Scaling laws of formal reasoning

Sean Welleck

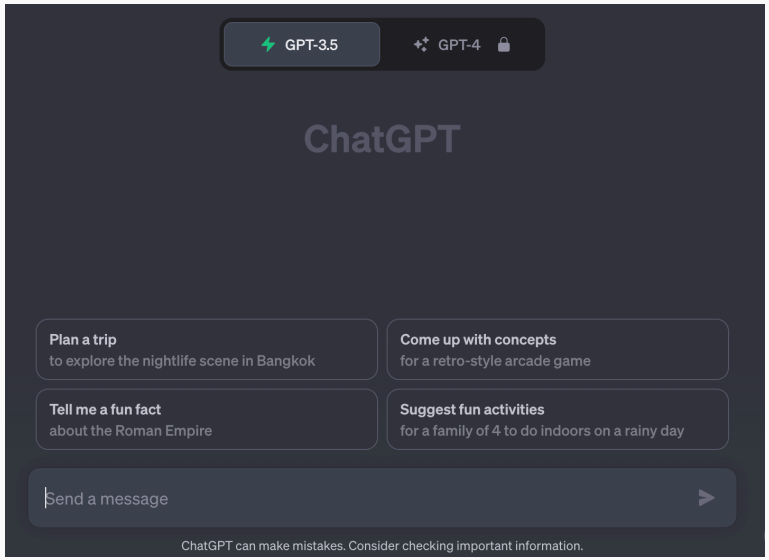
April 3, 2024

Carnegie Mellon University

Foundation models and large language models (LLMs)

Platform for sequence generation

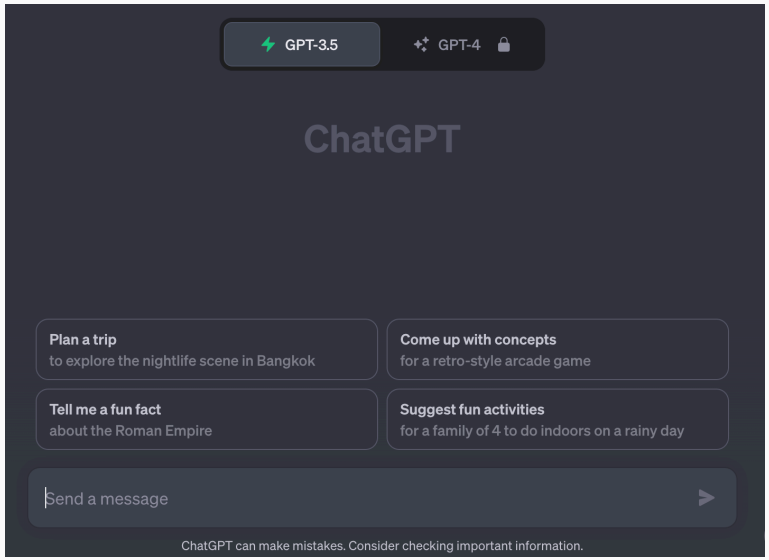
- Summarize documents
- Generate code from a screenshot
- ...



Foundation models and large language models (LLMs)

Core idea:

- Collect diverse data
- Train generative model
- Adapt to tasks (e.g., prompting)



Language models

Language models in an expert domain

- Finance
- Medicine
- Mathematics

Language models

Language models 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), ...

Problem solving

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.

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]
```

Scaling laws

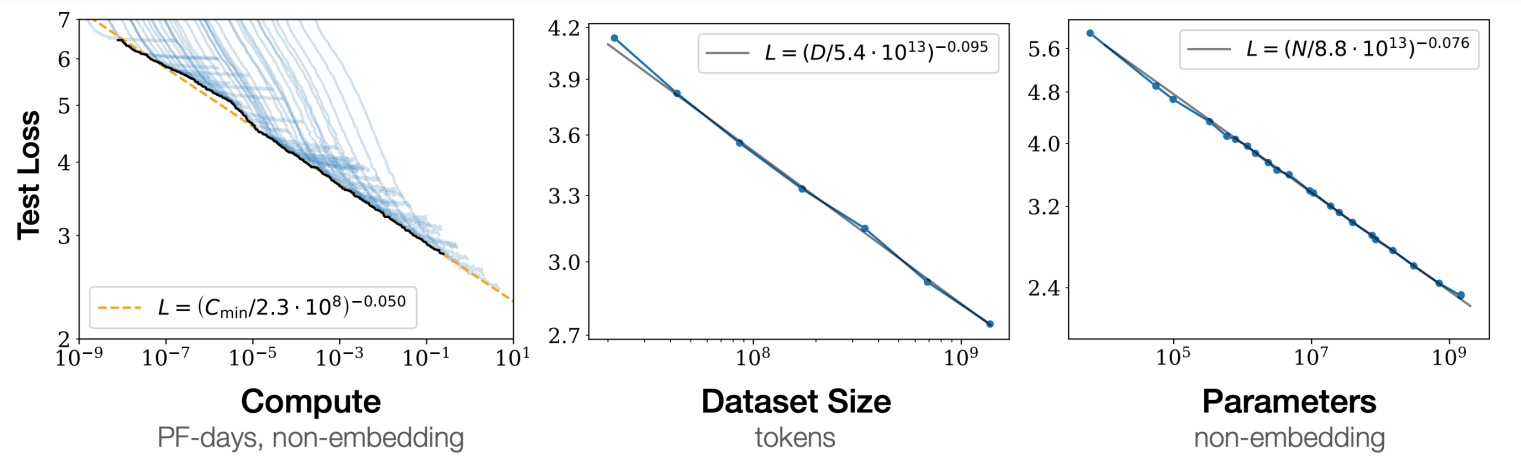


Figure 2: Increasing compute predictably improves language modeling.[1]

How do we predictably improve mathematical reasoning?

[1] Image from [Kaplan et al 2020]. See [2, 4] for more recent scaling laws.

Today's talk

1. Scaling data *quality* and *quantity*
    - $\rightarrow$ Llemma and ProofPile II
2. Scaling *inference* and *evaluation*
    - $\rightarrow$ Easy-to-hard evaluation

I. Llemma

Llemma: An Open Language Model For Mathematics

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, Sean Welleck

ICLR 2024.

Approach 1: train a good generalist

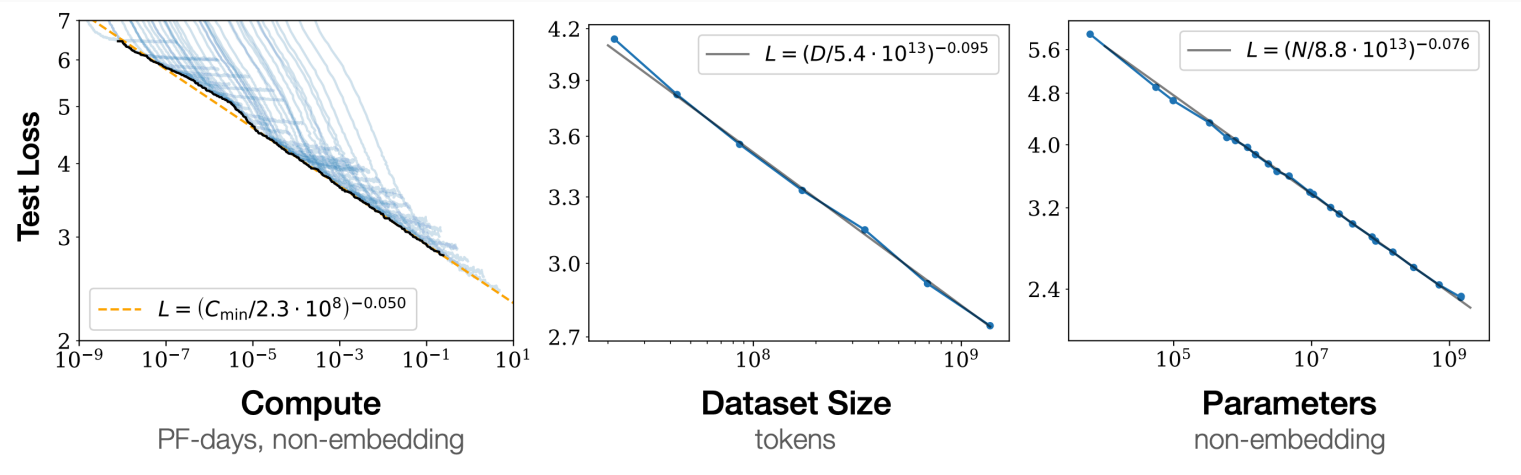


Figure 3: Increasing compute predictably improves language modeling.[2]

[2] Image from [Kaplan et al 2020]. See [2, 4] for more recent scaling laws.

Approach 1: train a good generalist

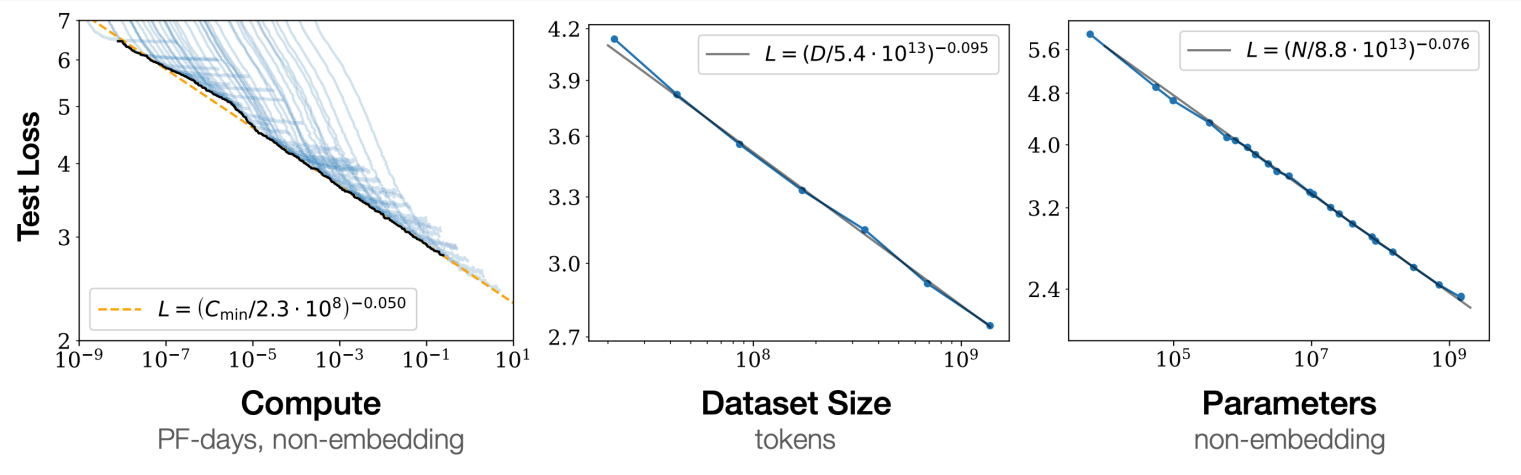


Figure 3: Increasing compute predictably improves language modeling.[2]

- *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, 4] for more recent scaling laws.

Train a good generalist?

Example: Llama 2

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

Train a good generalist?

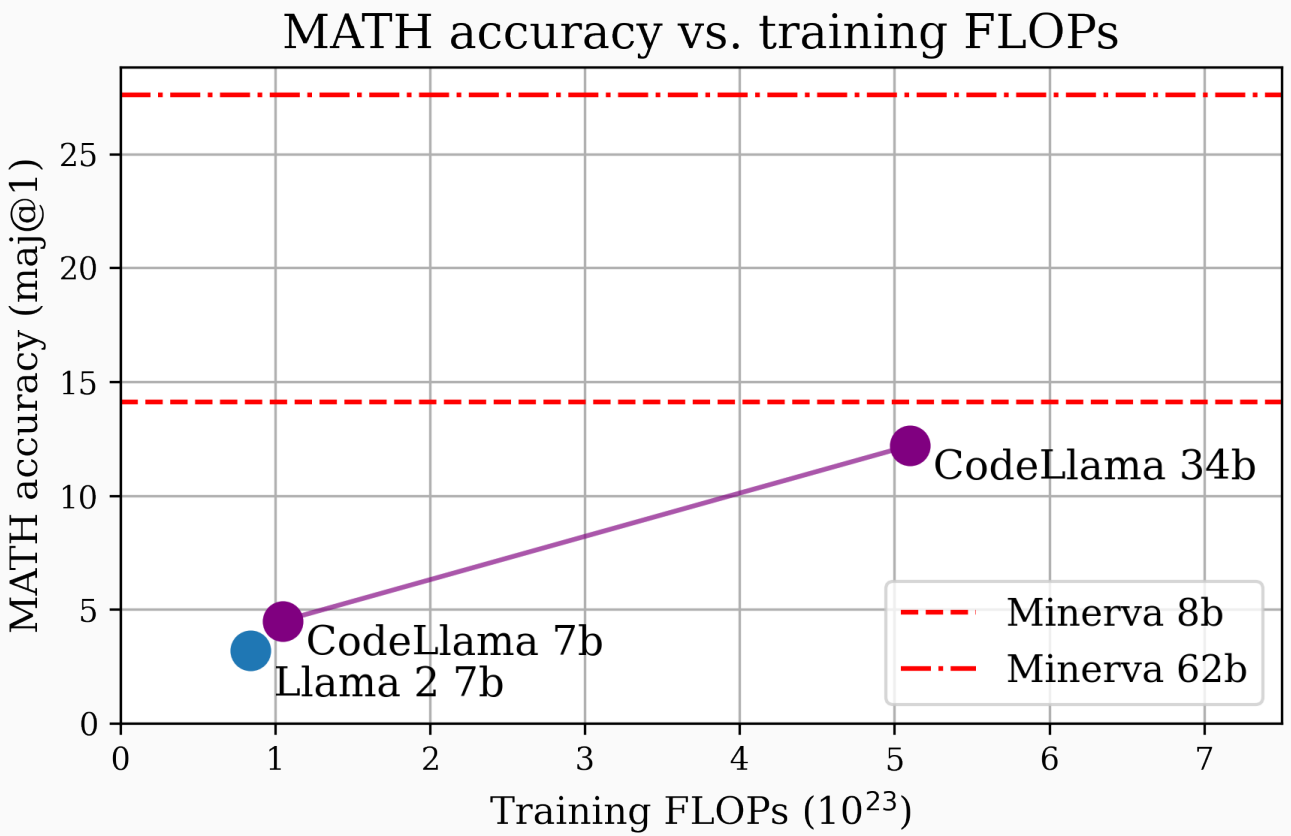


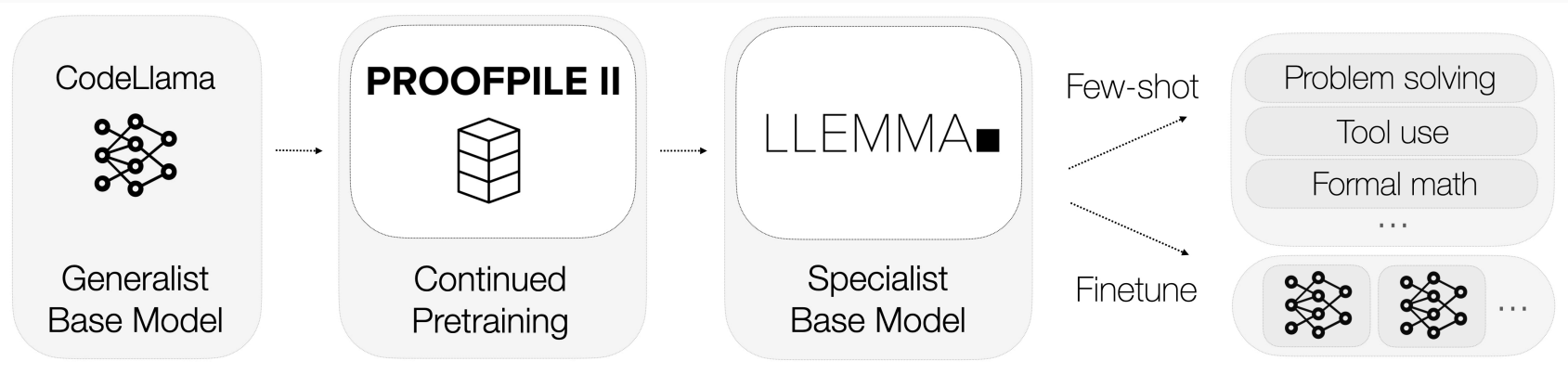
Figure 4: Training a good generalist can be inefficient

Minerva: Google LLM finetuned on math webpages + papers

Approach 2: specialize a generalist model

Recipe for specializing a foundation model to mathematical data:

- Collect high-quality, diverse math-related corpus, PROOFPILE II
- Continue pretraining a base model (e.g., Code LLama) on PROOFPILE II



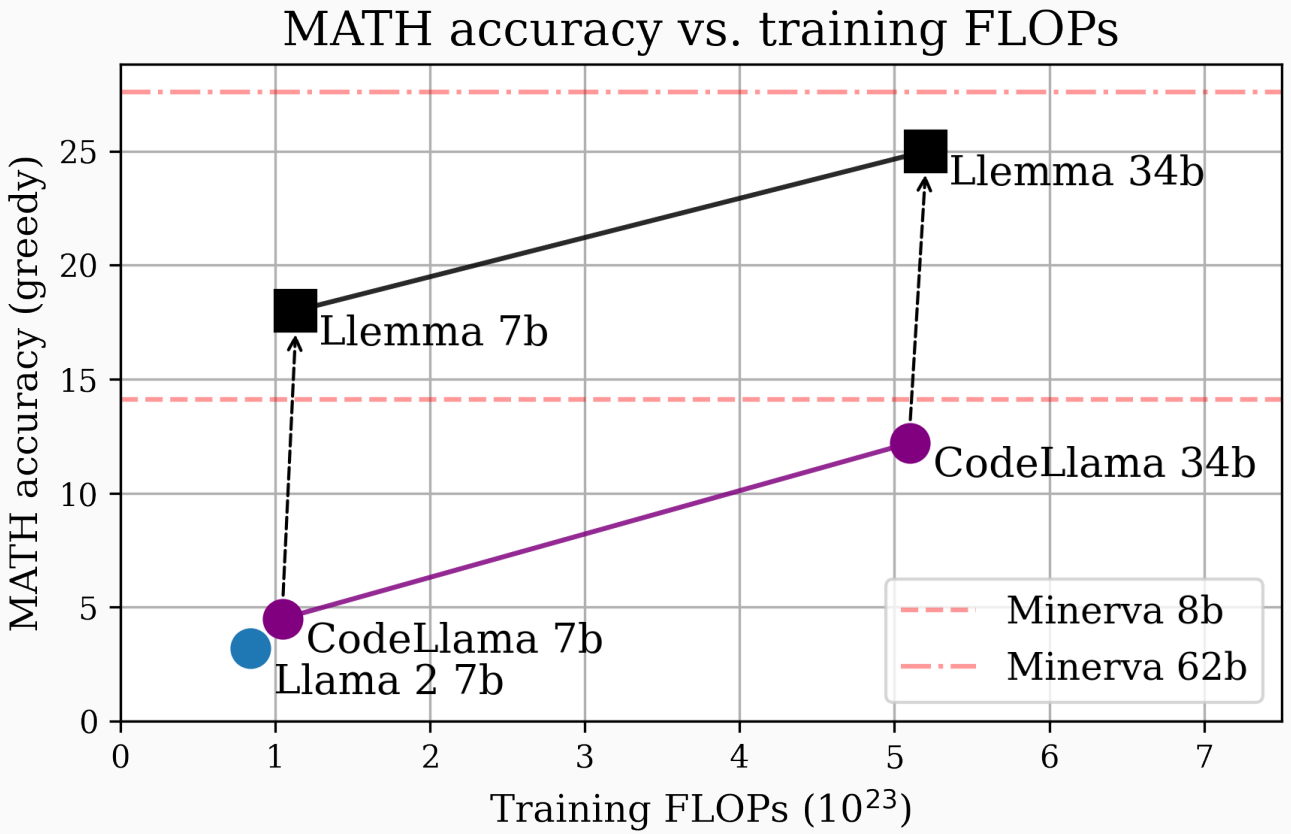


Figure 5: Llemma improves with a modest amount of math-specific compute

Data: Proofpile II

Proofpile II: 55 billion tokens[3]

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

[3] Available at huggingface.co/datasets/EleutherAI/proof-pile-2

Code: AlgebraicStack

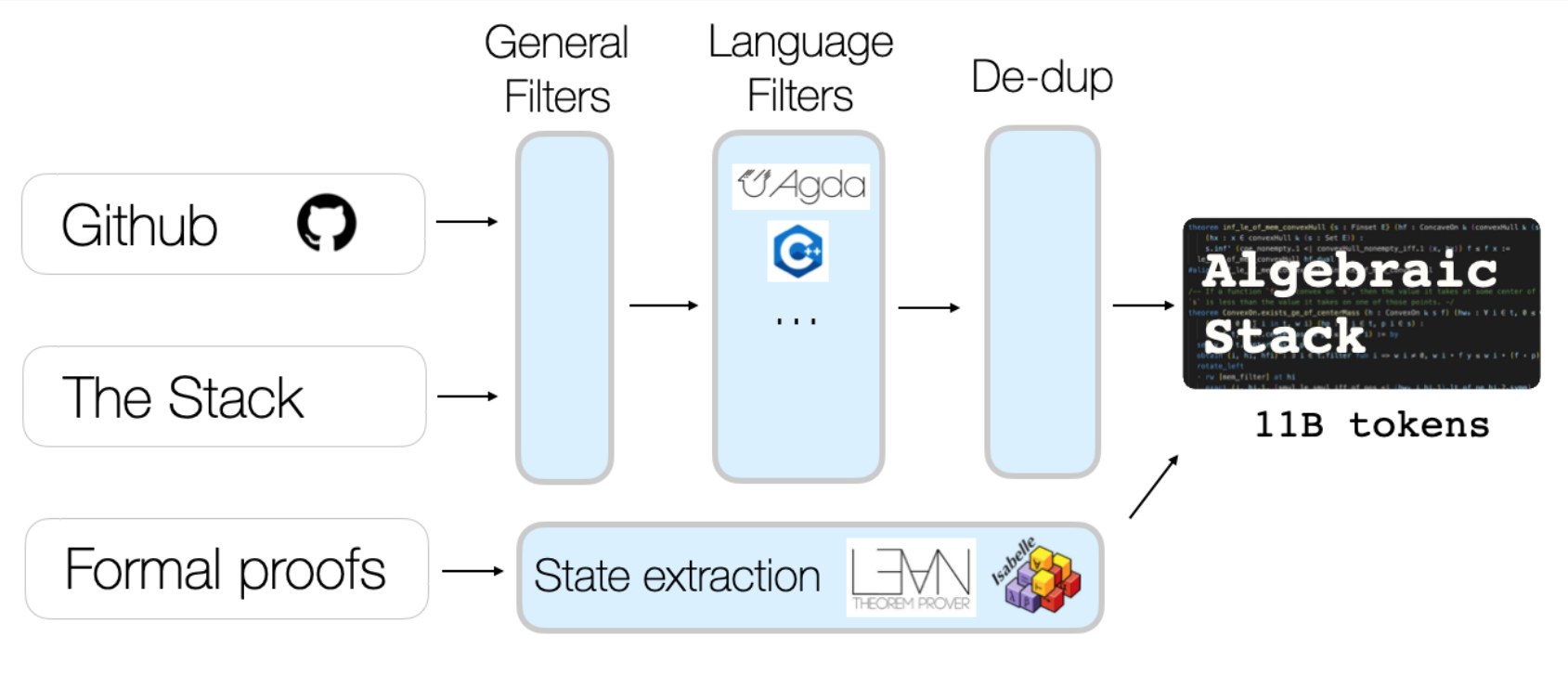


Figure 6: AlgebraicStack pipeline

Web: OpenWebMath

Web: OpenWebMath [Paster et al 2023][4]

- 14.7 billion tokens of math-related web data

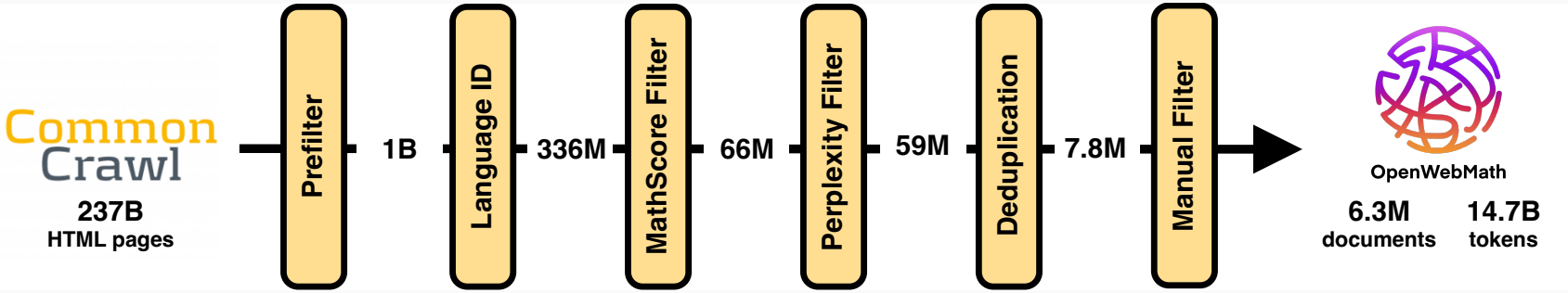


Figure 7: OpenWebMath pipeline.

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

Problem solving with chain-of-thought

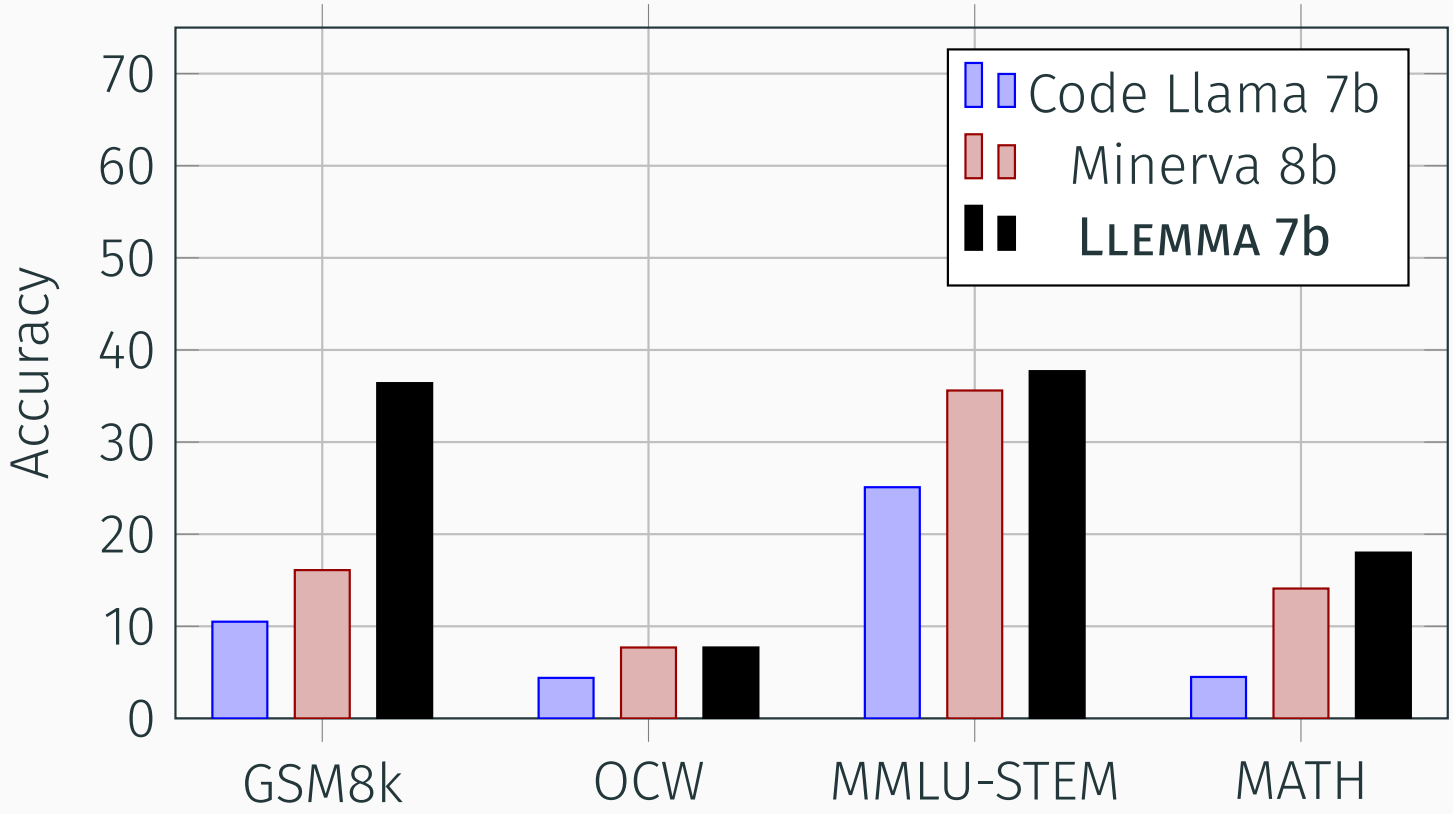


Figure 8: Few-shot problem solving (greedy decoding)

LLEMMA formal-to-formal theorem proving

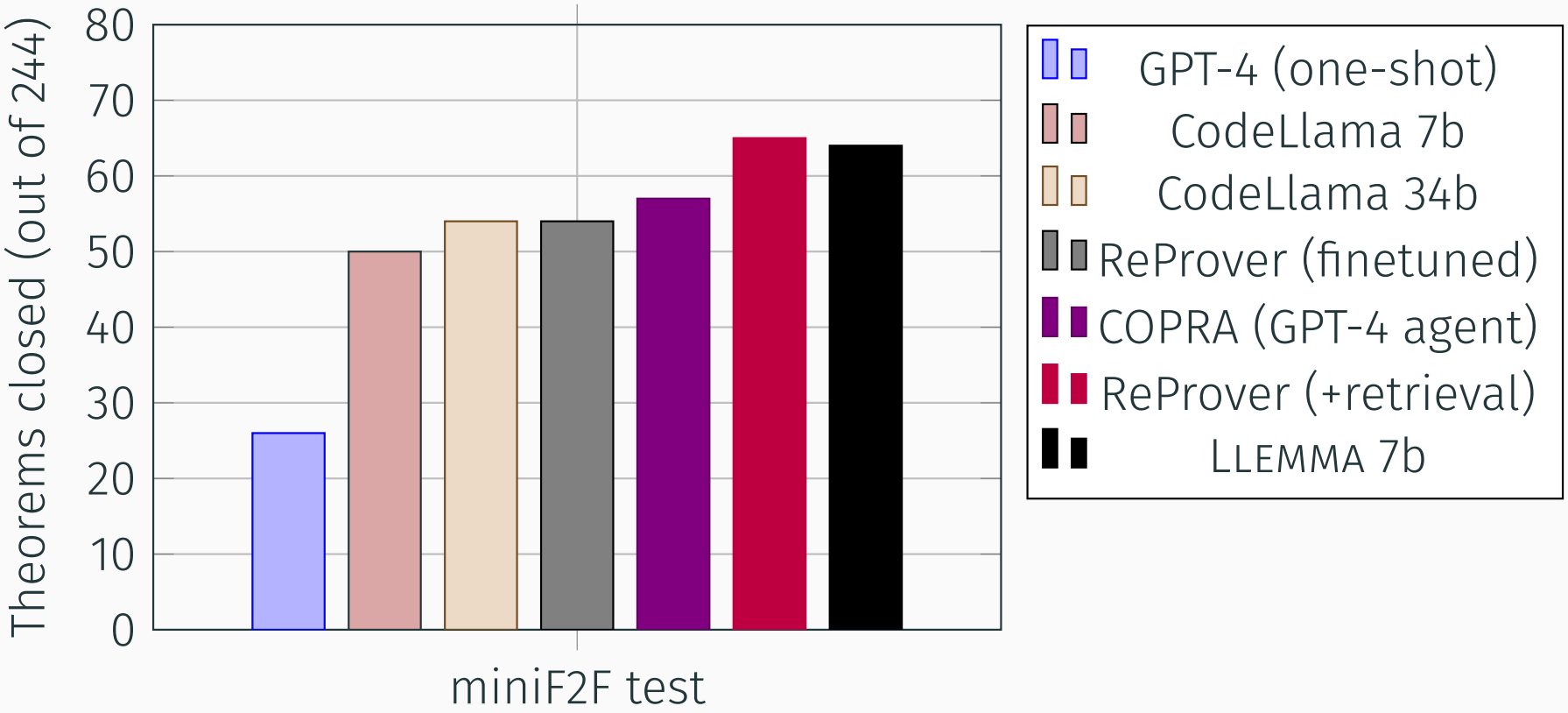


Figure 9: Few-shot formal-2-formal proving with LLEMMA

Integrating LLEMMA and the Lean theorem prover

LLMLEAN: tools that integrate LLMs into the Lean proof assistant

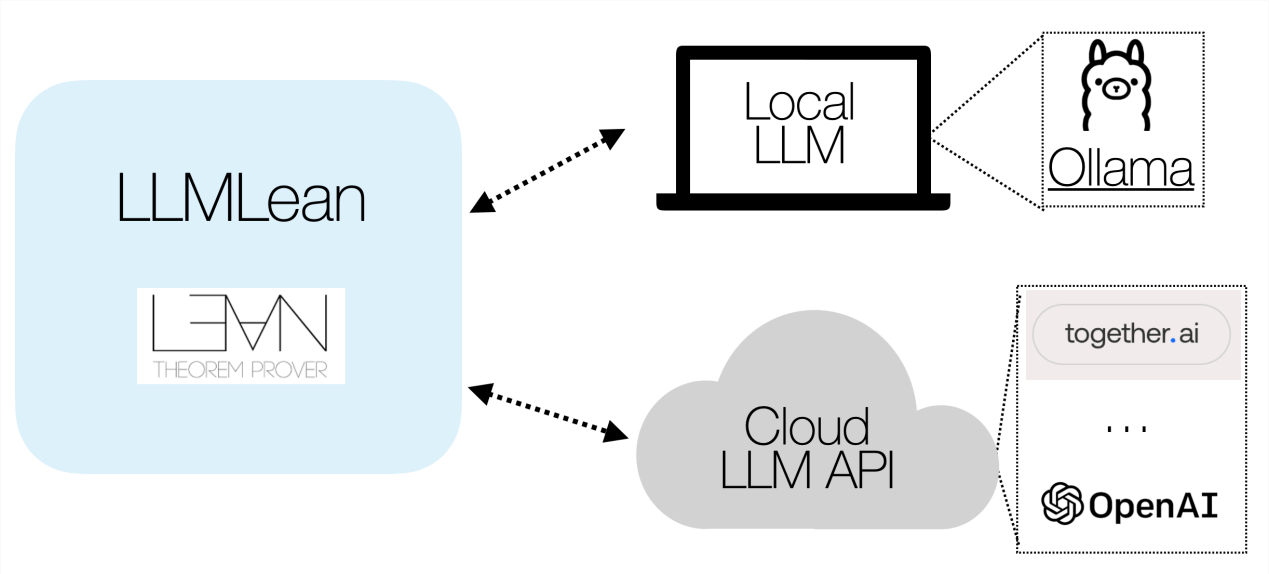


Figure 10: github.com/wellecks/llmlean

Based on *llmstep: LLM proofstep suggestions in Lean* [Welleck & Saha 2023]

Integrating LLEMMA and the Lean theorem prover

Example: Verified LLEMMA suggestions on a MacBook Pro:

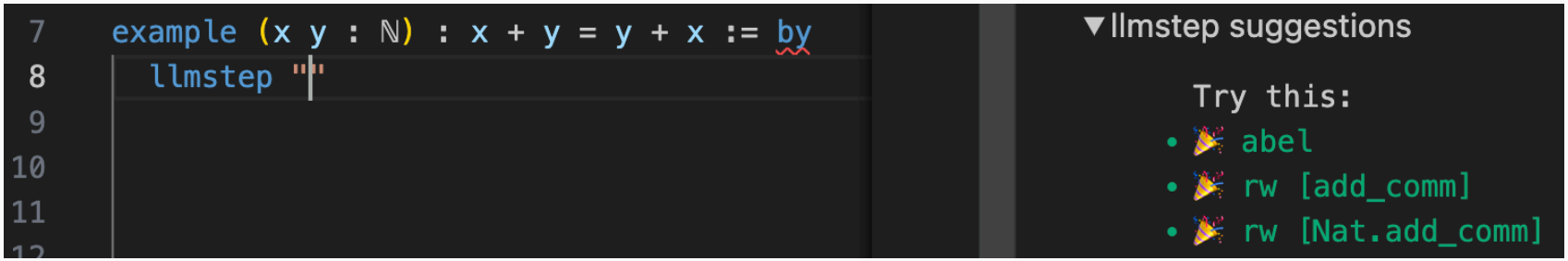


Figure 11: github.com/wellecks/llmlean

Based on *llmstep: LLM proofstep suggestions in Lean* [Welleck & Saha 2023]

Recap

- 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*

Recap

- Focus on data *quality* and *transfer* can enable efficient scaling

Recap

- Focus on data *quality* and *transfer* can enable efficient scaling

Next: going *beyond* human data

II. Easy-to-hard generalization

II. Easy-to-hard generalization

Easy-to-Hard Generalization: Scalable Alignment Beyond Human Supervision

Zhiqing Sun*, Longhui Yu*, Yikang Shen, Weiyang Liu
Yiming Yang^{+}, Sean Welleck^{+}, Chuang Gan^{+}

arXiv 2024.

Easy-to-hard generalization

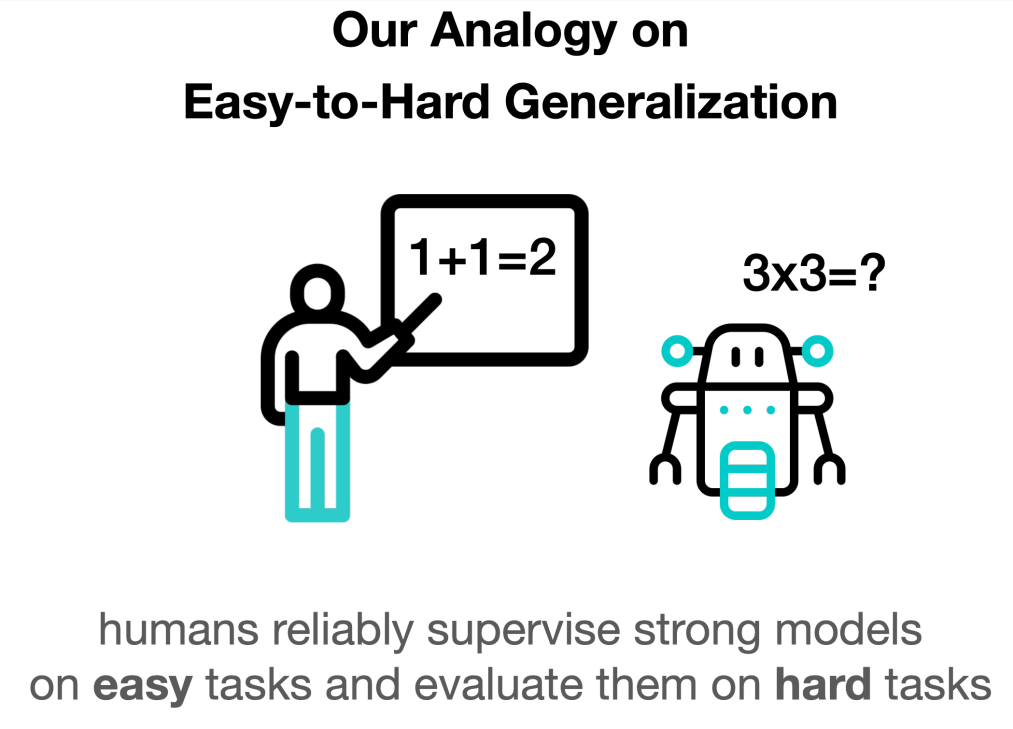


Figure 12: Easy-to-hard generalization

Easy-to-hard generalization

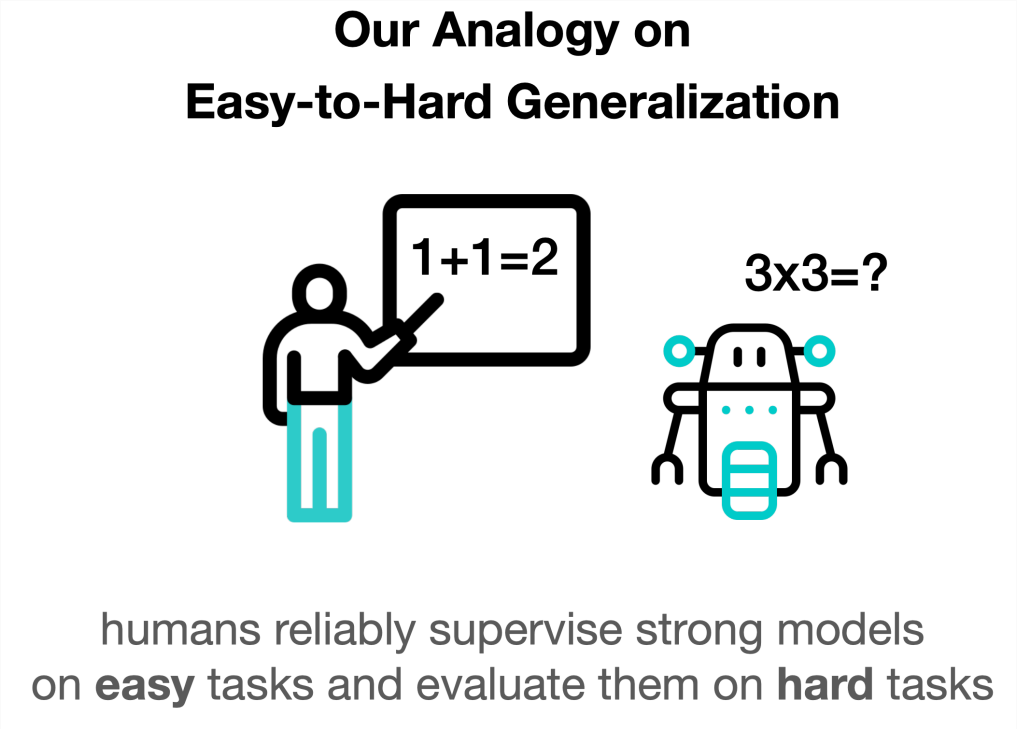


Figure 12: Easy-to-hard generalization

Operationalize as:

- **Train** on **easy** problems: MATH level 1-3
- **Test** on **hard** problems: MATH level 4-5

Easy-to-hard generalization

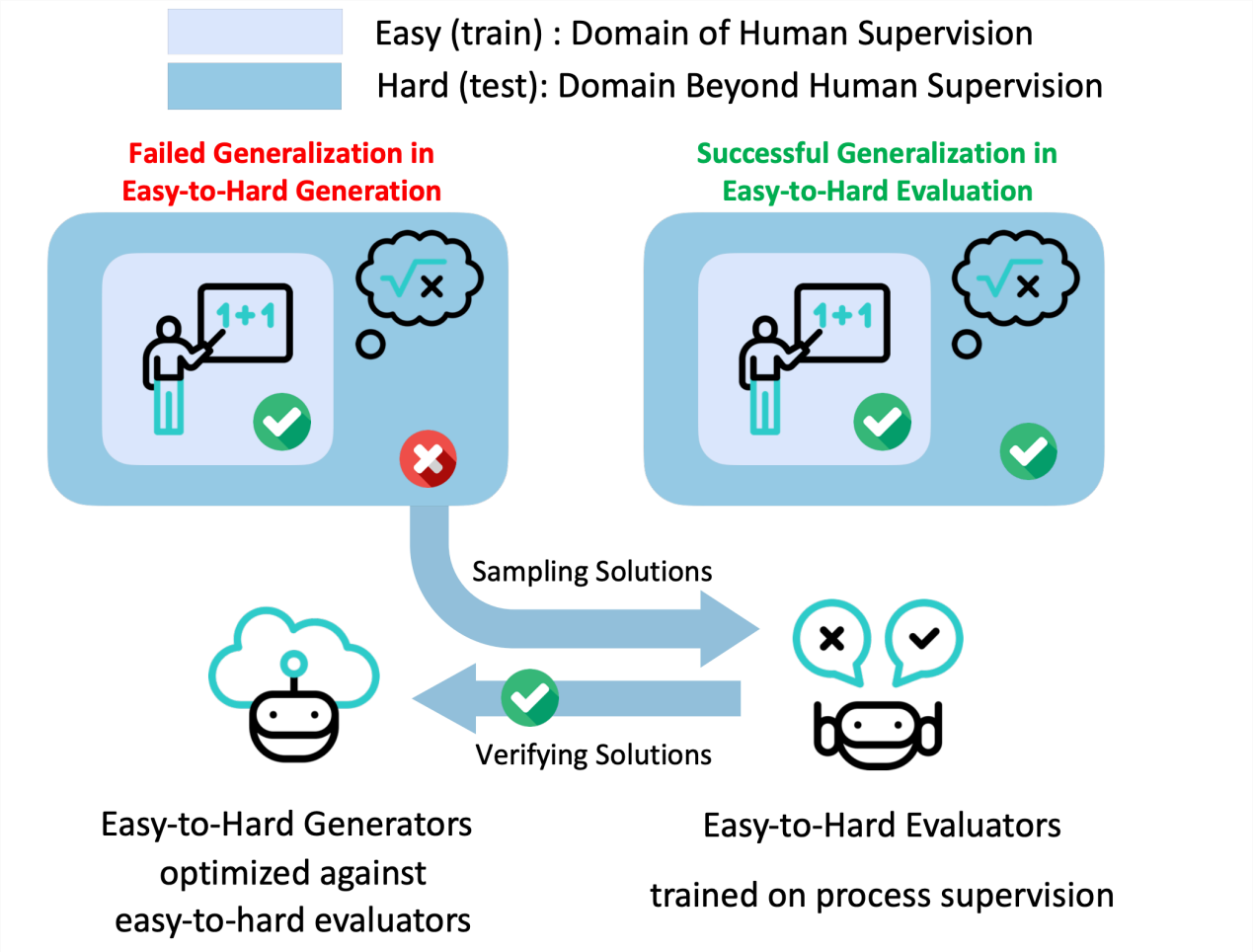


Figure 13: *Key idea:* easy-to-hard evaluation can facilitate easy-to-hard generation

Easy-to-hard generalization

- Train an evaluator $g_\phi(y) \to [0, 1]$ on easy problems
- Use evaluator to select generated solutions $\{y^n\}_{n=1}^N$ on hard problems

Evaluator: *OPRM*

Evaluator: Outcome-process reward model (OPRM)

- Outcome: given solution y, predict correctness[5]
- Process: given steps $y_1, \ldots, y_k$, predict correctness[6]

OPRM is trained to predict both

[5] *E.g., Cobbe et al 2021, Training Verifiers to Solve Math Word Problems*
[6] *E.g., Lightman et al 2023, Let's Verify Step-by-Step*

Inference-time

Select a solution by **weighted majority voting**:

- Generate many solutions (e.g. 1024)
- Score each solution using the evaluator $g_\phi(y)$
- Group the solutions by answer, choose group with highest score

Inference-time scaling on *hard* problems

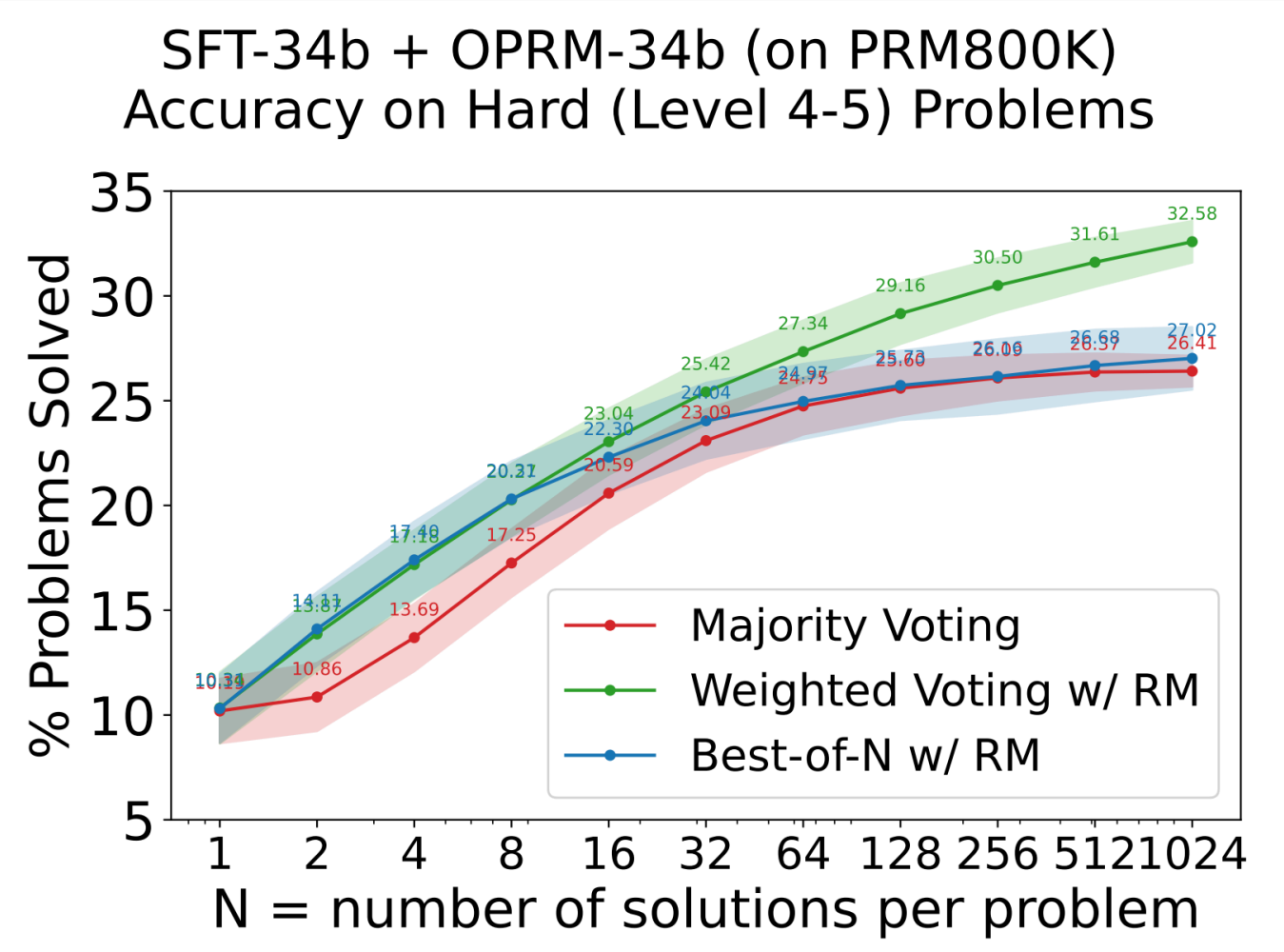


Figure 14: OPRM inference scaling on hard problems

Inference-time scaling on *all* problems

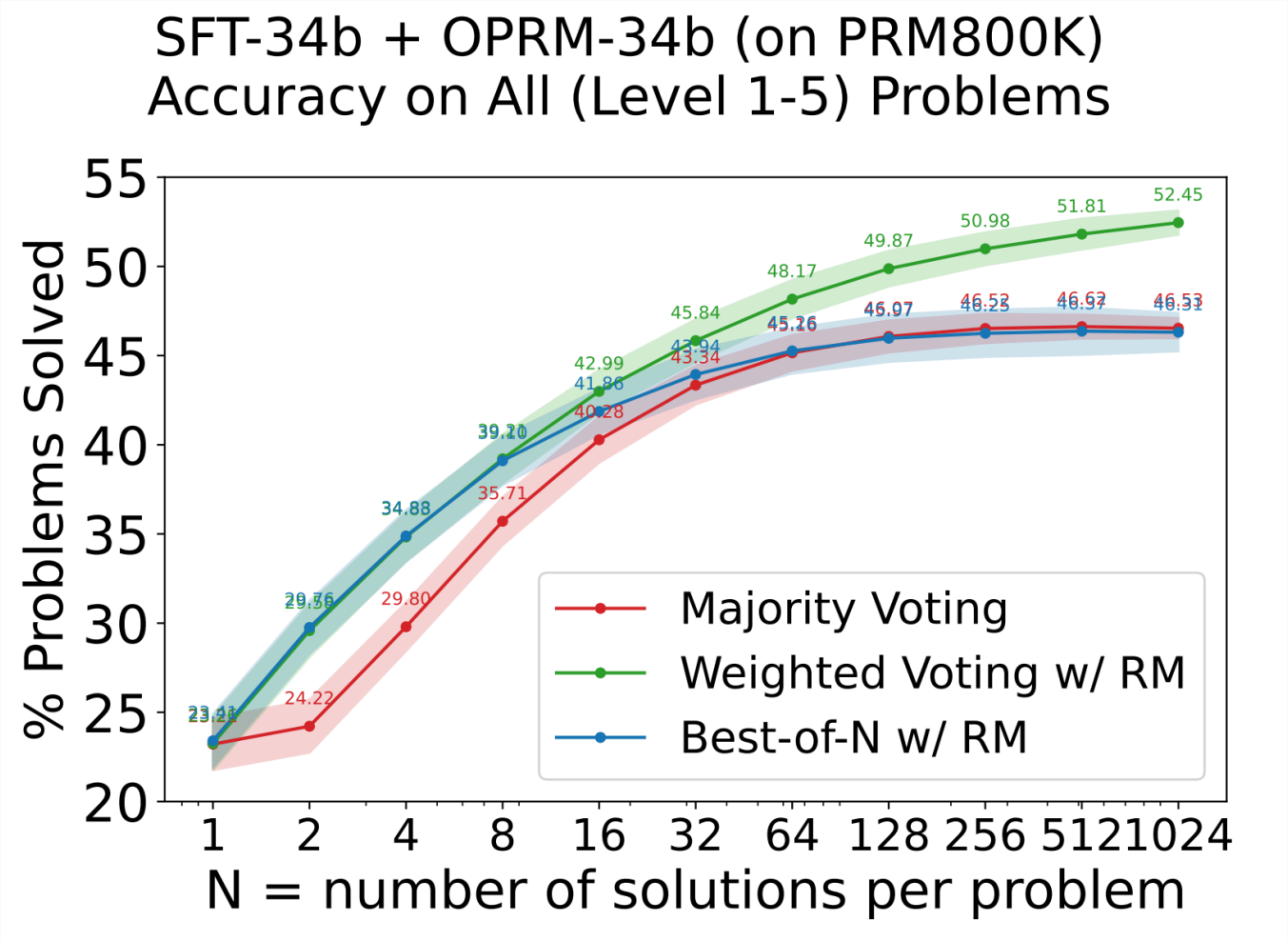
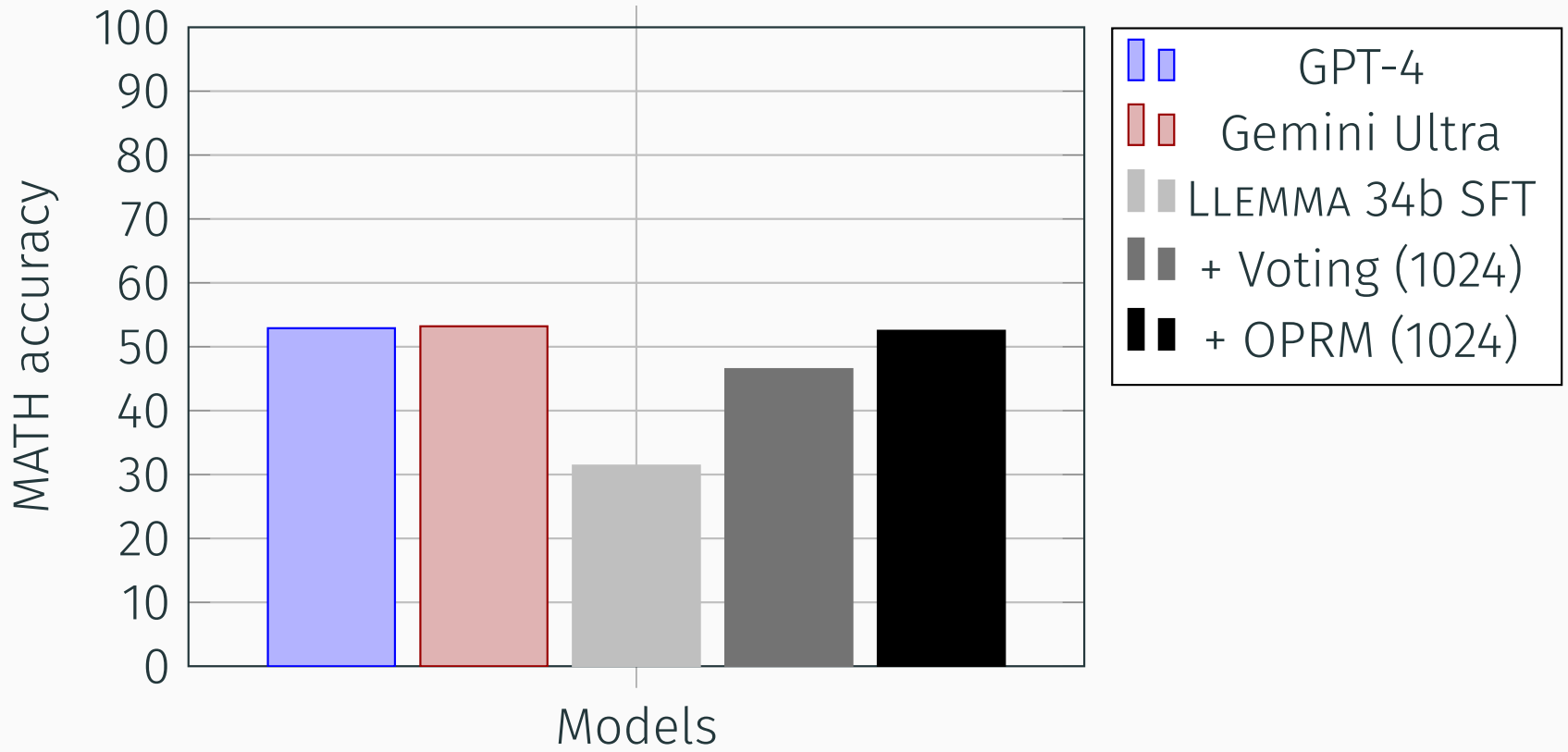


Figure 15: OPRM inference scaling on hard problems

Performance (all problems)

Comparison with black-box API models (GPT-4, Gemini Ultra):[7]



[7] Experiment setting: *PRM 800k*, 34B model

Using the evaluator for reinforcement learning

1. Generate solutions on easy and hard problems
2. Use easy-to-hard evaluator as a reward function

Using the evaluator for reinforcement learning

Outperforms finetuning on *all* problems:[8]

[8] Experiment setting: 7B model, RL with PPO

Recap

1. Scaling data *quality* and *quantity*
    - Llemma and ProofPile II: high-quality data and transfer
2. Scaling *inference* and *evaluation*
    - Scalable evaluation can facilitate easy-to-hard generalization

Looking ahead | data scaling

New data algorithms for mining high-quality data

- E.g., DeepSeek-Math [5]:

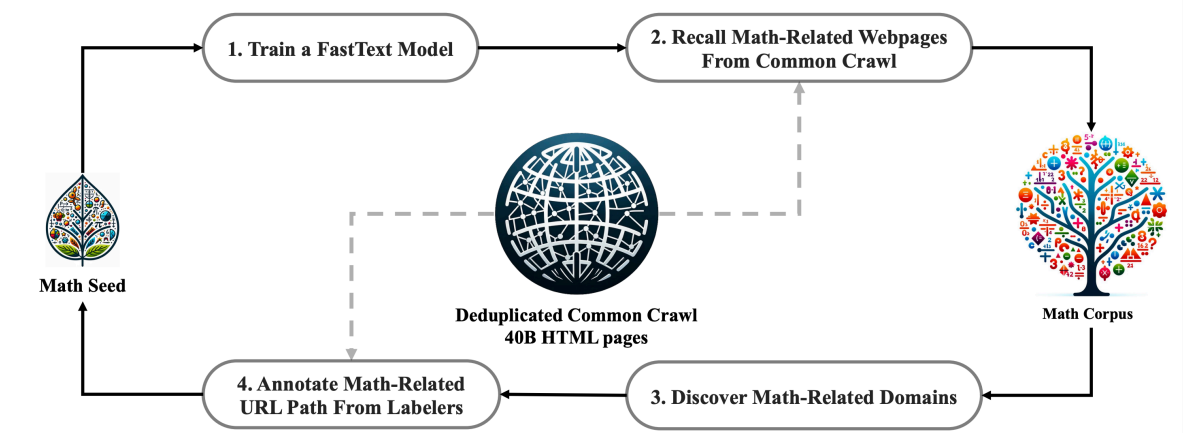


Figure 2 | An iterative pipeline that collects mathematical web pages from Common Crawl.

And synthetic data…

Looking ahead | scaling evaluation

Scalable evaluation beyond 0/1 correctness

- E.g. language feedback and fine-grained rubrics [Kim et al 2024 [3]]

Looking ahead

How do we enable systems that discover new knowledge and improve over time?

Thank you!

- Zhangir Azerbayev (Princeton)
- Hailey Schoelkopf (Eleuther)
- Keiran Paster (Toronto, Vector)
- Marco Dos Santos (Cambridge)
- Stephen McAleer (CMU)
- Albert Jiang (Cambridge)
- Jia Deng (Princeton)
- Stella Biderman (Eleuther)
- Zhiqing Sun (CMU)
- Longhui Yu (Peking U)
- Yikang Shen (MIT-IBM Watson AI Lab)
- Weiyang Liu (Cambridge, MPI)
- Yiming Yang (CMU)
- Chuang Gan (MIT-IBM Watson AI Lab, UMass Amherst)

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

https://github.com/Edward-Sun/easy-to-hard

Sean Welleck (CMU)
Learning, Language, and Logic (L3) Lab

References i

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.

References ii


S. Kim, J. Shin, Y. Cho, J. Jang, S. Longpre, H. Lee, S. Yun, S. Shin, S. Kim, J. Thorne, and M. Seo.
Prometheus: Inducing evaluation capability in language models.
In *The Twelfth International Conference on Learning Representations*, 2024.


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.


Q. Z. R. X. J. S. M. Z. Y. L. Y. W. D. G. Zhihong Shao, Peiyi Wang.
Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.