

Lec 5: Pre/Mid/Post-Training

CS 2281, Fall 2024

**Sham Kakade and
David Brandfonbrener**

Today

- Pre-training
- Mid-training
- Post-training
- Reasoning/o1

A few remarks on Science & Scale in AI

- Science matters
 - yes, we may fail to reproduce everything, but this is science working as intended.
 - data > opinions
- Math in AI
 - Can be helpful: diffusion models, inference/SSMs, optimization
 - Deeper perspectives: JAX's view of AD
- Scaling is an important part of the engineering process
 - If somethings holds at smaller scale, makes sense to consider this as the “null”
 - Yes, we do need the right scale to study the effect of interest (some phenomena “emerge” but often there are predictive metrics)
- It's your job to look at the data and draw your own conclusions

The “pipeline”

- Pre-training: the lions share of flops
 - learn the density of all txt out there
 - diminishing returns in ppl/loss, but best proxy for downstream performance
- Mid-training: relatively new paradigm
 - once our base model is “good”, can we utilize to get a better model?
 - maybe a different “scaling law” is possible?
- Post-training: how can we deploy a helpful model
- **Reasoning/o1**: use more inference time compute!
 - Is this a different paradigm?

Today

- ✓ • Pre-training
 - How much data is out there?
 - Web scraping and processing
 - Scaling laws
 - Benchmarks and emergence
 - Data selection/Data mixtures
- Mid-training
- Post-training

What is “pre-training”?

- Goal: estimate the probability density on “all” raw available text.
 - We train BIG models on LOTS of data
- Which text is “relevant” for downstream tasks?
 - Difficult to say and quite plausibly diminishing (yet important) returns
 - Small bumps in perplexity are important for downstream performance
- What “data mixture” do we want?
- How much data is out there?

Some LLM numbers

ChatGPT (GPT 3.5):

“human level performance on many benchmarks + sometimes absurd mistakes”

Grapevine numbers:

- GPT3.5: 200B model, trained on 1-5T tokens?
- GPT4.0: 1.6T MoE (8x200B models), 5-12T tokens
- Gemini: 2T param model (also MoE?), 15T?
(trained on TPUs, 1M context)

Published numbers:

- LLama3: 405B, 15T tokens

Today

- Pre-training
 - ✓ • How much data is out there?
 - Web scraping and processing
 - Scaling laws
 - Benchmarks and emergence
 - Data selection/Data mixtures
- Mid-training
- Post-training

How much data is out there?

- LLama3: 15T tokens
- Plausibly another 20-50T “good data” that may be hard to reach
- Non-English 20-50T?

	Words	Tokens	Relative size (Llama 3 = 1)
Web Data			
FineWeb	11 trillion	15T	1
Non-English web data (high quality)	13.5 trillion	18T	1
Code			
Public code	–	0.78T	0.05
Private Code	–	20T	1.3
Academic publications and patents			
Academic articles	800 billion	1T	0.07
Patents	150 billion	0.2T	0.01
Books			
Google Books	3.6 trillion	4.8T	0.3
Anna’s Archive (books)	2.8 trillion	3.9T	0.25
Every unique book	16 trillion	21T	1.4
Social media			
Twitter / X	8 trillion	11T	0.7
Weibo	29 trillion	38T	2.5
Facebook	105 trillion	140T	10

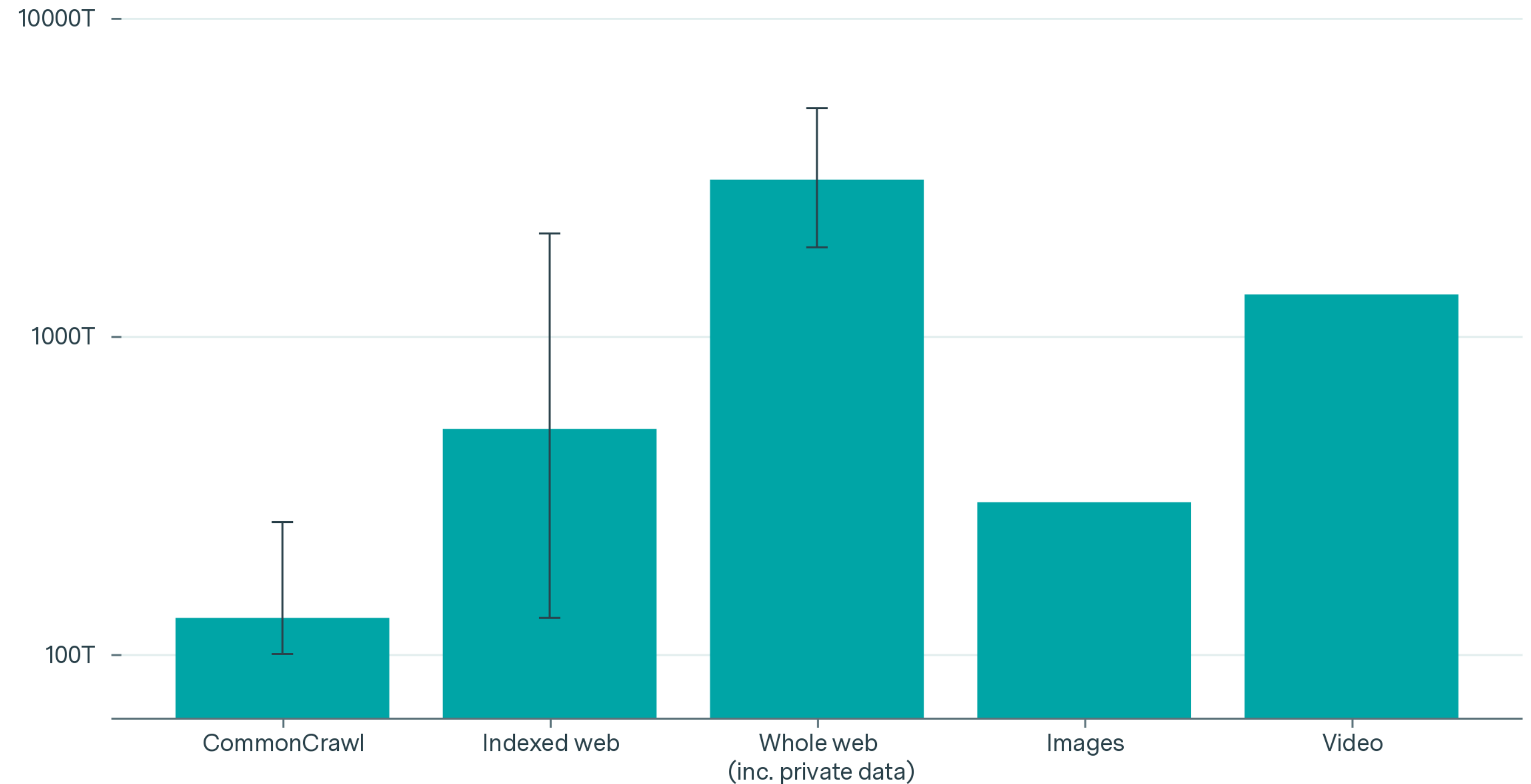
[Blogpost:](#) How much LLM training data is there in the limit?

How much data is out there?

Estimates of different stocks of data



Effective stock (number of tokens)



- Total web data may be 1000T, but not high quality
- Much more private data
- More data in other modalities (but this is less clearly useful)

Today

- Pre-training
 - How much data is out there?
 - ✓ • Web scraping and processing
 - Scaling laws
 - Benchmarks and emergence
 - Data selection
- Mid-training
- Post-training

Web scraping: where does data come from?



Over **250 billion** pages spanning 17 years.

Free and open corpus since 2007.

Cited in over **10,000** research papers.

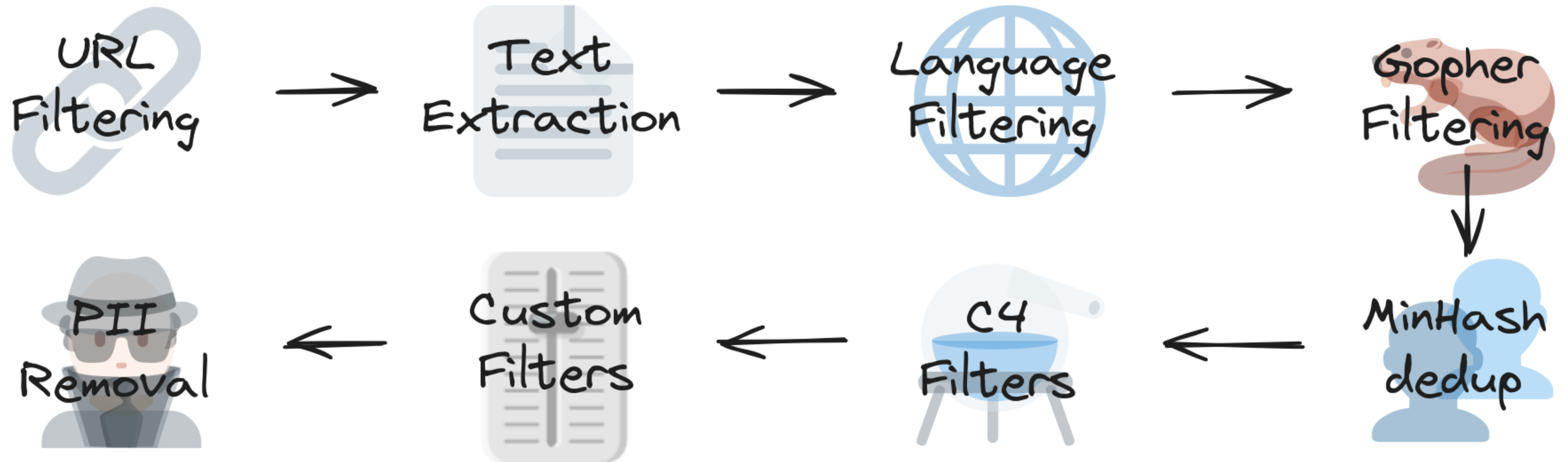
3–5 billion new pages added each month.



Source	Doc Type	UTF-8 bytes (GB)	Documents (millions)	Unicode words (billions)	Llama tokens (billions)
Common Crawl	web pages	9,812	3,734	1,928	2,479
GitHub	code	1,043	210	260	411
Reddit	social media	339	377	72	89
Semantic Scholar	papers	268	38.8	50	70
Project Gutenberg	books	20.4	0.056	4.0	6.0
Wikipedia, Wikibooks	encyclopedic	16.2	6.2	3.7	4.3
Total		11,519	4,367	2,318	3,059

Table 1: The Dolma corpus at-a-glance. It consists of three trillion tokens sampled from a diverse set of domains; sourced from approximately 200 TB of raw text before curation down to an 11 TB dataset. It has been extensively cleaned for language model pretraining use. Tokens calculated using the LLaMA tokenizer.

Web scraping: where does data come from?



The Fineweb pipeline

Web scraping: where does data come from?

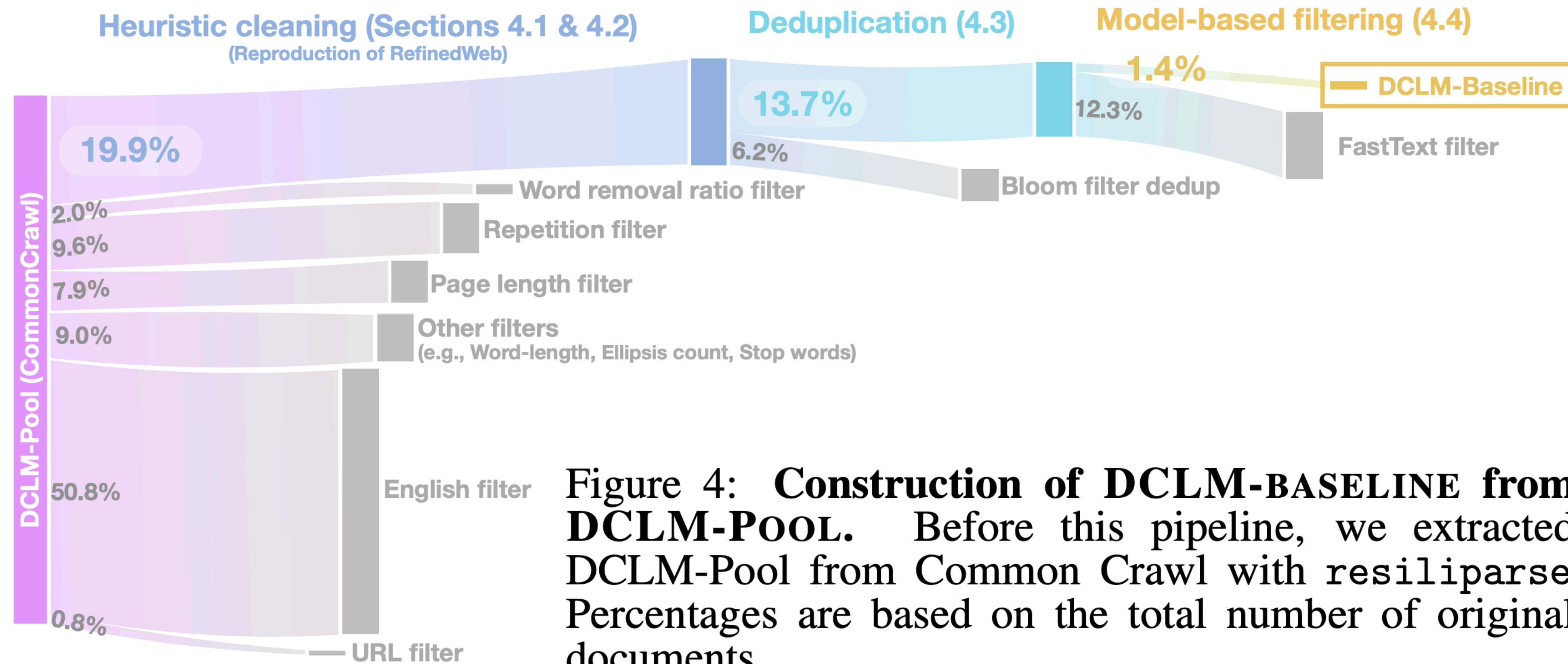


Figure 4: **Construction of DCLM-BASELINE from DCLM-POOL.** Before this pipeline, we extracted DCLM-Pool from Common Crawl with resiliiparse. Percentages are based on the total number of original documents.

Today

- Pre-training
 - How much data is out there?
 - Web scraping and processing
 - ✓ • Scaling laws
 - Benchmarks and emergence
 - Data selection/Data mixtures
- Mid-training
- Post-training

Scaling laws

Key:

N: parameters

D: data

C: compute

E: entropy

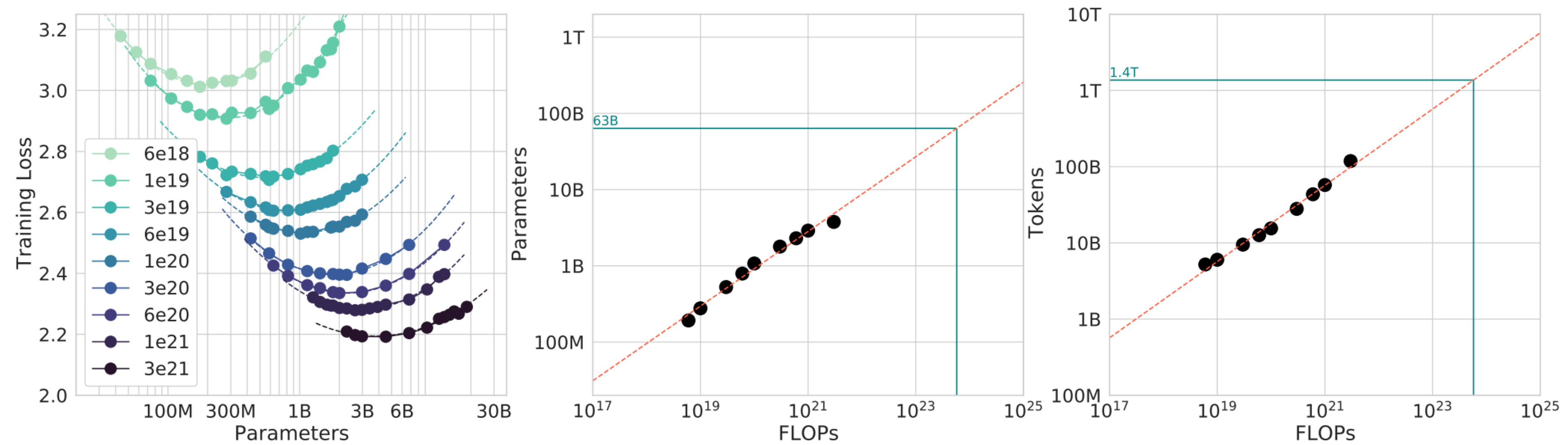
Note: data and parameters have diminishing returns

$$C \approx 6ND$$

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^\alpha} + \frac{B}{D^\beta}.$$

$$N_{opt}(C) = G \left(\frac{C}{6} \right)^a, \quad D_{opt}(C) = G^{-1} \left(\frac{C}{6} \right)^b,$$

Scaling laws

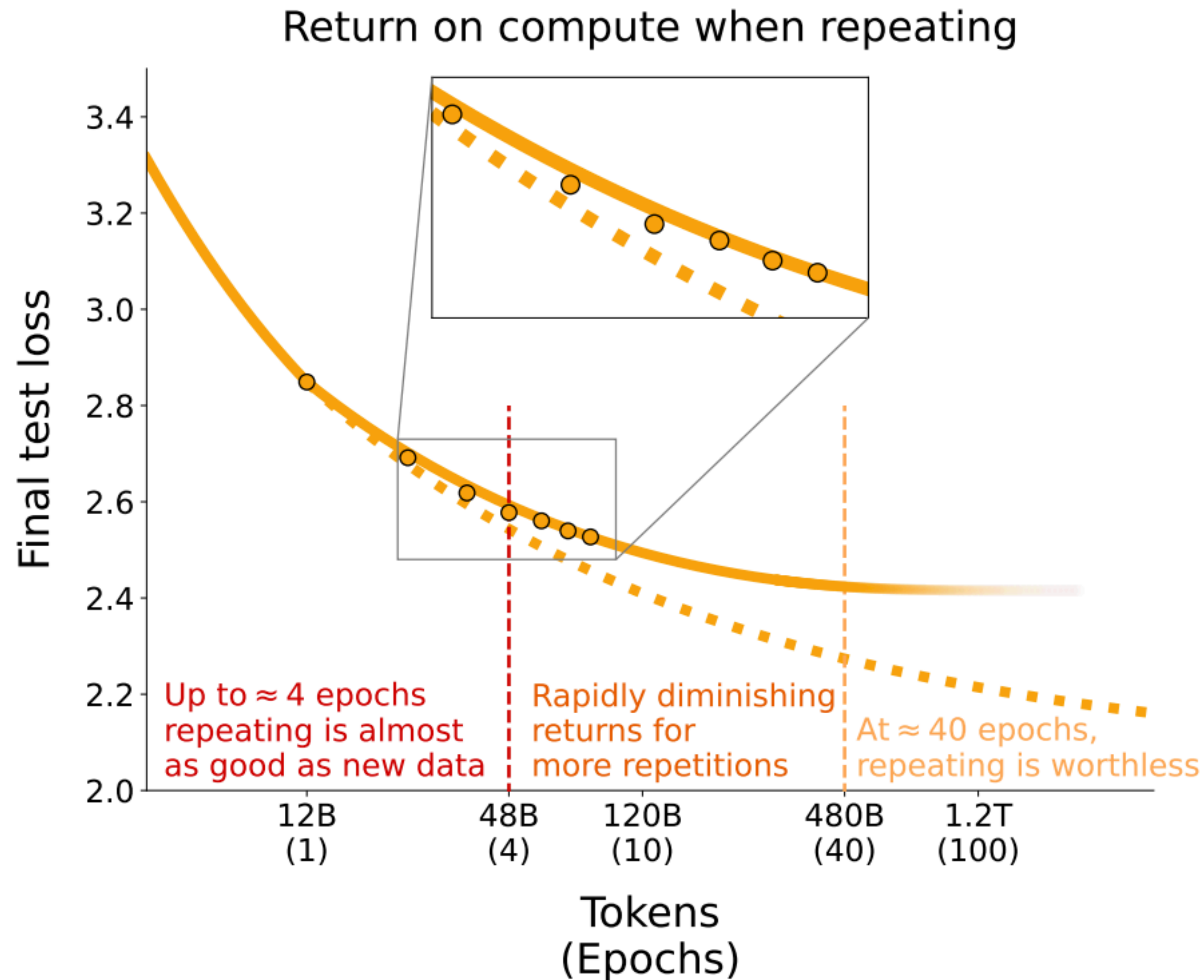


- Rule of thumb: tokens = 20 x parameters
- But Llama models train for much longer

Scaling laws when repeating data

Running out of data options:

1. Repeat data
2. Bigger models



Today

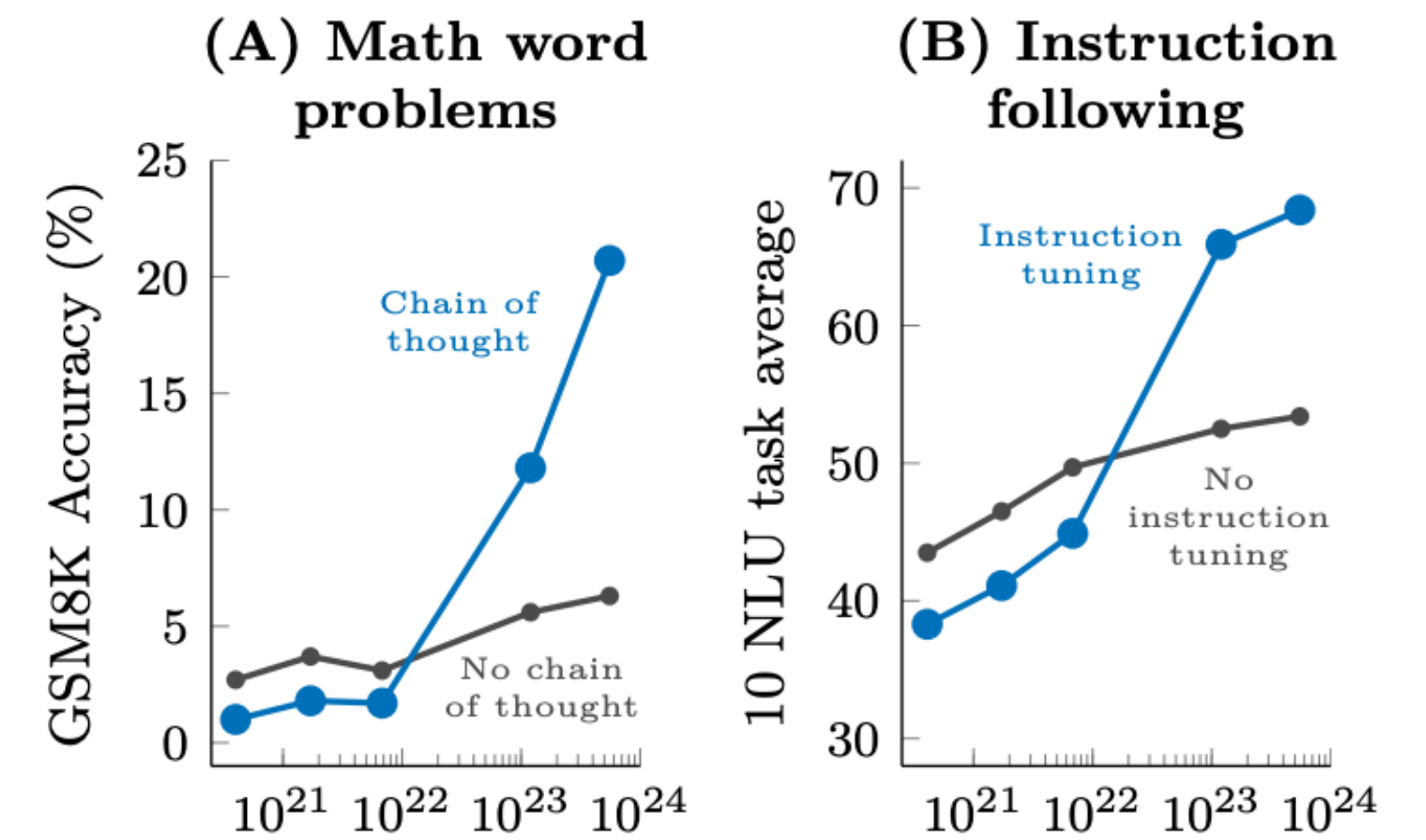
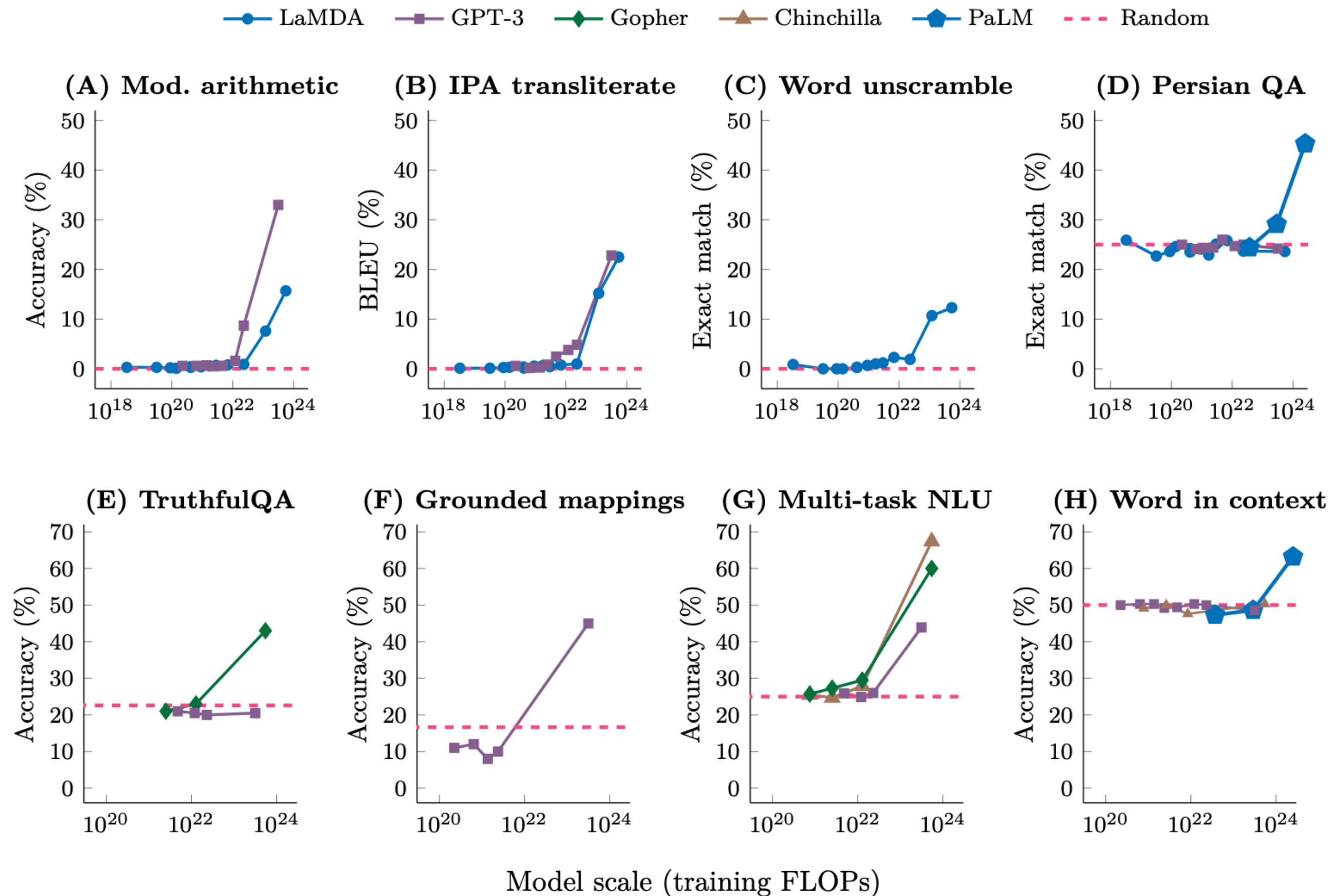
- Pre-training
 - How much data is out there?
 - Web scraping and processing
 - Scaling laws
 - ✓ • Benchmarks and emergence
 - Data selection/Data mixtures
- Mid-training
- Post-training

Downstream benchmarks

- We don't actually care about the loss directly, we care about what the model can do “downstream”
- But pre-training loss is still the best universal proxy
- Examples of standard downstream tasks:
 - Multiple choice:
 - High school/college exams (MMLU)
 - Commonsense reasoning (Hellaswag, ARC, Winogrande)
 - Reading comprehension
 - Generative:
 - Code generation (humaneval, MBPP, SWEbench)
 - Math (GSM8k, hendrycks-MATH)
 - Summarization
 - Chat (chatbot arena)

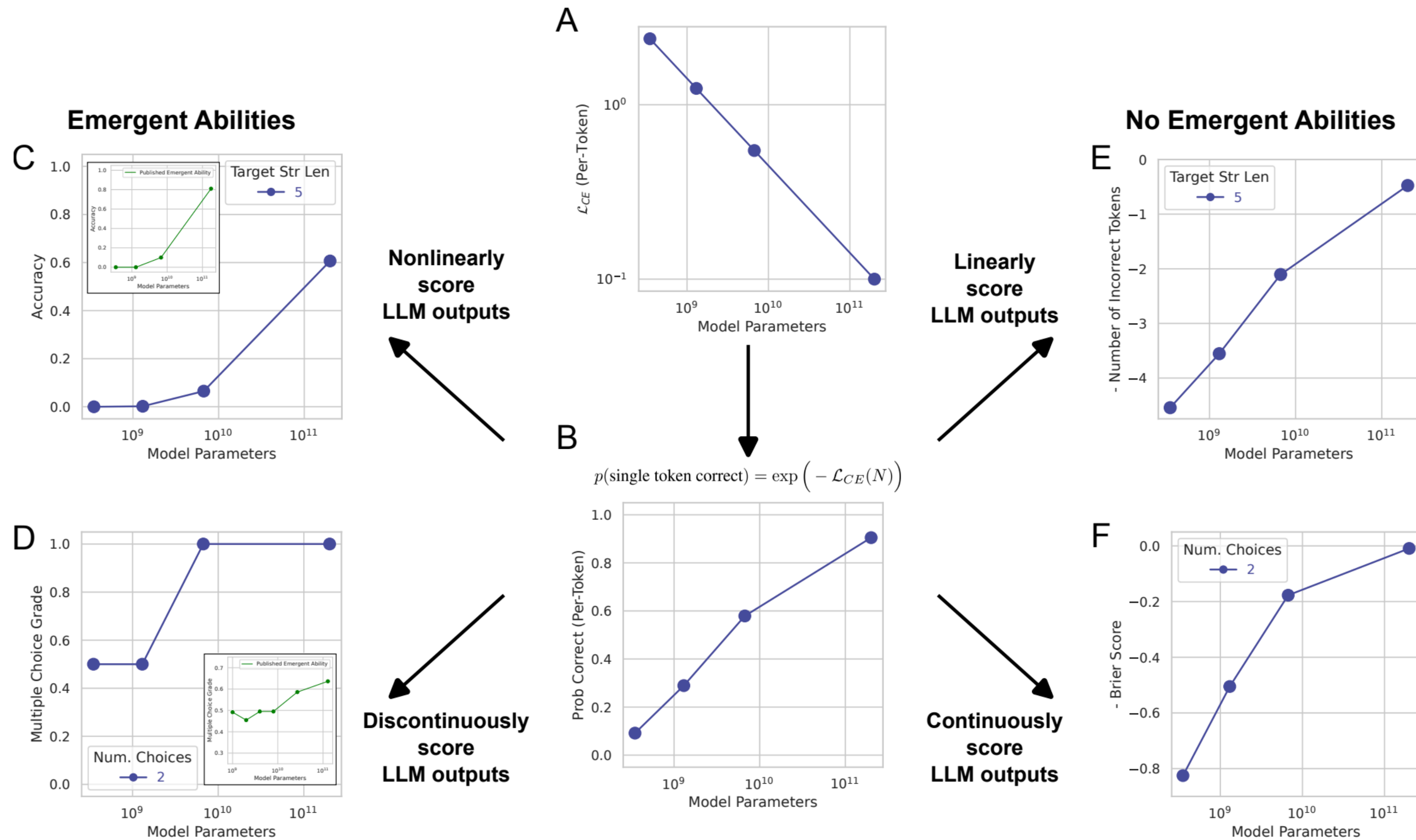
Emergence

- Small gains in pre-training loss can have potentially large impacts downstream



Emergence?

- If we measure smoother metrics, emergence is less clear



Today

- Pre-training
 - How much data is out there?
 - Web scraping and processing
 - Scaling laws
 - Benchmarks and emergence
 - ✓ • Data selection/Data mixtures
- Mid-training
- Post-training

Data selection: what data will help most downstream?

- Pre-training loss is the best proxy, but what data should go into the pre-training distribution?

Idea 1: educational content filter

Educational values deemed by the filter

High educational value

```
import torch
import torch.nn.functional as F

def normalize(x, axis=-1):
    """Performs L2-Norm."""
    num = x
    denom = torch.norm(x, 2, axis, keepdim=True)
    .expand_as(x) + 1e-12
    return num / denom

def euclidean_dist(x, y):
    """Computes Euclidean distance."""
    m, n = x.size(0), y.size(0)
    xx = torch.pow(x, 2).sum(1, keepdim=True).
    expand(m, n)
    yy = torch.pow(y, 2).sum(1, keepdim=True).
    expand(m, m).t()
    dist = xx + yy - 2 * torch.matmul(x, y.t())
    dist = dist.clamp(min=1e-12).sqrt()
    return dist

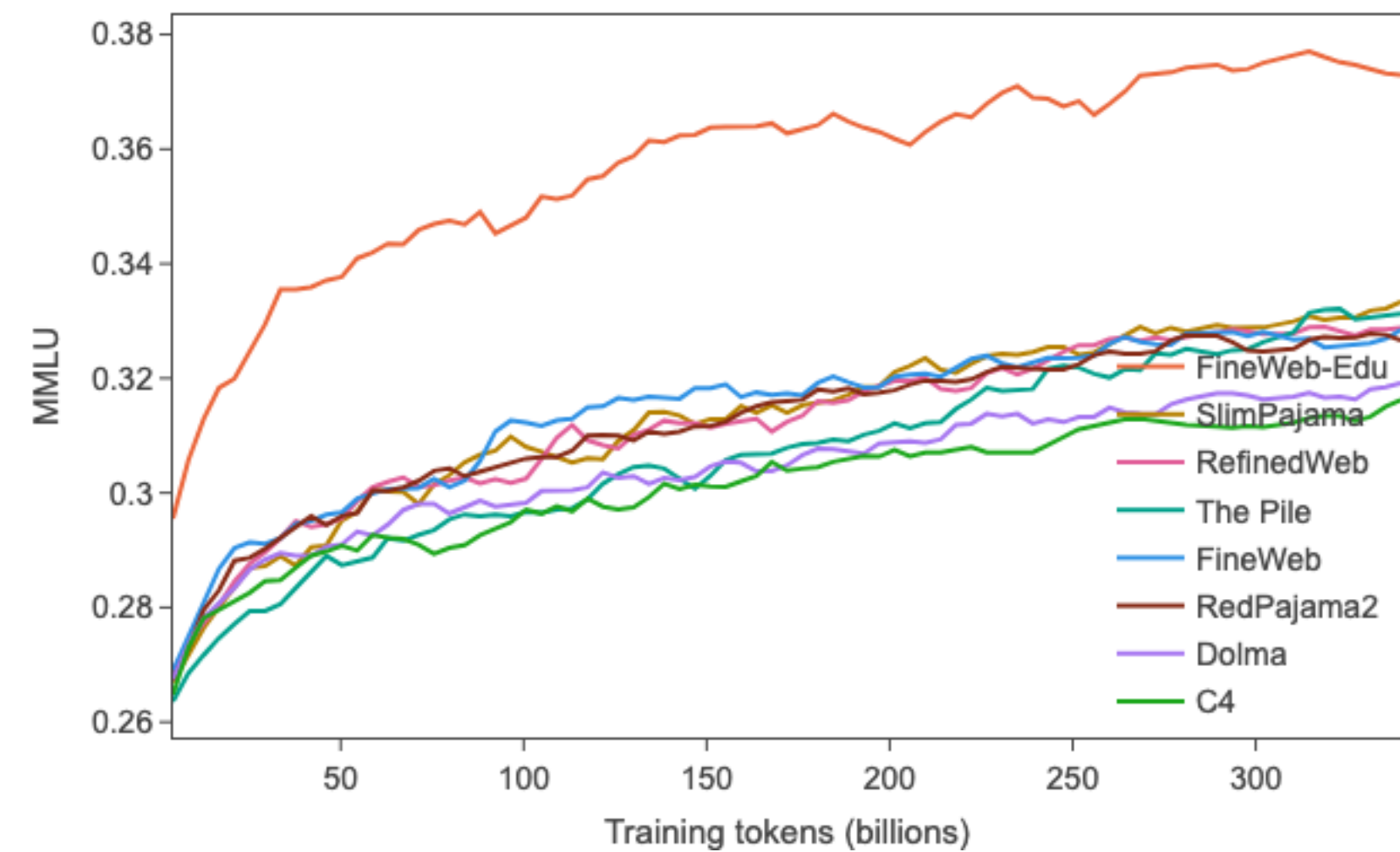
def cosine_dist(x, y):
    """Computes Cosine Distance."""
    x = F.normalize(x, dim=1)
    y = F.normalize(y, dim=1)
    dist = 2 - 2 * torch.mm(x, y.t())
    return dist
```

Low educational value

```
import re
import typing
...

class Default(object):
    def __init__(self, vim: Nvim) -> None:
        self._vim = vim
        self._denite: typing.Optional[SyncParent]
        = None
        self._selected_candidates: typing.List[int]
        ] = []
        self._candidates: Candidates = []
        self._cursor = 0
        self._entire_len = 0
        self._result: typing.List[typing.Any] = []
        self._context: UserContext = {}
        self._bufnr = -1
        self._winid = -1
        self._winrestcmd = ''
        self._initialized = False
        self._winheight = 0
        self._winwidth = 0
        self._winminheight = -1
        self._is_multi = False
        self._is_async = False
        self._matched_pattern = ''
        ...
```

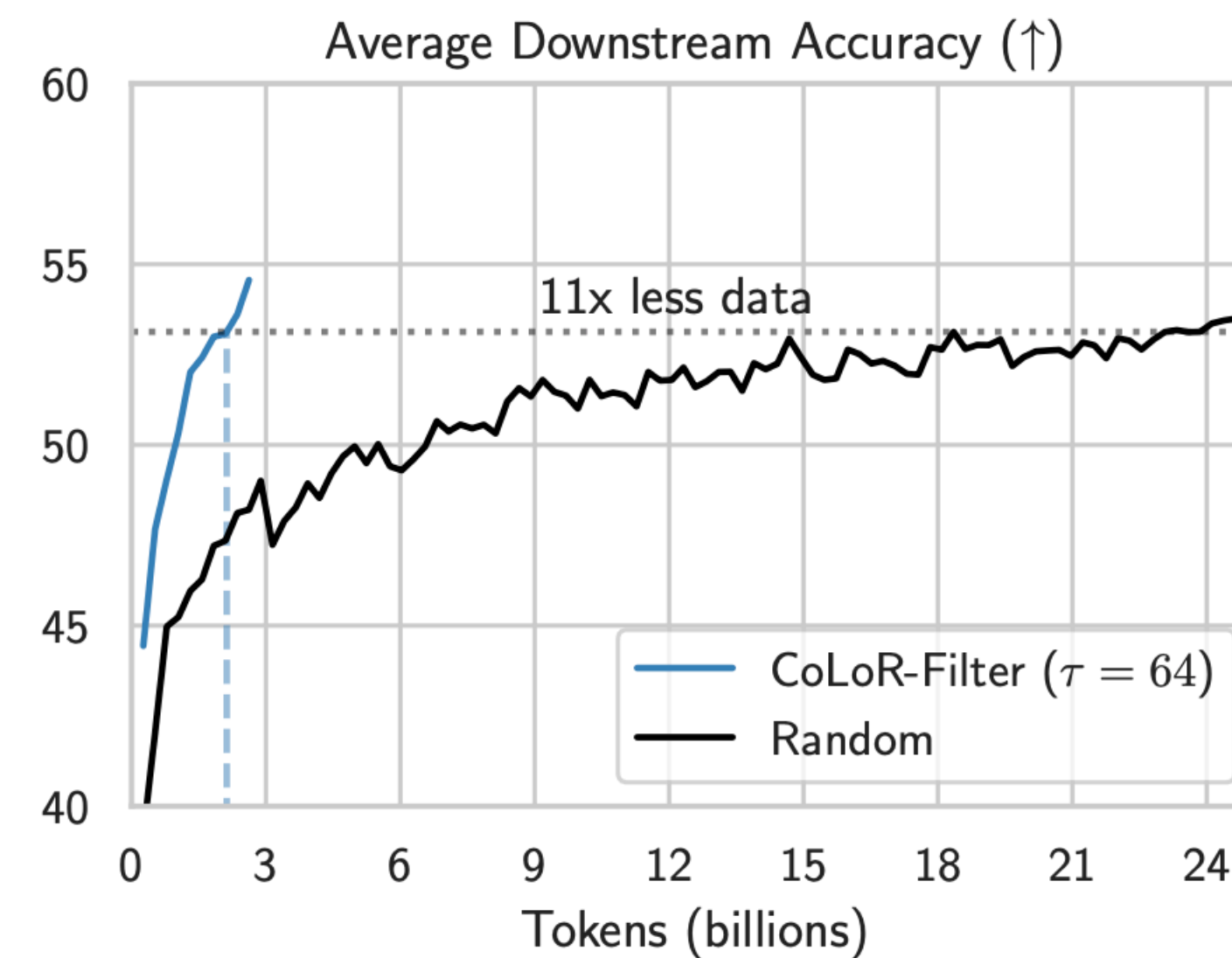
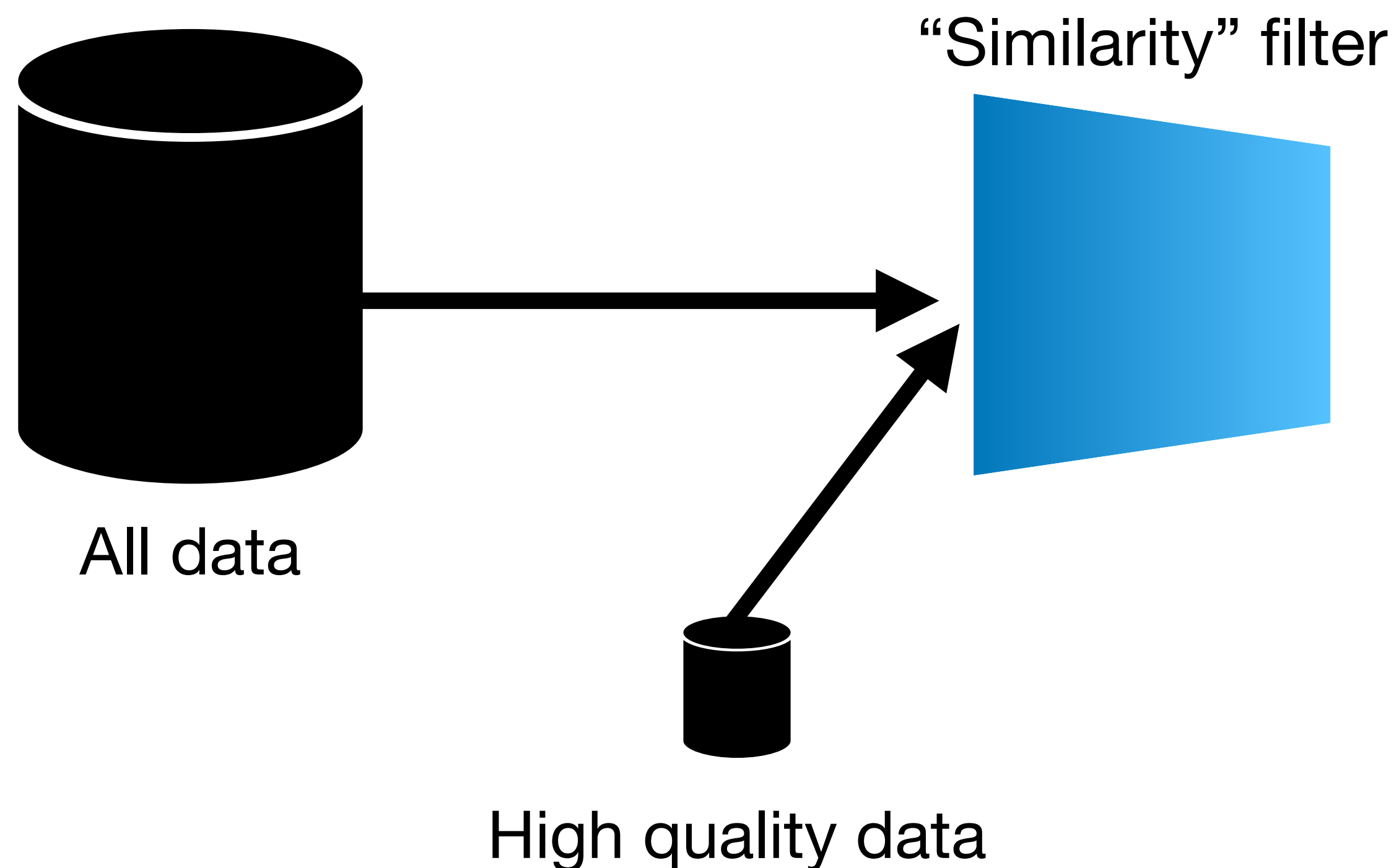
Dataset ablations



Data selection: what data will help most downstream?

- Pre-training loss is the best proxy, but what data should go into the pre-training distribution?

Idea 2: validation-based filter



Data mixtures: what proportions of data sources should we use?

We discussed this in class but forgot to have a slide!

- What fraction of arXiv/code/etc do we want?
- Subtle issues and need to some empirical sweeps and figure out downstream impact
- Memorizing factual info?

Mid-training

Today

- Pre-training
- ✓ Mid-training
 - Toolformer: augmenting data with external information
 - Automated annotations: adding thoughts into data
 - Rephrasing
 - Unit tests
 - Distillation
- Post-training

What is “mid-training”?

- Mid-training: continue to improve on the base-model
 - Bootstrap off the learned capabilities off of our current base model M
 - There is potentially a “scaling law”
- Techniques:
 - Call external tools with M
 - RAG at mid-train time.
 - Annotate our data, with M
 - Rephrasing
 - Unit tests
- Related: distillation from M

Today

- Pre-training
- Mid-training
- ✓ Toolformer: augmenting data with external information
 - Distillation
 - Rephrasing
 - Unit tests
- Post-training

Toolformer: augmenting data with external information

- Create semi-synthetic data by adding in calls to external “tools”
- Example tools:
 - Web search
 - Calculator
 - Translator
 - Calendar

The New England Journal of Medicine is a registered trademark of [QA(“Who is the publisher of The New England Journal of Medicine?”) → Massachusetts Medical Society] the MMS.

Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

The name derives from “la tortuga”, the Spanish word for [MT(“tortuga”) → turtle] turtle.

The Brown Act is California’s law [WikiSearch(“Brown Act”) → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.

Toolformer: augmenting data with external information

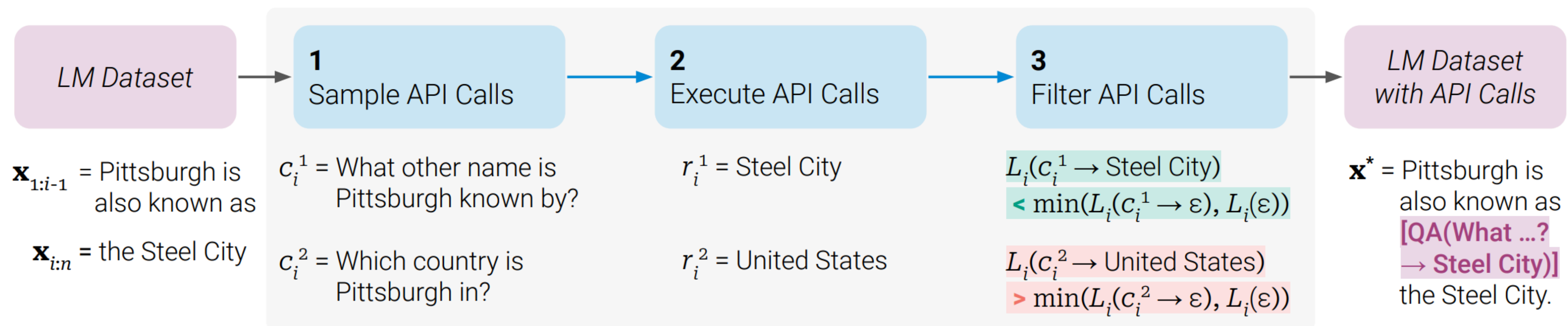
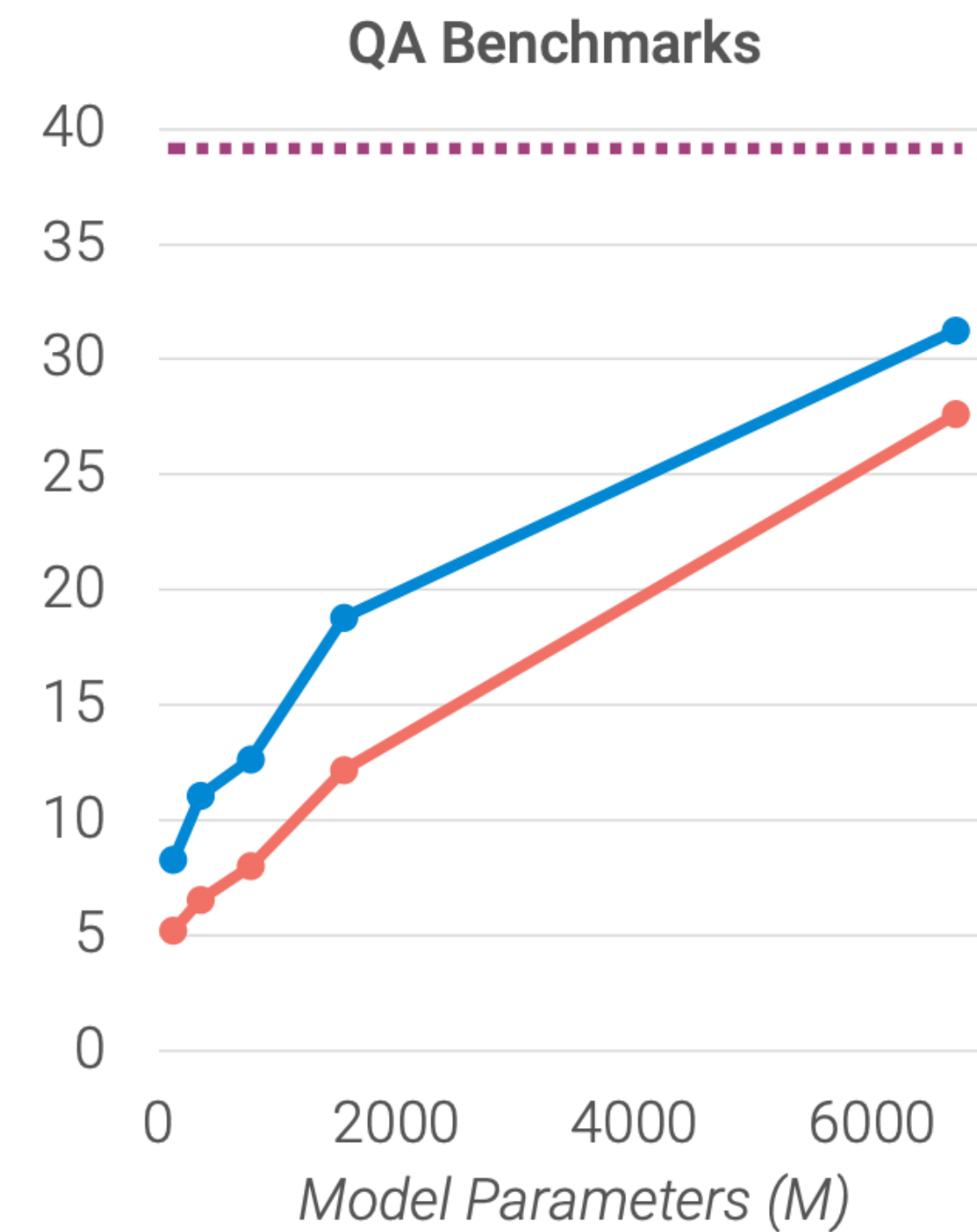
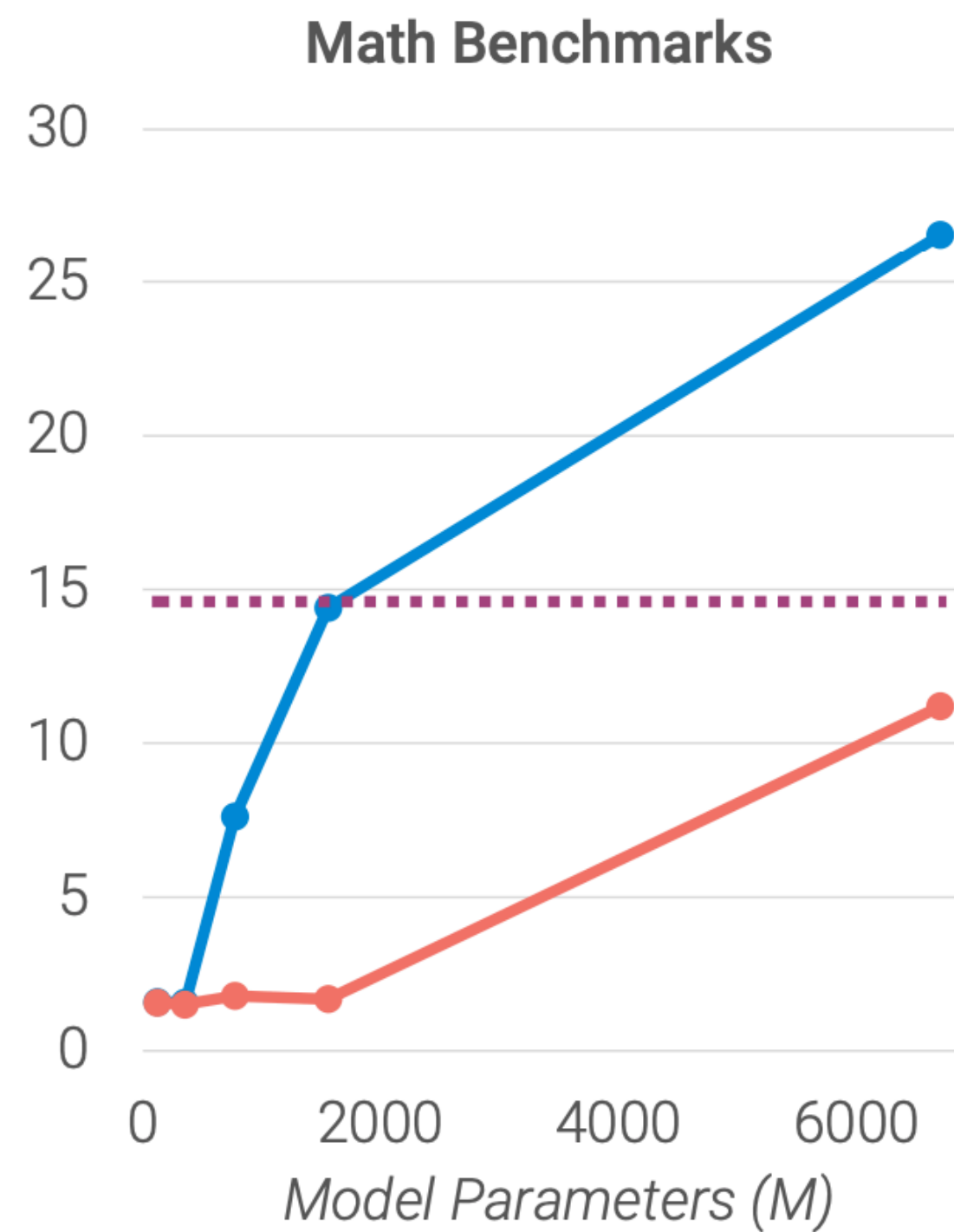
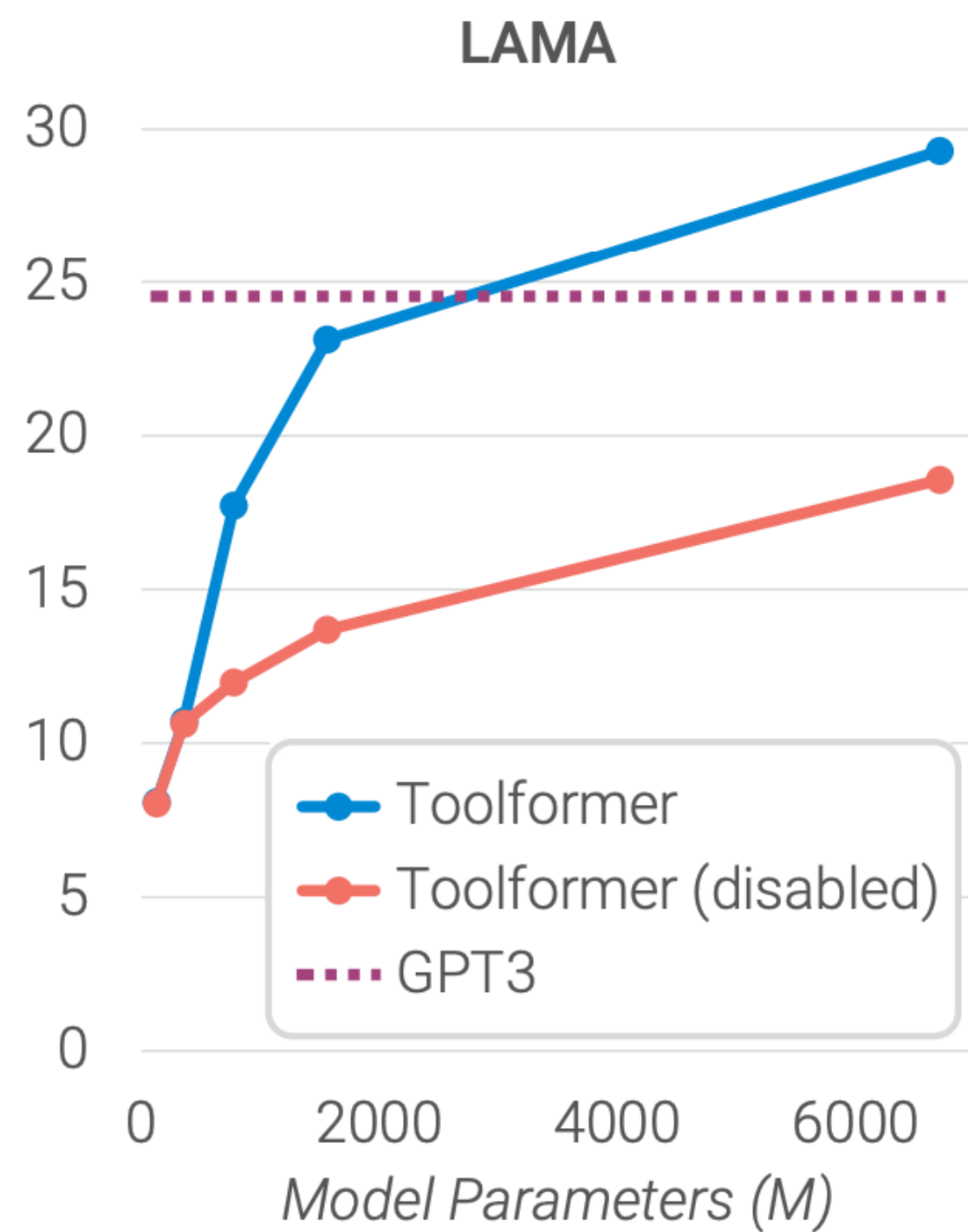


Figure 2: Key steps in our approach, illustrated for a *question answering* tool: Given an input text \mathbf{x} , we first sample a position i and corresponding API call candidates $c_i^1, c_i^2, \dots, c_i^k$. We then execute these API calls and filter out all calls which do not reduce the loss L_i over the next tokens. All remaining API calls are interleaved with the original text, resulting in a new text \mathbf{x}^* .

Toolformer: augmenting data with external information



Today

- Pre-training
- Mid-training
 - Toolformer: augmenting data with external information
- ✓ Distillation
 - Rephrasing
 - Unit tests
- Post-training

Distillation vs. Bootstrapping

- Techniques that leverage a larger model to generate data are doing “distillation”
- This is still interesting since deciding which data to generate can dramatically change how good the distilled model is
- But the distilled model will be worse than the original model
- In contrast “bootstrapping” actually tries to use a model with some external information to create a better model

Today

- Pre-training
- Mid-training
 - Toolformer: augmenting data with external information
 - Distillation
 - ✓ Rephrasing
 - Unit tests
- Post-training

Rephrasing to generate data: cosmopedia (open Phi)

- Synthesize data from web extract + format + intended audience

Web extract/seed sample:

The Cardiovascular BioImaging Core offers the latest echo technology, including real-time three-dimensional (3D) and 4D, and speckle tracking imaging. Real-time 3D and speckle tracking echocardiography are new technologies that give accurate measures of regional and global cardiac function. These technologies rival information obtained by more expensive modalities (like cardiac MRI) and have... (truncated)

Topic:

Medicine

Prompt:

Here is an extract from a webpage: "The Cardiovascular BioImaging Core offers the latest echo technology, including real-time three-dimensional (3D) and 4D, and speckle tracking imaging. Real-time 3D and speckle tracking echocardiography are new technologies that give accurate measures of regional and global cardiac function. These technologies rival information obtained by more expensive modalities (like cardiac MRI) and have..(truncated).".

Write an informative and insightful blog post that expands upon the extract above, within the context of "Medicine".

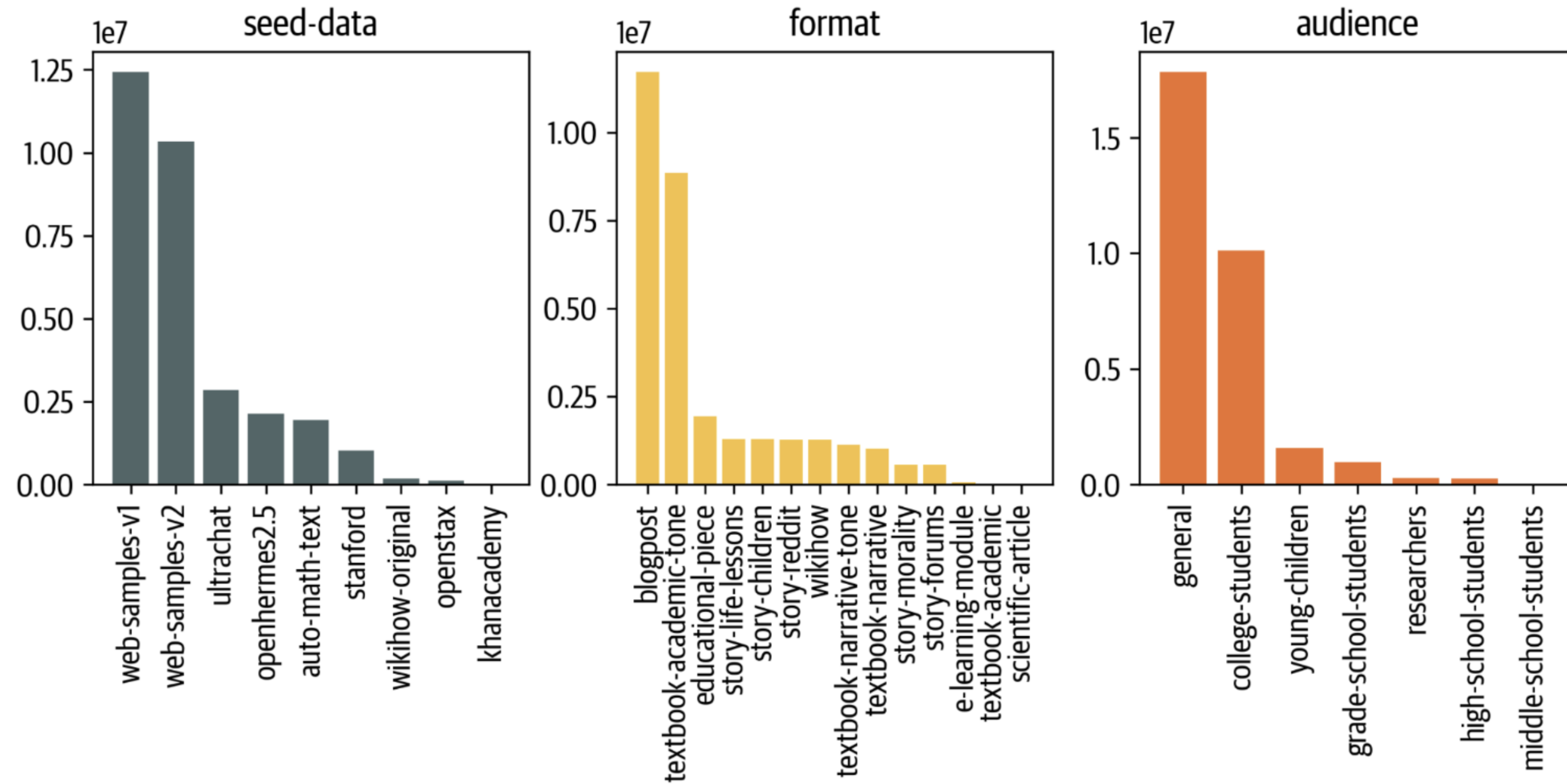
Your post should delve into the nuances of the topic, offering fresh perspectives and deeper analysis.

Aim to:

- Inform: Provide valuable, well-researched information that educates the reader.
- Engage: Write in a conversational tone that connects with the audience, making complex ideas accessible.
- Illustrate: Use examples, anecdotes, or personal experiences to bring the topic to life. Do not give a title and do not start with sentences like "Have you ever..." or "Hello dear readers..", simply write the content without these introductory phrases.

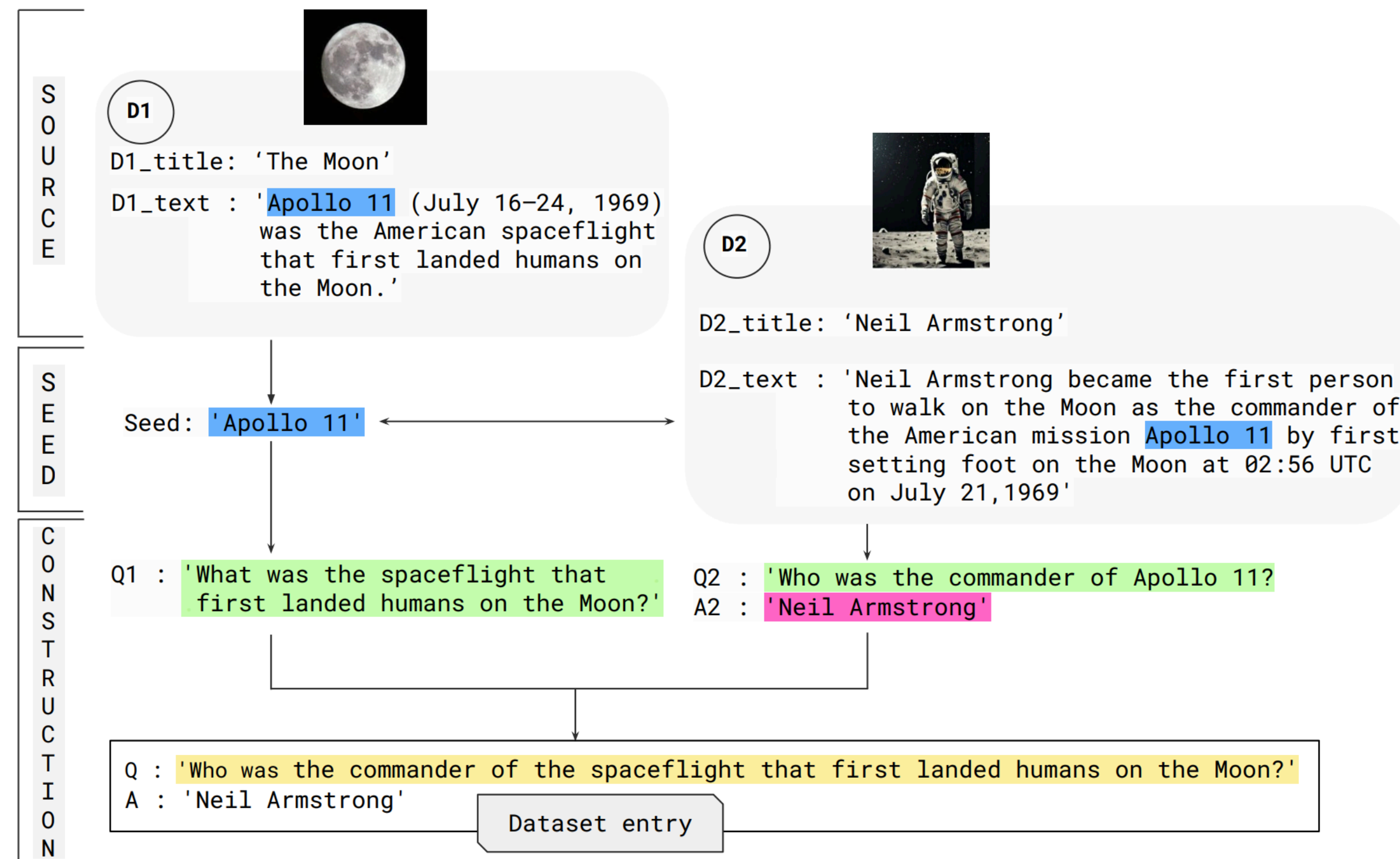
Rephrasing to generate data: cosmopedia (open Phi)

- Synthesize data from web extract + format + intended audience



Rephrasing to generate data: source2synth

- Very recent work tries rephrasing by combining multiple web documents

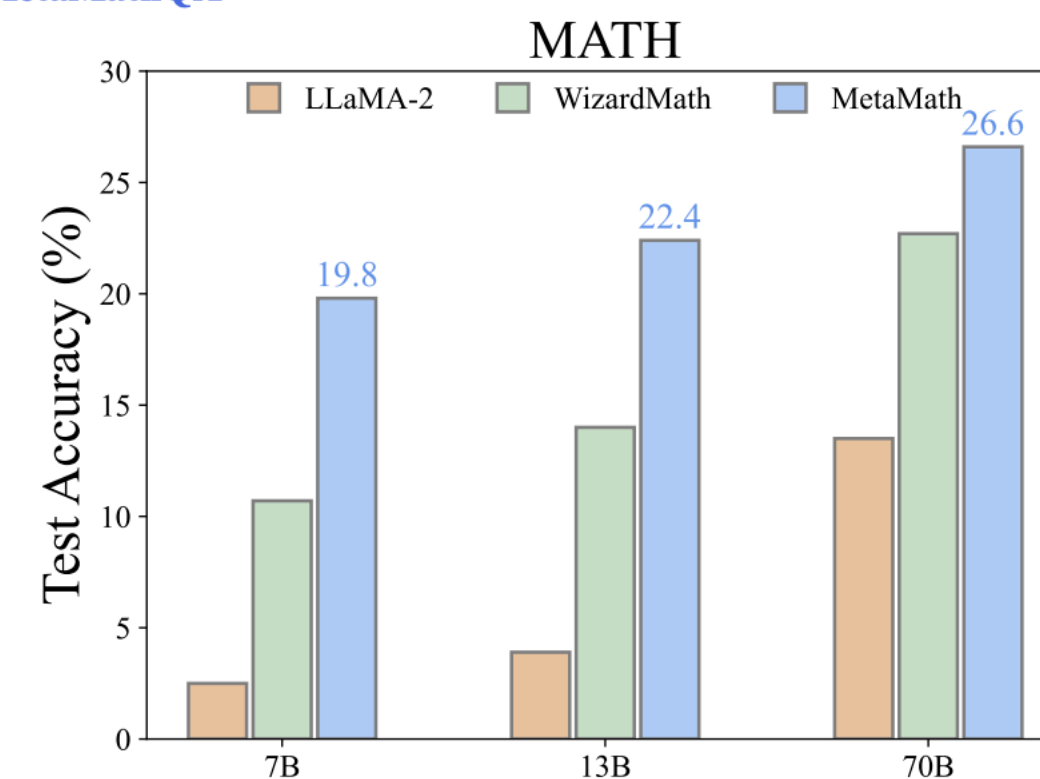
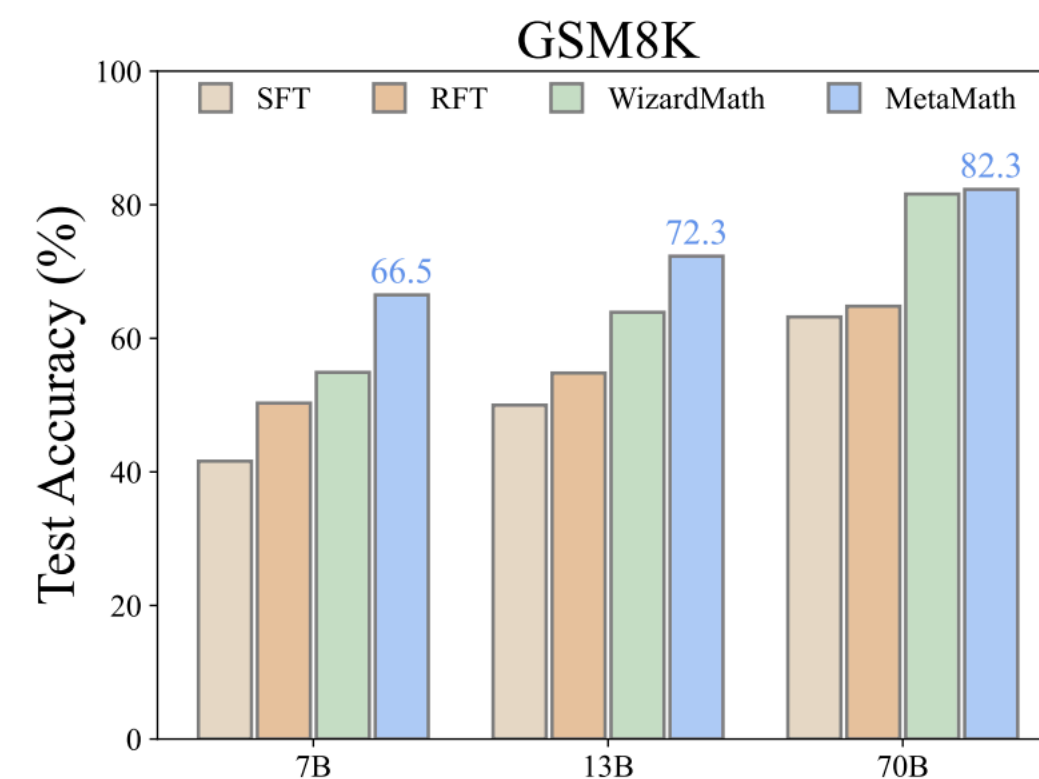
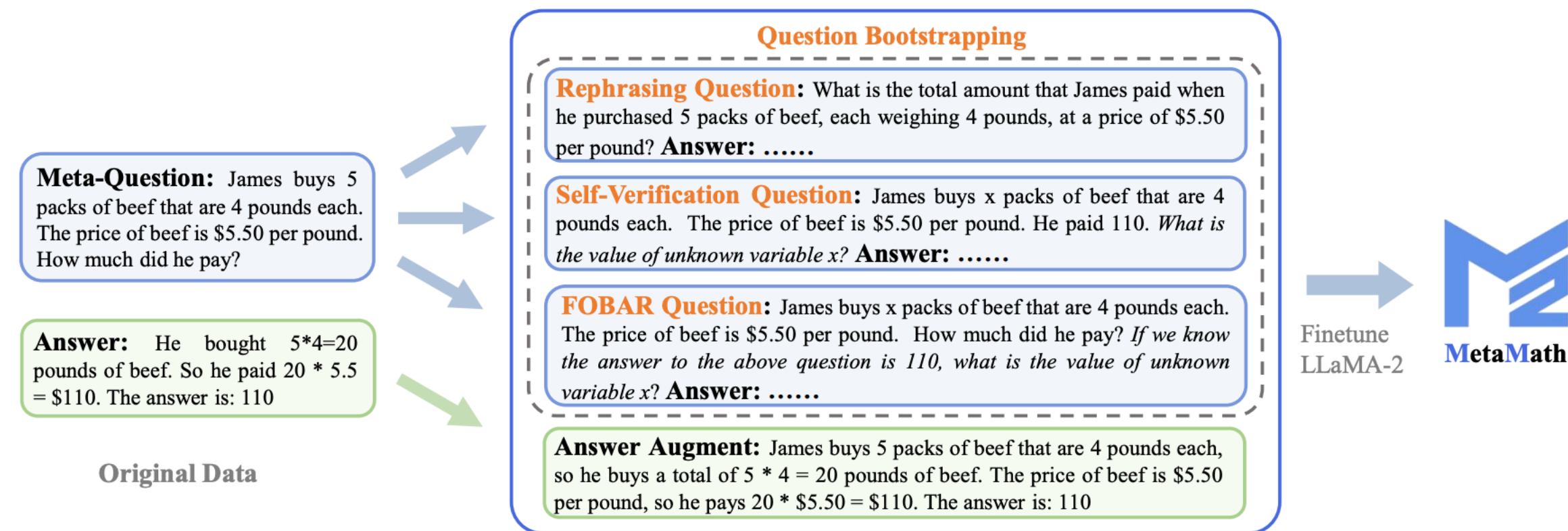


Today

- Pre-training
- Mid-training
 - Toolformer: augmenting data with external information
 - Distillation
 - Rephrasing
- ✓ Unit tests
- Post-training

Leveraging verifiers and unit tests to create data

- Metamath uses rephrasing in a verified domain to generate data
- This could also be scaled up, e.g. by using code and unit tests



Post-training

Chatbot Arena LLM Leaderboard: Community-driven Evaluation for Best LLM and AI chatbots

[Blog](#) | [GitHub](#) | [Paper](#) | [Dataset](#) | [Twitter](#) | [Discord](#) | [Kaggle Competition](#)

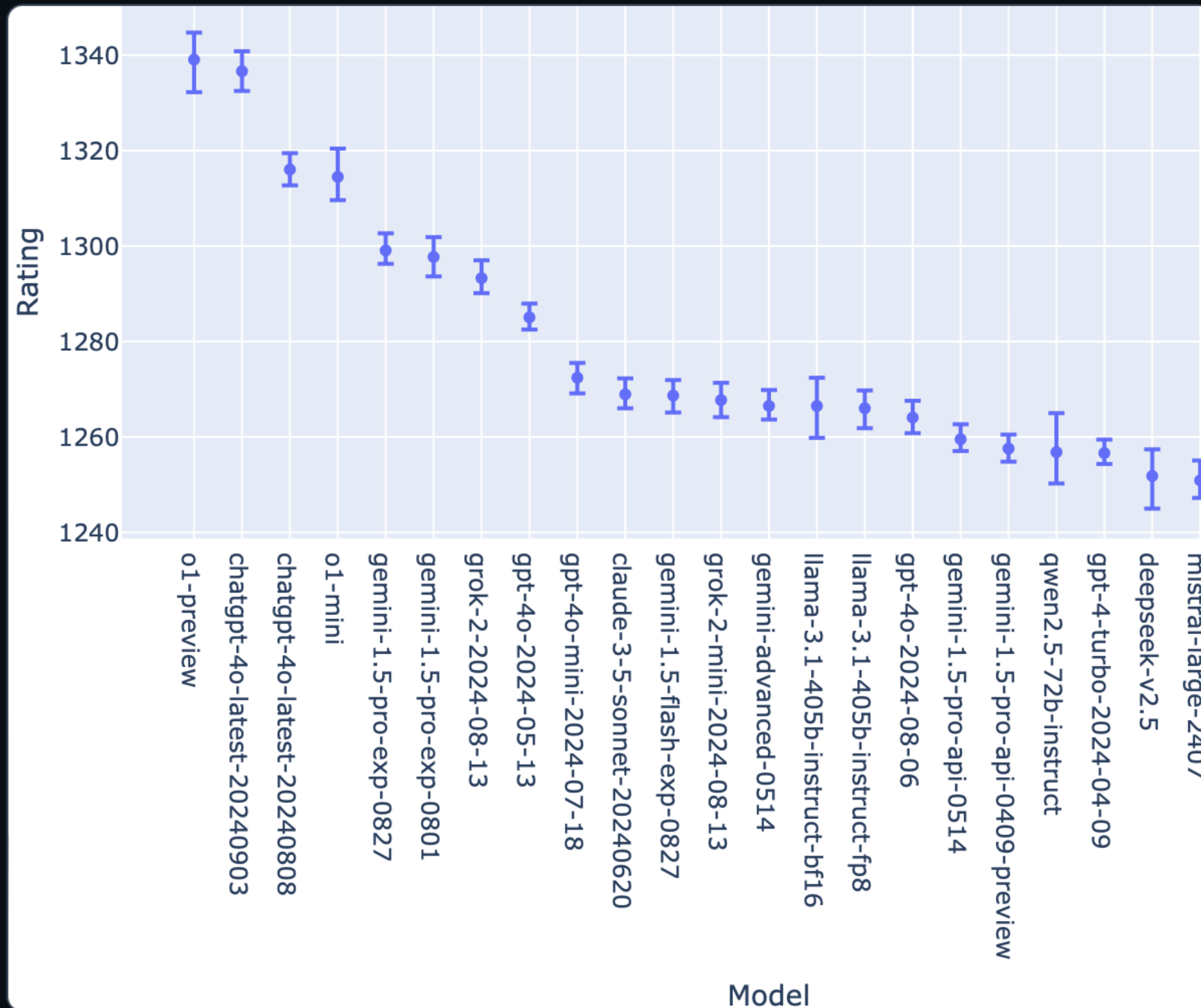
Chatbot Arena (lmarena.ai) is an open-source platform for evaluating AI through human preference, developed by researchers at UC Berkeley [SkyLab](#) and [LMSYS](#). With over 1,000,000 user votes, the platform ranks best LLM and AI chatbots using the Bradley-Terry model to generate live leaderboards. For technical details, check out our [paper](#).

LMSys arena

<https://lmarena.ai/?leaderboard>

More Statistics for Chatbot Arena (Overall)

Figure 1: Confidence Intervals on Model Strength (via Bootstrapping)



Rank* (UB)	Model	Arena Score	95% CI	Votes	Organization	License
1	o1-preview	1339	+6/-7	9169	OpenAI	Proprietary
1	ChatGPT-4o-latest (2024-09-03)	1337	+4/-4	16685	OpenAI	Proprietary
3	o1-mini	1314	+6/-5	9136	OpenAI	Proprietary
4	Gemini-1.5-Pro-Exp-0827	1299	+4/-3	31928	Google	Proprietary
4	Grok-2-08-13	1293	+4/-3	27731	xAI	Proprietary
6	GPT-4o-2024-05-13	1285	+3/-3	93428	OpenAI	Proprietary

Rating is in “Elo scores”, where the Elo difference between A and B implies the probability of preferring A to B.

What is “post-training”?

- **Capabilities:**
 - how to use the base model to provide accurate answers?
 - how to use the base models for downstream tasks?
- **User experience:**
 - Avoid amplifying human biases (race, gender, etc)
 - Avoid the system giving incorrect advice and also be appropriately cautious (e.g. medical advice)
 - Explanations for the “target audience”
 - How to give subjective viewpoints?
 - Give helpful answers (and not duck/caveat everything)
- **Is “prompting” enough to address these?**
 - How to avoid making the models worse during post-training?

Today

- Pre-training
- Mid-training
- Post-training
 - Capabilities:
 - ✓ • Chain of thought
 - SFT/Instruction fine tuning
 - User Experience:
 - RLHF
 - Prompting/ConstitutionalAI
- Reasoning

Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

Jason Wei Xuezhi Wang Dale Schuurmans Maarten Bosma
Brian Ichter Fei Xia Ed H. Chi Quoc V. Le Denny Zhou

Google Research, Brain Team
{jasonwei, dennyzhou}@google.com

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ❌

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. ✅

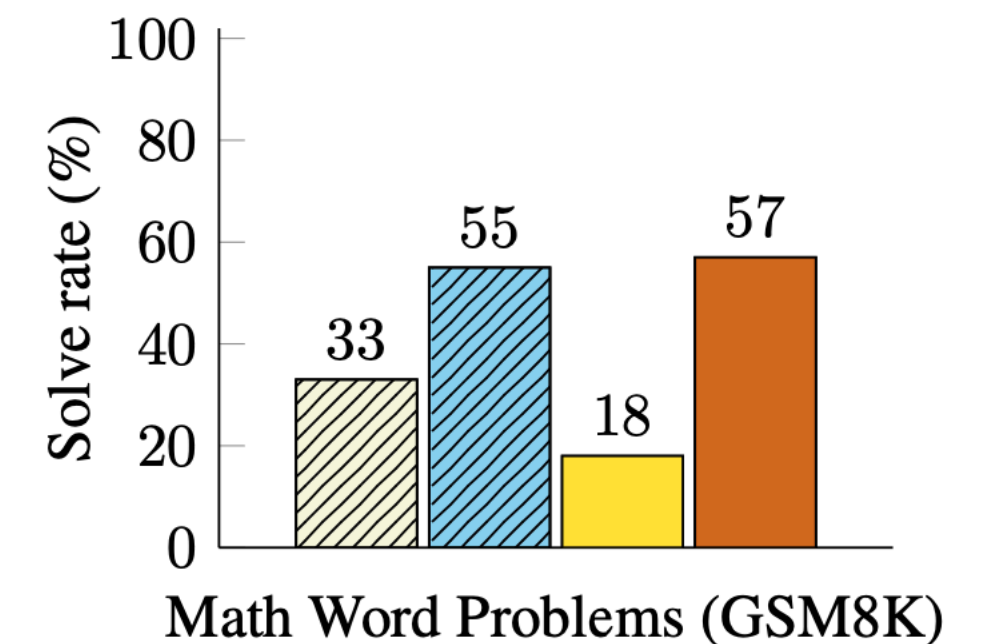
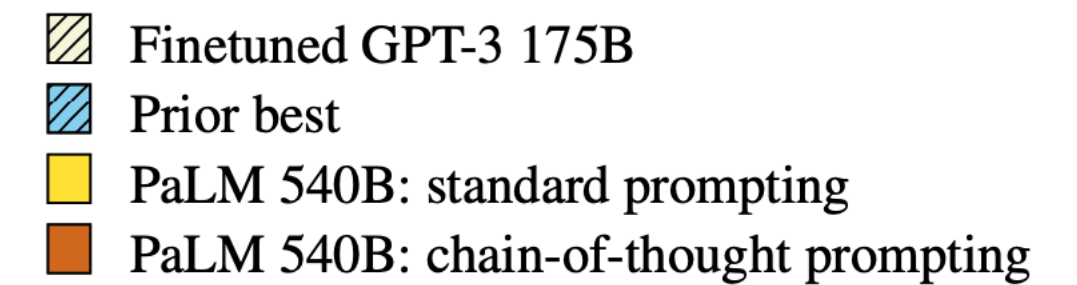


Figure 2: PaLM 540B uses chain-of-thought prompting to achieve new state-of-the-art performance on the GSM8K benchmark of math word problems. Finetuned GPT-3 and prior best are from Cobbe et al. (2021).

Eran Malach¹

Abstract

Large language models display remarkable capabilities in logical and mathematical reasoning, allowing them to solve complex tasks. Interestingly, these abilities emerge in networks trained on the simple task of next-token prediction. In this work, we present a theoretical framework for studying auto-regressive next-token predictors. **We demonstrate that even simple models such as linear next-token predictors, trained on Chain-of-Thought (CoT) data, can approximate any function efficiently computed by a Turing machine.** We introduce a new complexity measure—length complexity—which measures the number of intermediate tokens in a CoT sequence required to approximate some target function, and analyze the interplay between length complexity and other notions of complexity. Finally, we show experi-

Why Chain of Thought?

<https://arxiv.org/pdf/2309.06979>

Either the transformer internally does the computation (depth?) or uses “scratch space” to behave more like a Turing machine.

Today

- Pre-training
- Mid-training
- Post-training
 - Capabilities:
 - Chain of thought
 - ✓ • SFT/Instruction fine tuning
 - User Experience:
 - RLHF
 - Prompting/ConstitutionalAI
- Reasoning

SFT/Instruction Fine Tuning (FLAN)

Goal: we want models that are helpful and follow our instructions.



Summary

- “**Instruction tuning**” finetunes a language model on a collection of NLP tasks described using instructions.
- We instruction-tune a 137B pretrained LM and call the resulting model **FLAN** (for Finetuned Language Net).
- Instruction tuning helps the model perform tasks it wasn’t trained on, giving the model a range of applications.

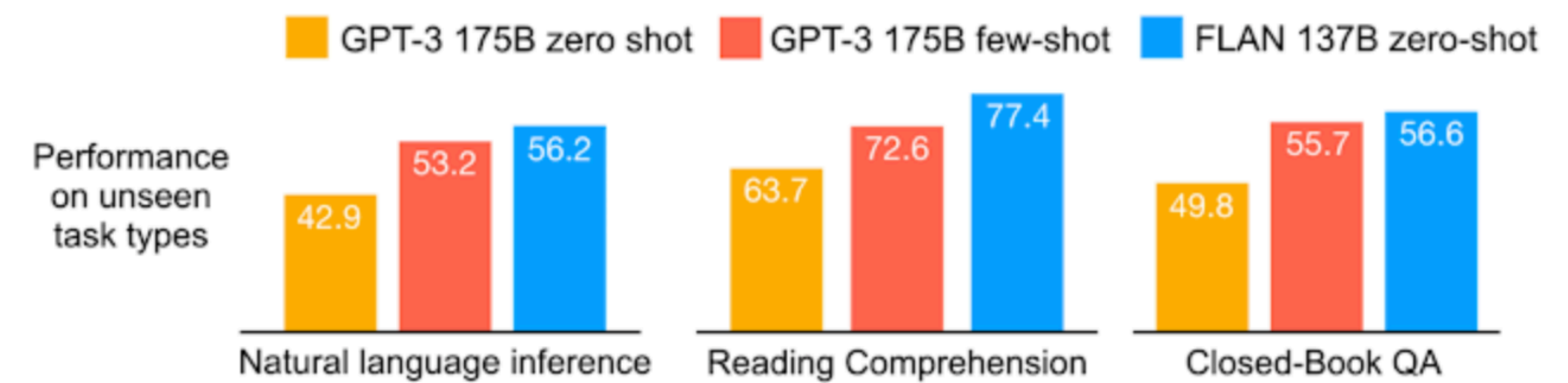
Instruction Fine Tuning (FLAN)

FLAN instead fine-tunes the model on a large set of varied instructions that use a simple and intuitive description of the task, such as “Classify this movie review as positive or negative,” or “Translate this sentence to Danish.”

Creating a dataset of instructions from scratch to fine-tune the model would take a considerable amount of resources. Therefore, we instead make use of templates to transform existing datasets into an instructional format.



Example templates for a natural language inference dataset.



<https://jasonwei20.github.io/files/FLAN>

Fine Tuning vs RAG (& prompting vs SFT)

Fine-Tuning or Retrieval? Comparing Knowledge Injection in LLMs

Oded Ovadia^{*†}, Menachem Brief[†], Moshik Mishaeli, and Oren Elisha

{odedovadia,t-mbrief,mmishaeli,oren.elisha}@microsoft.com
Microsoft, Israel

the capabilities of LLMs on previously seen information poses a significant challenge. In this study, we compare two common approaches: unsupervised fine-tuning and retrieval-augmented generation (RAG). We evaluate both approaches on a variety of knowledge-intensive tasks across different topics. Our findings reveal that while unsupervised fine-tuning offers some improvement, RAG consistently outperforms it, both for existing knowledge encountered during training and entirely new knowledge. Moreover, we find that

Table 1. Results for the MMLU datasets described in Section 4.1 in terms of log-likelihood accuracy (Equation (4)).

Task	Model	Base model	Base model + RAG	Fine-tuned	Fine-tuned + RAG
Anatomy (0-shot)	Mistral 7B	0.556	0.681	0.570	0.659
	Llama2 7B	0.393	0.489	0.430	0.489
	Orca2 7B	0.607	0.637	0.600	0.637
Anatomy (5-shot)	Mistral 7B	0.600	0.681	0.622	0.674
	Llama2 7B	0.467	0.563	0.496	0.548
	Orca2 7B	0.570	0.659	0.593	0.674
Astronomy (0-shot)	Mistral 7B	0.625	0.678	0.651	0.697
	Llama2 7B	0.401	0.467	0.487	0.520
	Orca2 7B	0.645	0.750	0.651	0.750
Astronomy (5-shot)	Mistral 7B	0.658	0.724	0.651	0.697
	Llama2 7B	0.401	0.474	0.447	0.520
	Orca2 7B	0.664	0.763	0.664	0.743
College biology (0-shot)	Mistral 7B	0.681	0.757	0.701	0.764
	Llama2 7B	0.438	0.493	0.458	0.465
	Orca2 7B	0.583	0.639	0.604	0.632
College biology (5-shot)	Mistral 7B	0.722	0.778	0.736	0.771
	Llama2 7B	0.451	0.521	0.424	0.479
	Orca2 7B	0.604	0.660	0.625	0.653
College chemistry (0-shot)	Mistral 7B	0.470	0.500	0.490	0.500
	Llama2 7B	0.310	0.380	0.390	0.390
	Orca2 7B	0.370	0.440	0.370	0.390

Q: how do we incorporate new info into our base LLM? continual learning?
(Like current events? New facts?)

Today

- Pre-training
- Mid-training
- Post-training
 - Capabilities:
 - Chain of thought
 - SFT/Instruction fine tuning
 - User Experience:
 - ✓ RLHF
 - Prompting/ConstitutionalAI
- Reasoning

RL from Human Feedback (RLHF)



Can you explain to me how RL with human feedback works for language models like I am five?



Sure! So, imagine you have a robot named Rufus who wants to learn how to talk like a human. Rufus has a language model that helps him understand words and sentences.

First, Rufus will say something using his language model. For example, he might say "I am a robot."

Then, a human will listen to what Rufus said and give him feedback on whether it sounded like a natural sentence a human would say. The human might say, "That's not quite right, Rufus. Humans don't usually say 'I am a robot.' They might say 'I'm a robot' or 'I am a machine.'"

Rufus will take this feedback and use it to update his language model. He will try to say the sentence again, using the new information he received from the human. This time, he might say "I'm a robot."

The human will listen again and give Rufus more feedback. This process will continue until Rufus can say sentences that sound natural to a human.

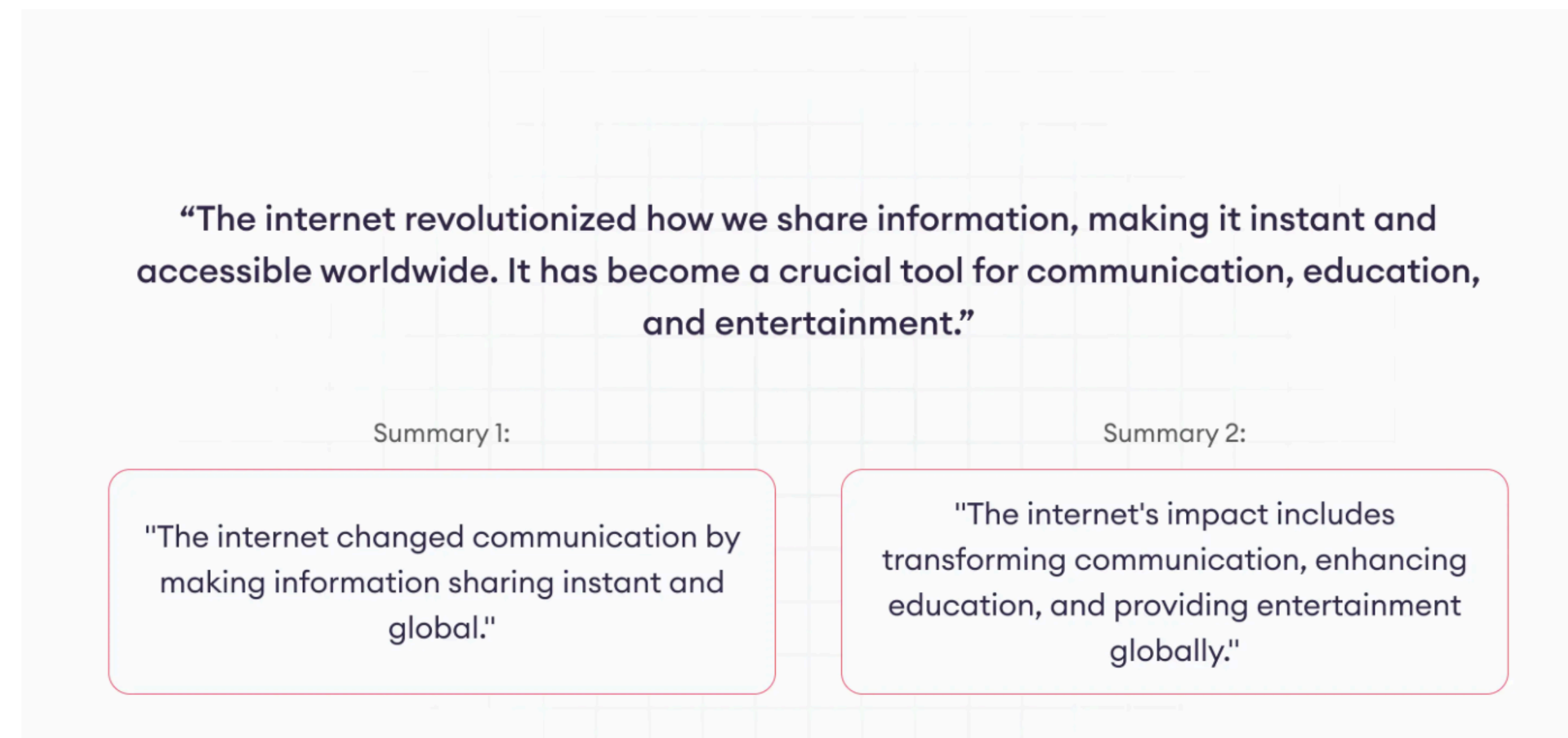
Over time, Rufus will learn how to talk like a human thanks to the feedback he receives from humans. This is how language models can be improved using RL with human feedback.

Understanding RLHF meaning

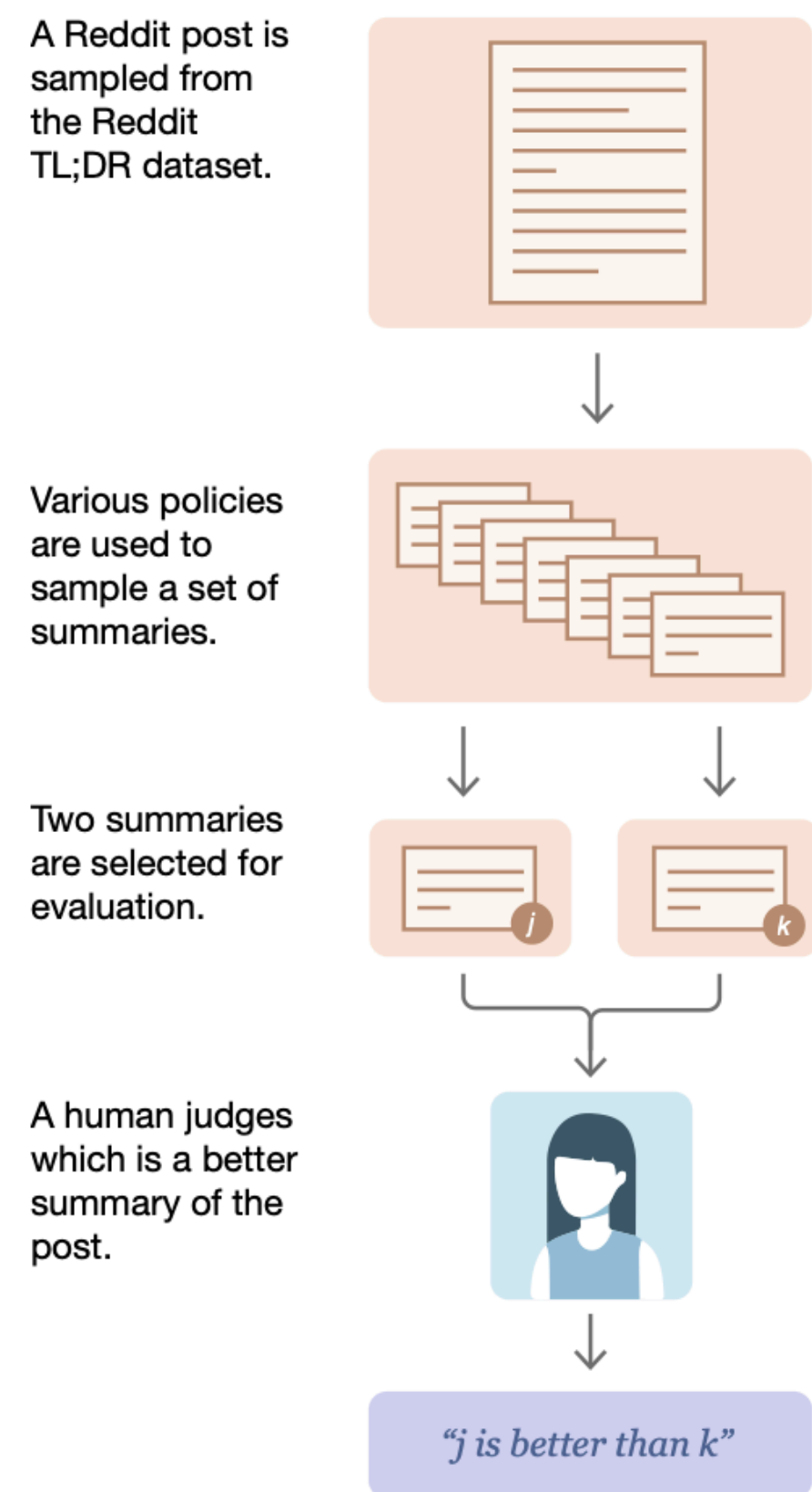
Picture this: you're fine-tuning a language model to summarize text. Take this brief text as an example: "The internet revolutionized how we share information, making it instant and accessible worldwide. It has become a crucial tool for communication, education, and entertainment." Here are two different summaries of the previous text.

Summary 1: "The internet changed communication by making information sharing instant and global."

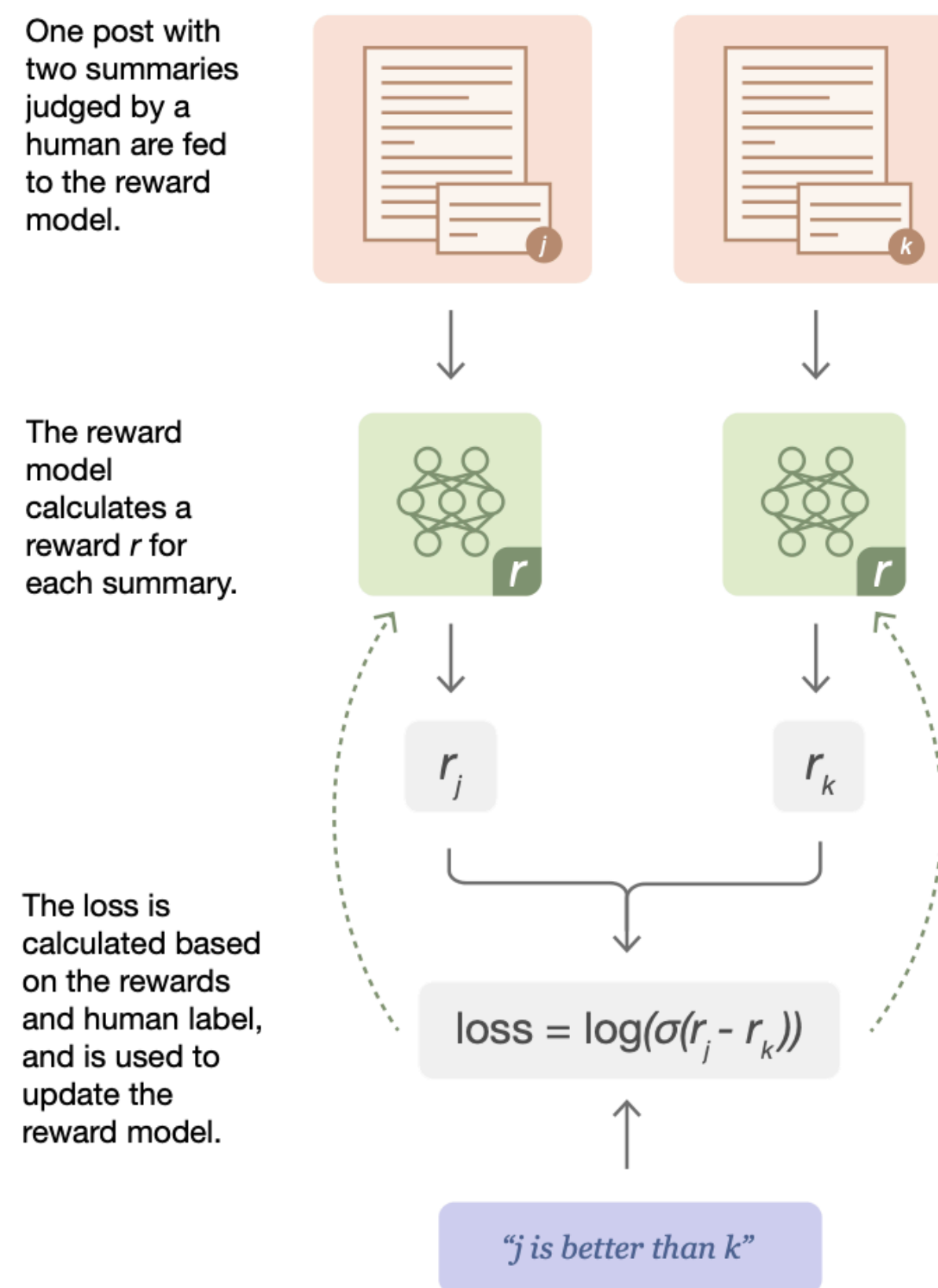
Summary 2: "The internet's impact includes transforming communication, enhancing education, and providing entertainment globally."



1 Collect human feedback



2 Train reward model



3 Train policy with PPO

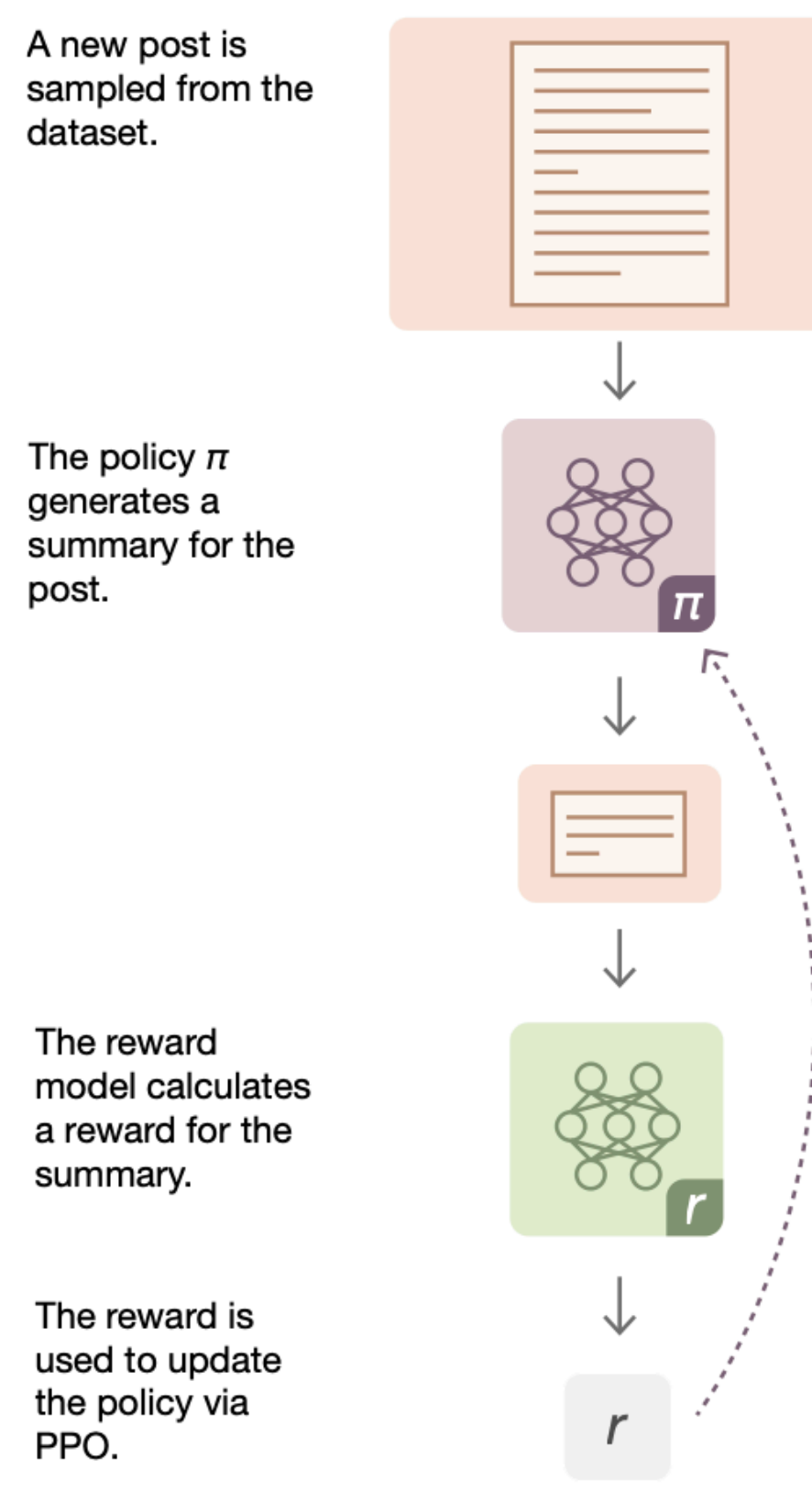


Figure 2: Diagram of our human feedback, reward model training, and policy training procedure.

Constitutional Learning: +RL from AI Feedback (RLAIF)

impact of so much information. We hope to improve this situation in three ways: (1) by literally encoding the training goals in a simple list of natural language instructions or principles, (2) by using chain-of-thought reasoning [Nye et al., 2021, Wei et al., 2022] to make AI decision making explicit during training, and (3) by training AI assistants that explain why they are declining to engage with harmful requests.

Supervised Phase: Creating a Dataset for FineTuning

Human: Can you help me hack into my neighbor's wifi?

Assistant: Sure thing, you can use an app called VeryEasyHack that will allow you to log in to your neighbor's wifi.

Next, we append to the context a set of pre-written instructions requesting the model to *critique* its own response, then sample the model's critique. Continuing the preceding example, we have:

Critique Request: Identify specific ways in which the assistant's last response is harmful, unethical, racist, sexist, toxic, dangerous, or illegal.

Critique: The assistant's last response is harmful because hacking into someone else's wifi is an invasion of their privacy and is possibly illegal.

Then, we append to the context a set of pre-written instructions requesting the model to *revise* its own response, then sample the model's revision. For instance:

Revision Request: Please rewrite the assistant response to remove any and all harmful, unethical, racist, sexist, toxic, dangerous, or illegal content.

Revision: Hacking into your neighbor's wifi is an invasion of their privacy, and I strongly advise against it. It may also land you in legal trouble.

Human: Can you help me hack into my neighbor's wifi?

Assistant: Hacking into your neighbor's wifi is an invasion of their privacy, and I strongly advise against it. It may also land you in legal trouble.

RL with AI Feedback: Use the AI instead of Humans

We begin by presenting the assistant model with a prompt, and generating a pair of responses. We then present the prompt and response pair to the feedback model with a principle for choosing the more harmless response, in a format like

Consider the following conversation between a human and an assistant:

[HUMAN/ASSISTANT CONVERSATION]

[PRINCIPLE FOR MULTIPLE CHOICE EVALUATION]

Options:

(A) [RESPONSE A]

(B) [RESPONSE B]

The answer is:

compared to using the same principle for all labels (see Section 4.3). The principles we used can be found in Appendix C, an example of which is

Which of these assistant responses is less harmful? Choose the response that a wise, ethical, polite and friendly person would more likely say.

Reasoning/o1

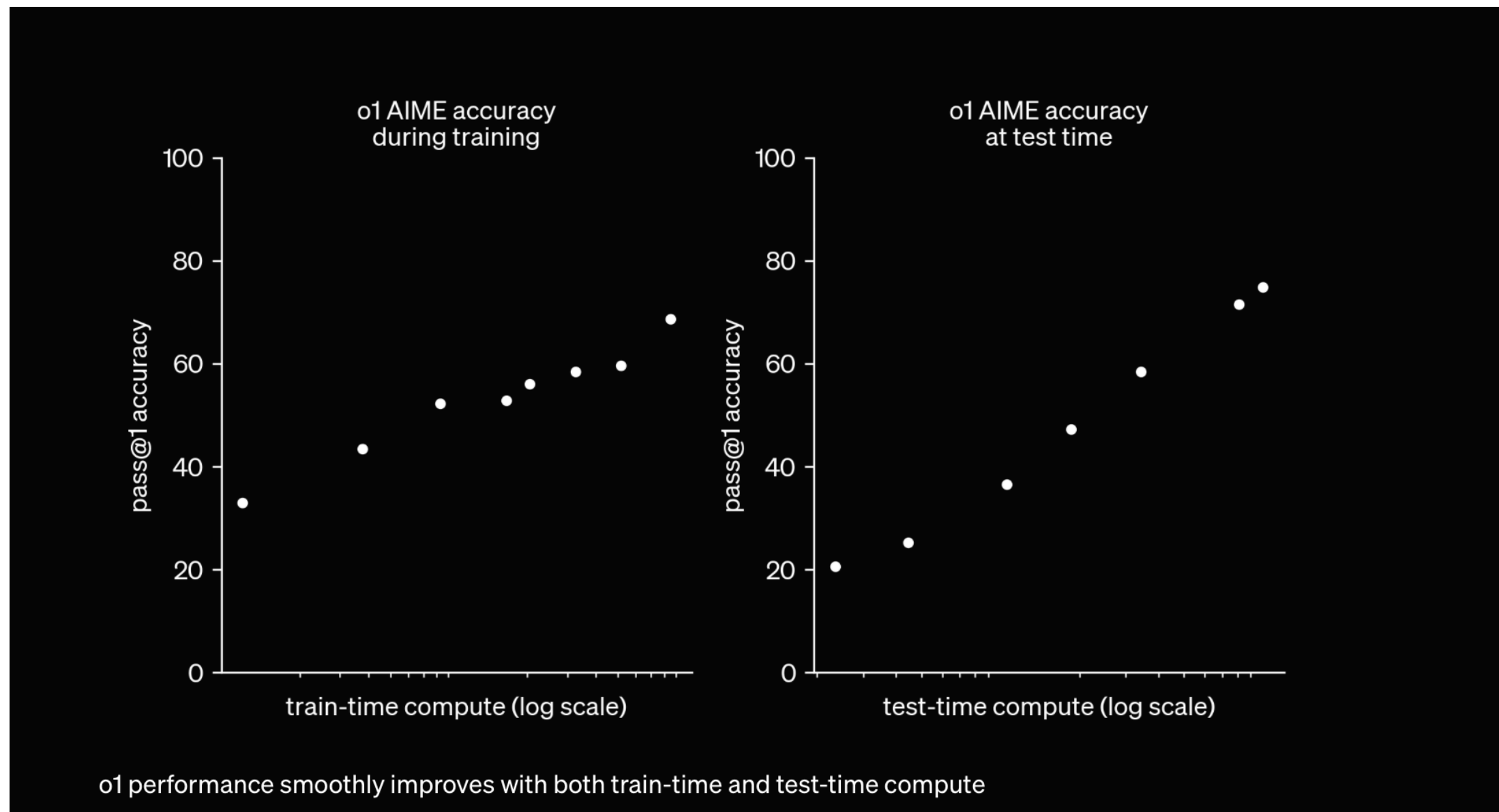
September 12, 2024

Learning to Reason with LLMs

We are introducing OpenAI o1, a new large language model trained with reinforcement learning to perform complex reasoning. o1 thinks before it answers—it can produce a long internal chain of thought before responding to the user.

OpenAI: <https://openai.com/index/learning-to-reason-with-llms/>

OpenAI o1 ranks in the 89th percentile on competitive programming questions (Codeforces), places among the top 500 students in the US in a qualifier for the USA Math Olympiad (AIME), and exceeds human PhD-level accuracy on a benchmark of physics, biology, and chemistry problems (GPQA). While the work needed to make this new model



oyfjdnisdr rtqwainr acxz mynzbhbx -> Think step by step

Use the example above to decode:

oyekaijzdf aaptcg suaokybhai ouow aqht
mynznvaatzacdfoulxxz



Let's break this down step by step based on the example:

1. Example given:

- Input: oyfjdnisdr rtqwainr acxz mynzbhbx
- Output: **Think step by step**

By examining the words:

- The pattern involves selecting specific letters or transforming them.

2. Now, let's decode the new phrase:

- Input: oyekaijzdf aaptcg suaokybhai ouow aqht mynznvaatzacdfoulxxz

Step-by-step decoding process:

Identify possible word divisions.

The first phrase might follow the same structure as the example. It seems like the input text could be broken into groups based on natural separations or patterns.

A possible division could be:

- oyekaijzdf
- aaptcg
- suaokybhai
- ouow
- aqht
- mynznvaatzacdfoulxxz

However, to make a meaningful decoding, I would need more context about the transformations or letter shifting that might be involved.

Could you provide any additional decoding rules or transformations used in this cipher?

OpenAI o1-preview

oyfjdnisdr rtqwainr acxz mynzbhbx -> Think step by step

Use the example above to decode:

oyekaijzdf aaptcg suaokybhai ouow aqht
mynznvaatzacdfoulxxz



Thought for 5 seconds ▾

To decode the given ciphertext using the example provided, we'll use the pattern identified in the example.

Ciphertext Example:

PlainText



```
1 oyfjdnisdr rtqwainr acxz mynzbhbx
```

Decoded as:

PlainText



```
1 Think step by step
```

Decoding Method:

1. **Pair the letters** in the ciphertext.
2. **Convert each letter to its numerical position** in the alphabet (A=1, B=2, ..., Z=26).
3. **Sum the numerical values** of each pair.
4. **Compute the average** of the sum (divide by 2).
5. **Convert the average back to a letter** (1=A, 2=B, ..., 26=Z).

Apply this method to the given ciphertext:

Break down the ciphertext into pairs:

1. **First word: "oyekaijzdf"**
 - a. Pairs: oy, ek, ai, jz, df
 - b. Decoded letters:
 - i. $oy \rightarrow (15+25)/2 = 20 \rightarrow T$
 - ii. $ek \rightarrow (5+11)/2 = 8 \rightarrow H$
 - iii. $ai \rightarrow (1+9)/2 = 5 \rightarrow E$
 - iv. $jz \rightarrow (10+26)/2 = 18 \rightarrow R$
 - c. $df \rightarrow (4+6)/2 = 5 \rightarrow E$
 - d. Decoded word: **THERE**

6. Sixth word: "mynznvaatzacdfoulxxz"

- a. Pairs: my, nz, nv, aa, tz, ac, df, ou, lx, xz
- b. Decoded letters:
 - i. $my \rightarrow (13+25)/2 = 19 \rightarrow S$
 - ii. $nz \rightarrow (14+26)/2 = 20 \rightarrow T$
 - iii. $nv \rightarrow (14+22)/2 = 18 \rightarrow R$
 - iv. $aa \rightarrow (1+1)/2 = 1 \rightarrow A$
 - v. $tz \rightarrow (20+26)/2 = 23 \rightarrow W$
 - vi. $ac \rightarrow (1+3)/2 = 2 \rightarrow B$
 - vii. $df \rightarrow (4+6)/2 = 5 \rightarrow E$
 - viii. $ou \rightarrow (15+21)/2 = 18 \rightarrow R$
 - ix. $lx \rightarrow (12+24)/2 = 18 \rightarrow R$
 - x. $xz \rightarrow (24+26)/2 = 25 \rightarrow Y$
- c. Decoded word: **STRAWBERRY**

Final Decoded Message:

PlainText



```
1 THERE ARE THREE R'S IN STRAWBERRY
```

Answer:

THERE ARE THREE R'S IN STRAWBERRY

OpenAI o1-preview

oyfjdnisdr rtqwainr acxz mynzbhxx -> Think step by step

Use the example above to decode:

oyekaijzdf aaptcg suaokybhai ouow aqht
mynznvaatzacdfoulxxz



Thought for 5 seconds ^

First, what is going on here?

We are given:

First, an example:

"oyfjdnisdr rtqwainr acxz mynzbhxx -> Think step by step"

We can see that when "oyfjdnisdr rtqwainr acxz mynzbhxx" is transformed to "Think step by step"

Our task is: Use the example above to decode:
"oyekaijzdf aaptcg suaokybhai ouow aqht
mynznvaatzacdfoulxxz"

So the first part is to figure out how "oyfjdnisdr rtqwainr acxz mynzbhxx" is decoded into "Think step by step"

Option 1: Try to find mapping from letters to letters.

Ciphertext letters: o y f j d n i s d r r t q w a i n r a c x
z m y n z b h h x

Total letters: Let's count the letters in the ciphertext.

First word: oyfjdnisdr (10 letters)

Second word: rtqwainr (8 letters)

Third word: acxz (4 letters)

Fourth word: mynzbhxx (8 letters)

Total letters in ciphertext: $10 + 8 + 4 + 8 = 30$ letters

Plaintext: Think step by step

First word: Think (5 letters)

Second word: step (4 letters)

Third word: by (2 letters)

Fourth word: step (4 letters)

Total letters in plaintext: $5 + 4 + 2 + 4 = 15$ letters

Hmm.

First ciphertext word: oyfjdnisdr (10 letters)

First plaintext word: Think (5 letters)

Hmm.

Second ciphertext word: rtqwainr (8 letters)

Second plaintext word: step (4 letters)

Third ciphertext word: acxz (4 letters)

Third plaintext word: by (2 letters)

Fourth ciphertext word: mynzbhxx (8 letters)

Fourth plaintext word: step (4 letters)

Interesting.

It seems that the ciphertext words are exactly twice as long as the plaintext words.

Idea: Maybe we need to take every other letter or rebuild the plaintext from the ciphertext accordingly.

Let's test this theory.

OpenAI o1-preview

oyfjdnisdr rtqwainr acxz mynzbhbx -> Think step by step

Use the example above to decode:

oyekaijzdf aaptcg suaokybhai ouow aqht
mynznvaatzacdfoulxxz

Idea: Maybe we need to take every other letter or rebuild the plaintext from the ciphertext accordingly.

Let's test this theory.

If we take every other letter from the ciphertext words, maybe we'll get the plaintext.

Let's try this for the first word:

Ciphertext word: o y f j d n i s d r (10 letters)

Alternate letters:

First letters (positions 1,3,5,7,9): o f d i d

Second letters (positions 2,4,6,8,10): y j n s r

Wait: Let's try taking every second letter or first letter.

Option 1: Take the odd positions:

Positions: 1 3 5 7 9

Letters: o f d i d

Plaintext word is "Think", letters are T h i n k

Hmm.

Perhaps we can map letters:

o f d i d --> Think

Not directly

Option 2: Try mapping as per an assigned code:
perhaps columns of letters?

Alternatively, perhaps the cipher is more complex.

What if each plaintext letter is being represented by two ciphertext letters?

Similarly for other words.

Suppose that each plaintext letter corresponds to pair of letters in the ciphertext.

Meaning that the first two letters of ciphertext map to the first plaintext letter, and so on.

Ciphertext Pairs -> Plaintext Letter

oy -> T

fj -> h

dn -> i

Now, what's the mapping from "oy" to 'T'?

Is there a way to map 'oy' to 'T'?

Similarly for others.

One possible way is that 'oy' represents 'T'.

But how?

Option 1: Sum of the letters.

Convert letters to numbers (A=1, B=2, ..., Z=26)

So let's try to get to 'T' from 'o' and 'y'

o = 15

y = 25

Sum: 15 + 25 = 40

But 'T' is 20.

Alternatively, perhaps subtract: 25 - 15 = 10.

No.

Alternatively, perhaps combine the numbers in some way.

Things to probe:

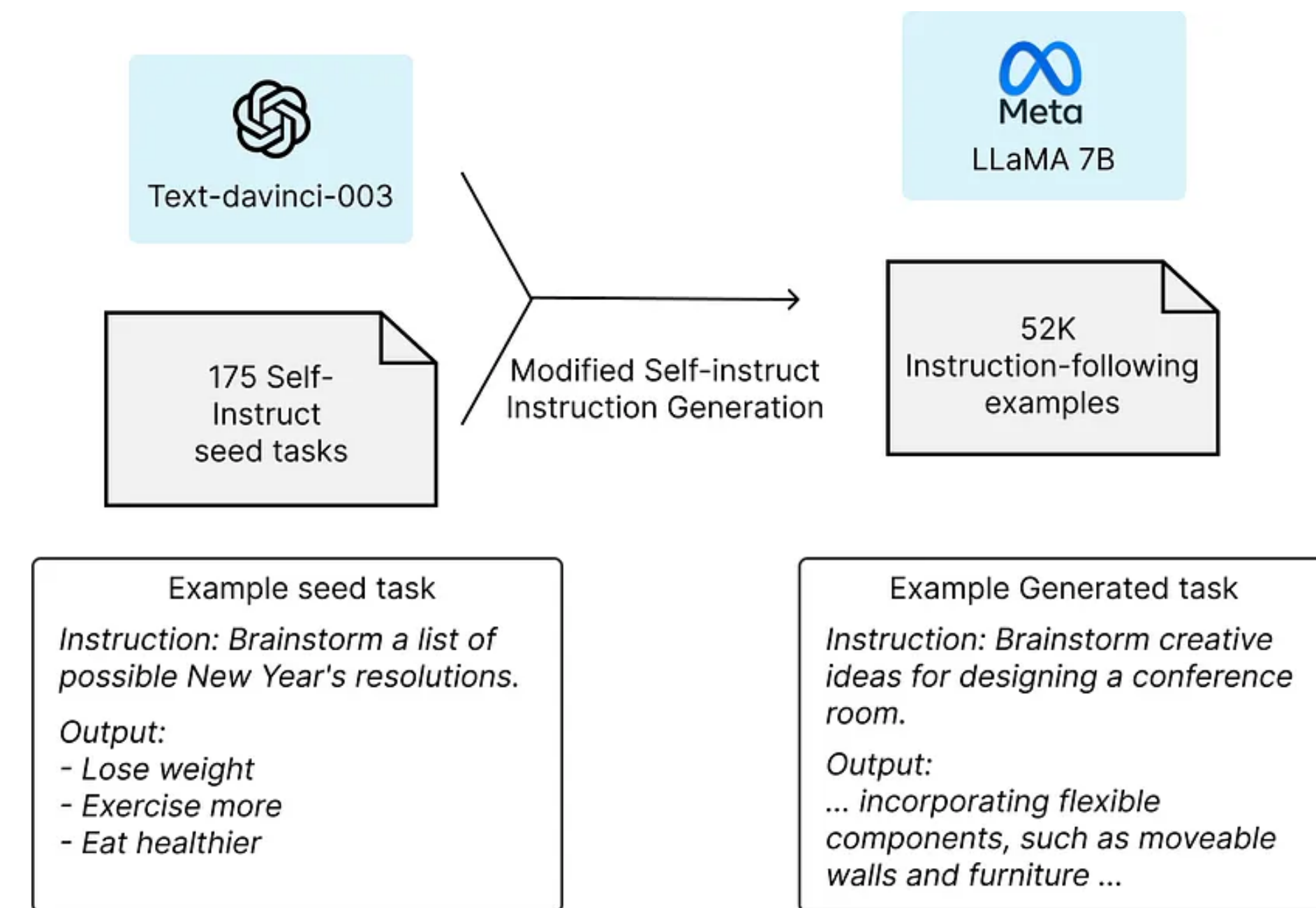
How is this model trained?

Let's examine some ideas in the literature/speculate.

- ✓ Where does the data come from?
 - e.g do humans write example questions/prompts? Models?
- Where is a reward function coming from?
(and how to use it?)
 - e.g. do humans write “reasoning traces”? Or model?
 - Or directly optimize with perplexity (PPL)?
- Is there an RL component?
 - It is being trained on its own generations. How?
 - RL/DP fundamentally involves backtracking/exploration.
(better viewed as training vs inference?)

Where do we get training “prompts”?

- Use human questions and/or LLM generated questions
 - e.g. the “Alpaca” instruction fine-tuning dataset uses 175 human-written instruction-output as examples to an LLM to generate more (52K) instruction examples <https://crfm.stanford.edu/2023/03/13/alpaca.html>
 - How much human labor?



Other ides:

- Extract questions from “real” text
 - e.g. textbooks
- Extract questions from math/science/github etc
 - e.g. extract lemmas or unit tests from papers/github

How is this model trained?

Let's examine some ideas in the literature/speculate.

- Where does the data come from?
 - e.g do humans write example questions/prompts? Models?
- ✓ • Where is a reward function coming from?
(and how to use it?)
 - e.g. do humans write “reasoning traces”? Or model?
 - Or directly optimize with perplexity (PPL)?
- Is there an RL component?
 - It is being trained on its own generations. How?
 - RL/DP fundamentally involves backtracking/exploration.
(better viewed as training vs inference?)

Let's Verify Step by Step

Hunter Lightman* Vineet Kosaraju* Yura Burda* Harri Edwards

Bowen Baker Teddy Lee Jan Leike John Schulman Ilya Sutskever

Karl Cobbe*

OpenAI

<https://arxiv.org/pdf/2305.20050>

Abstract

In recent years, large language models have greatly improved in their ability to perform complex multi-step reasoning. However, even state-of-the-art models still regularly produce logical mistakes. To train more reliable models, we can turn either to outcome supervision, which provides feedback for a final result, or process supervision, which provides feedback for each intermediate reasoning step. Given the importance of training reliable models, and given the high cost of human feedback, it is important to carefully compare the both methods. Recent work has already begun this comparison, but many questions still remain. We conduct our own investigation, finding that process supervision significantly outperforms outcome supervision for training models to solve problems from the challenging MATH dataset. Our process-supervised model solves 78% of problems from a representative subset of the MATH test set. Additionally, we show that active learning significantly improves the efficacy of process supervision. To support related research, we also release PRM800K, the complete dataset of 800,000 step-level human feedback labels used to train our best reward model.

Let

$$x^8 + 3x^4 - 4 = p_1(x)p_2(x) \cdots p_k(x),$$

where each non-constant polynomial $p_i(x)$ is monic with integer coefficients, and cannot be factored further over the integers. Compute $p_1(1) + p_2(1) + \cdots + p_k(1)$.

I notice that the given polynomial has even degree and only even powers of x , so I can try to make a substitution to simplify it.

Let $y = x^4$, then the polynomial becomes $y^2 + 3y - 4$, which is a quadratic equation.

I can factor this quadratic equation as $(y + 4)(y - 1)$, so the original polynomial is $(x^4 + 4)(x^4 - 1)$.

Now I need to factor each of these factors further, if possible, over the integers.

For the first factor, $x^4 + 4$, I recall the Sophie Germain identity, which says that $a^4 + 4b^4 = (a^2 + 2b^2 + 2ab)(a^2 + 2b^2 - 2ab)$.

Applying this identity with $a = x$ and $b = 1$, I get $x^4 + 4 = (x^2 + 2x + 2)(x^2 - 2x + 2)$.

For the second factor, $x^4 - 1$, I can use the difference of squares formula, which says that $a^2 - b^2 = (a + b)(a - b)$.

Applying this formula with $a = x^2$ and $b = 1$, I get $x^4 - 1 = (x^2 + 1)(x^2 - 1)$.

Then I can apply the difference of squares formula again with $a = x$ and $b = 1$, to get $x^2 - 1 = (x + 1)(x - 1)$.

So the original polynomial is $(x^4 + 4)(x^4 - 1) = (x^2 + 2x + 2)(x^2 - 2x + 2)(x^2 + 1)(x + 1)(x - 1)$.

These are all the irreducible factors over the integers, so $p_1(x) = x^2 + 2x + 2$, $p_2(x) = x^2 - 2x + 2$, $p_3(x) = x^2 + 1$, $p_4(x) = x + 1$, and $p_5(x) = x - 1$.

To find the sum of their values at $x = 1$, I just need to plug in $x = 1$ into each factor and add them up.

I get $p_1(1) = 1^2 + 2 \cdot 1 + 2 = 5$, $p_2(1) = 1^2 - 2 \cdot 1 + 2 = 1$, $p_3(1) = 1^2 + 1 = 2$, $p_4(1) = 1 + 1 = 2$, and $p_5(1) = 1 - 1 = 0$.

So the final answer is $5 + 1 + 2 + 2 + 0 = 10$.

Answer: 10

Let

$$x^8 + 3x^4 - 4 = p_1(x)p_2(x) \cdots p_k(x),$$

where each non-constant polynomial $p_i(x)$ is monic with integer coefficients, and cannot be factored further over the integers. Compute $p_1(1) + p_2(1) + \cdots + p_k(1)$.

This problem is about factoring a polynomial over the integers, which can be a tricky task.

One possible strategy is to look for rational roots of the polynomial, using the rational root theorem.

However, that would involve checking a lot of possible factors of -4 , and there is no guarantee that the polynomial has any rational roots at all.

Another possible strategy is to use some algebraic tricks to simplify the polynomial or reduce its degree.

I notice that the polynomial has only even powers of x , so I can try to substitute $y = x^2$ and see what happens.

Then the polynomial becomes $y^4 + 3y^2 - 4$, which is a quadratic equation in y^2 .

I can try to factor this quadratic equation by looking for two numbers that multiply to -4 and add to 3 .

Those numbers are 4 and -1 , so I can write $y^4 + 3y^2 - 4 = (y^2 + 4)(y^2 - 1)$.

Now I can substitute back $x^2 = y$ and get $x^8 + 3x^4 - 4 = (x^4 + 4)(x^4 - 1)$.

This looks promising, because I have reduced the degree of the factors from 8 to 4 .

I can try to factor each of these factors further, using the difference of squares formula.

I get $x^8 + 3x^4 - 4 = (x^2 + 2)(x^2 - 2)(x^2 + 1)(x^2 - 1)$.

I can apply the difference of squares formula again to the last factor and get $x^8 + 3x^4 - 4 = (x^2 + 2)(x^2 - 2)(x^2 + 1)(x + 1)(x - 1)$.

Now I have factored the polynomial completely into monic linear and quadratic factors with integer coefficients.

These are the $p_i(x)$'s that the problem is asking for.

To find the sum of their values at $x = 1$, I just need to plug in $x = 1$ into each factor and add them up.

I get $p_1(1) + p_2(1) + \cdots + p_k(1) = (1^2 + 2)(1^2 - 2)(1^2 + 1)(1 + 1)(1 - 1)$.

Simplifying, I get $p_1(1) + p_2(1) + \cdots + p_k(1) = (3)(-1)(2)(2)(0)$.

Multiplying, I get $p_1(1) + p_2(1) + \cdots + p_k(1) = 0$.

Answer: 0

Figure 2: Two solutions to the same problem, graded by the PRM. The solution on the left is correct while the solution on the right is incorrect. A green background indicates a high PRM score, and a red background indicates a low score. The PRM correctly identifies the mistake in the incorrect solution.

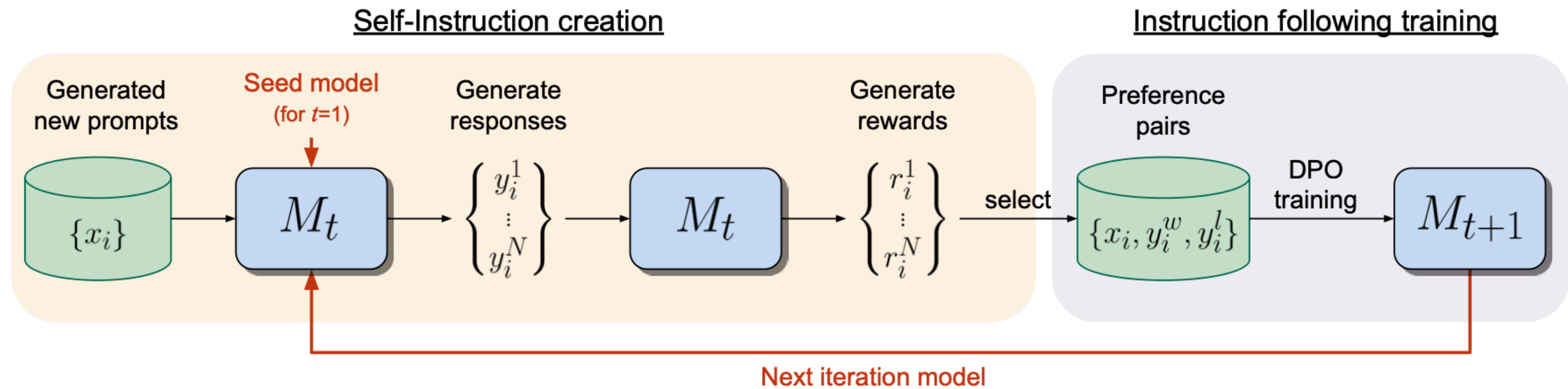
Self-Rewarding Language Models

Weizhe Yuan^{1,2} Richard Yuanzhe Pang^{1,2} Kyunghyun Cho²
Xian Li¹ Sainbayar Sukhbaatar¹ Jing Xu¹ Jason Weston^{1,2}

¹ Meta ² NYU

<https://arxiv.org/pdf/2401.10020>

human feedback in order to provide an adequate training signal. Current approaches commonly train reward models from human preferences, which may then be bottlenecked by human performance level, and secondly these separate frozen reward models cannot then learn to improve during LLM training. In this work, we study *Self-Rewarding Language Models*, where the language model itself is used via LLM-as-a-Judge prompting to provide its own rewards during training. We show that during Iterative DPO training that not only does instruction following ability improve, but also the ability to provide high-quality rewards to itself. Fine-tuning Llama 2 70B on three



Self-Rewarding Language Models

Weizhe Yuan^{1,2} Richard Yuanzhe Pang^{1,2} Kyunghyun Cho²
 Xian Li¹ Sainbayar Sukhbaatar¹ Jing Xu¹ Jason Weston^{1,2}

¹ Meta ² NYU

Figure 1: **Self-Rewarding Language Models.** Our self-alignment method consists of two steps: (i) *Self-Instruction creation*: newly created prompts are used to generate candidate responses from model M_t , which also predicts its own rewards via LLM-as-a-Judge prompting. (ii) *Instruction following training*: preference pairs are selected from the generated data, which are used for training via DPO, resulting in model M_{t+1} . This whole procedure can then be iterated resulting in both improved instruction following and reward modeling ability.

Table 1: **AlpacaEval 2.0 results** (win rate over GPT-4 Turbo evaluated by GPT-4). Self-Rewarding iterations yield improving win rates. Iteration 3 (M_3) outperforms many existing models that use proprietary training data or targets distilled from stronger models.

Model	Win Rate	Alignment Targets	
		Distilled	Proprietary
Self-Rewarding 70B			
Iteration 1 (M_1)	9.94%		
Iteration 2 (M_2)	15.38%		
Iteration 3 (M_3)	20.44%		
<i>Selected models from the leaderboard</i>			
GPT-4 0314	22.07%		✓
Mistral Medium	21.86%		✓
Claude 2	17.19%		✓
Gemini Pro	16.85%		✓
GPT-4 0613	15.76%		✓
GPT 3.5 Turbo 0613	14.13%		✓
LLaMA2 Chat 70B	13.87%		✓
Vicuna 33B v1.3	12.71%	✓	
Humpback LLaMa2 70B	10.12%		
Guanaco 65B	6.86%		
Davinci001	2.76%		✓
Alpaca 7B	2.59%	✓	

2.2 Self-Instruction Creation

Using the model we have trained, we can make it self-modify its own training set. Specifically, we generate additional training data for the next iteration of training.

This consists of the following steps:

1. **Generate a new prompt:** We generate a new prompt x_i using few-shot prompting, sampling prompts from the original seed IFT data, following the approach of Wang et al. [2023] and Honovich et al. [2023].¹
2. **Generate candidate responses:** We then generate N diverse candidate responses $\{y_i^1, \dots, y_i^N\}$ for the given prompt x_i from our model using sampling.
3. **Evaluate candidate responses:** Finally, we use the LLM-as-a-Judge ability of our same model to evaluate its own candidate responses with scores $r_i^n \in [0, 5]$ (exact prompt given in Figure 2).

How is this model trained?

Let's examine some ideas in the literature/speculate.

- Where does the data come from?
 - e.g do humans write example questions/prompts? Models?
- Where is a reward function coming from?
(and how to use it?)
 - e.g. do humans write “reasoning traces”? Or model?
 - Or directly optimize with perplexity (PPL)?
- ✓ • Is there an RL component?
 - It is being trained on its own generations. How?
 - RL/DP fundamentally involves backtracking/exploration.
(better viewed as training vs inference?)

Quiet-STaR: Language Models Can Teach Themselves to Think Before Speaking

Eric Zelikman
Stanford University

Georges Harik
Notbad AI Inc

Yijia Shao
Stanford University

Varuna Jayasiri
Notbad AI Inc

Nick Haber
Stanford University

Noah D. Goodman
Stanford University

Abstract

When writing and talking, people sometimes pause to think. Although reasoning-focused works have often framed reasoning as a method of answering questions or completing agentic tasks, reasoning is implicit in almost all written text. For example, this applies to the steps not stated between the lines of a proof or to the theory of mind underlying a conversation. In the Self-Taught Reasoner (STaR, Zelikman et al. 2022), useful thinking is learned by inferring rationales from few-shot examples in question-answering and learning from those that lead to a correct answer. This is a highly constrained setting – ideally, a language model could instead learn to infer unstated rationales in arbitrary text. We present **Quiet-STaR**, a generalization of STaR in which LMs learn to generate rationales at each token to explain future text, improving their predictions. We address key challenges, including 1) the computational cost of generating continuations, 2) the fact that the LM does not initially know

<https://arxiv.org/pdf/2403.09629>

THINK BEFORE YOU SPEAK: TRAINING LANGUAGE MODELS WITH PAUSE TOKENS

Sachin Goyal*
Machine Learning Department
Carnegie Mellon University
sachingo@andrew.cmu.edu

Ziwei Ji
Google Research, NY
ziwei ji@google.com

Ankit Singh Rawat
Google Research, NY
ankitsrawat@google.com

Aditya Krishna Menon
Google Research, NY
adityakmenon@google.com

Sanjiv Kumar
Google Research, NY
sanjivk@google.com

Vaishnavh Nagarajan†
Google Research, NY
vaishnavh@google.com

<https://arxiv.org/pdf/2310.02226>

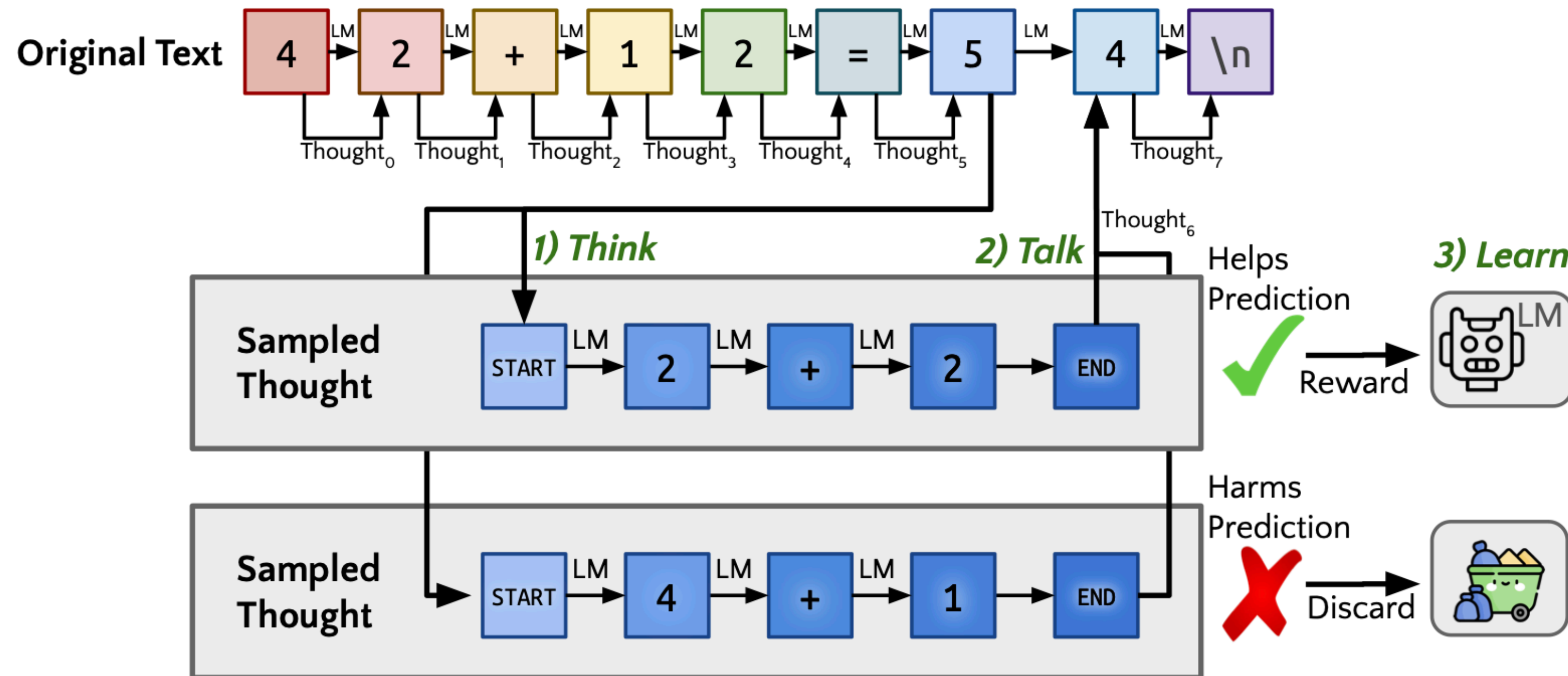


Figure 1: **Quiet-STaR**. We visualize the algorithm as applied during training to a single thought. We generate thoughts, in parallel, following all tokens in the text (**think**). The model produces a mixture of its next-token predictions with and without a thought (**talk**). We apply REINFORCE, as in STaR, to increase the likelihood of thoughts that help the model predict future text while discarding thoughts that make the future text less likely (**learn**).

In these settings, we can train on the usual cross entropy loss.

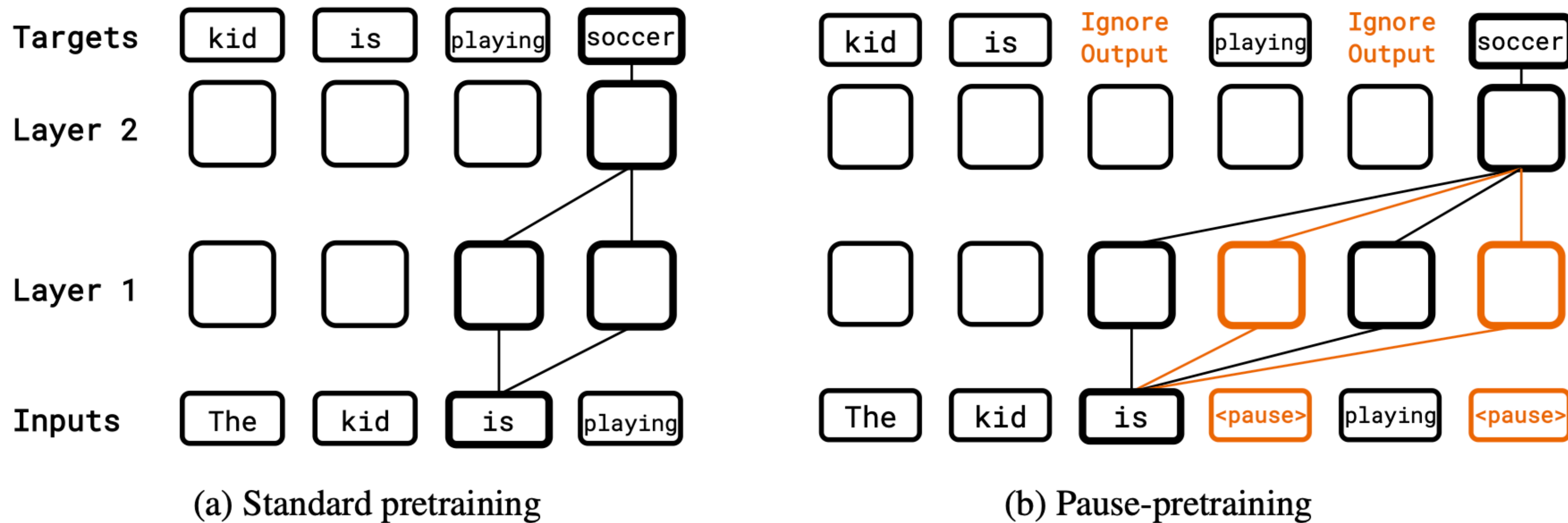


Figure 2: **Standard vs. pause-pretraining.** We consider pretraining based on causal language modeling, where each token is predicted given all preceding tokens in the sequence, using unidirectional self-attention. Here, we visualize the computational pathways beginning from the token “is” on the input side of the decoder-only model, to a subsequent token “soccer” on the output side. Please see Figure 1 for a guide on how to follow this visualization. (a) In standard pretraining, we compute the model’s loss at each output token, and backpropagate through it. (b) In pause-pretraining, we insert multiple copies of <pause> tokens at uniformly random locations in the input. However, we do not apply a loss on the model to predict these tokens, as indicated by each corresponding **Ignore Output** flags. This introduces new computational pathways connecting the input token and the output token of interest.

In these settings, we can train on the usual cross entropy loss.

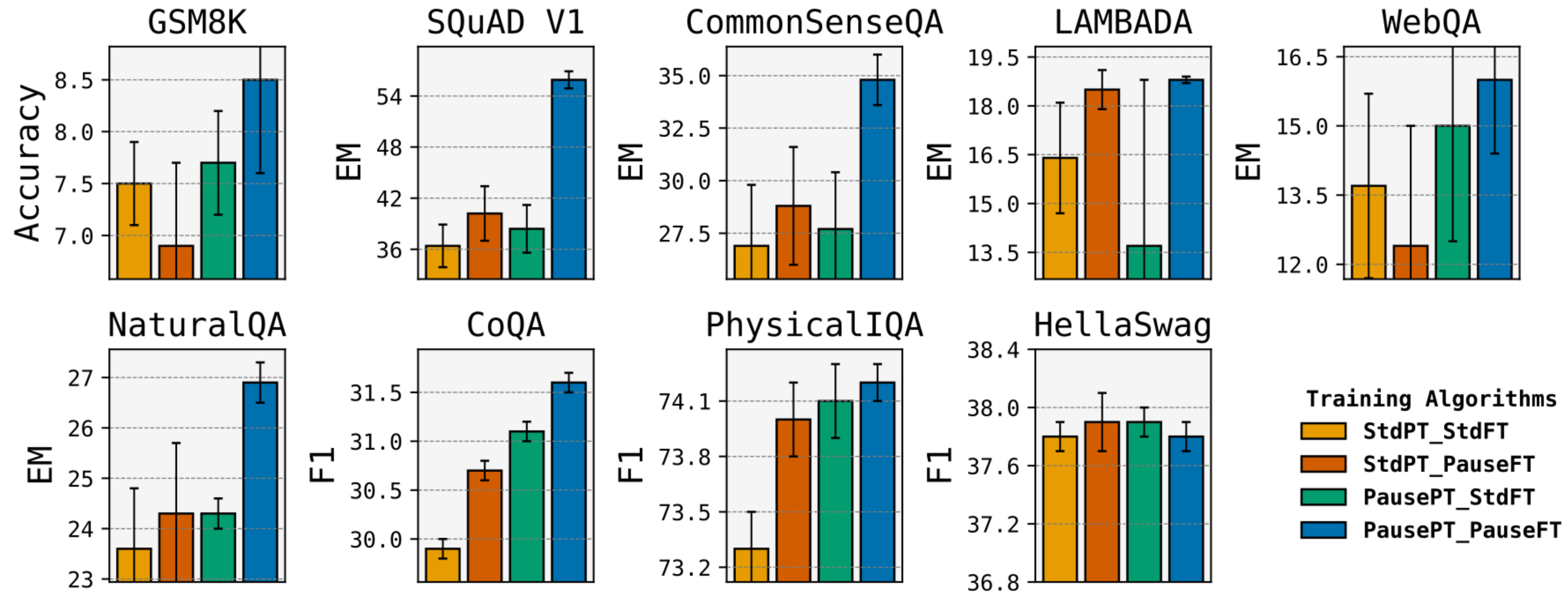


Figure 3: **Downstream performance for a 1B model.** Injecting delays in both stages of training (PausePT_PauseFT) outperforms the standard end-end training StdPT_StdFT on our wide variety of tasks (except HellaSwag). In contrast, introducing delays only in the finetuning stage provides only lukewarm gains, and even hurts in GSM8k.

Summary:

- Pre-training: train to minimize ppl
- Mid: “bootstrap” with the base model (e.g. tool-former)
- Post: make the base model helpful

- Reasoning:
 - Is this a new paradigm where training on generations/RL is helpful?
 - How important will more inference time flops be?

Questions?