf, E. W. Ayers, D. Radev, and J. Avigad.
Proofnet: Autoformalizing and formally proving undergraduate-level mathematics, 2023.

Z. Azerbayev, H. Schoelkopf, K. Paster, M. D. Santos, S. McAleer, A. Q. Jiang, J. Deng, S. R. Biderman, and S. Welleck.
Llemma: An open language model for mathematics.
ArXiv, abs/2310.10631, 2023.

R. B. et al.
On the opportunities and risks of foundation models, 2022.

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.

S. Welleck and R. Saha.
Llmstep: Llm proofstep suggestions in lean.
ArXiv, abs/2310.18457, 2023.

K. Yang, A. Swope, A. Gu, R. Chalamala, P. Song, S. Yu, S. Godil, R. Prenger, and A. Anandkumar.
LeanDojo: Theorem proving with retrieval-augmented language models.
In *Neural Information Processing Systems (NeurIPS)*, 2023.

K. Zheng, J. M. Han, and S. Polu.
minif2f: a cross-system benchmark for formal olympiad-level mathematics.
In *International Conference on Learning Representations*, 2022.