Test-time scaling (cont'd) + Compound AI systems

CS 6804: Frontier AI Systems

Spring 2026

https://tuvllms.github.io/ai-seminar-spring-2026/

Tu Vu

VIRGINIA TECH

Logistics

- Homework assignments
  - **Quiz 1** (later this week)
  - **Homework 2** (later this week)
    - 5% extra credits
  - **Main homework** on reinforcement learning
    - late March / early April
  - **In-class participation** (starting next week)

Student presentations

https://piazza.com/class/mko85xtkj4k3gn/post/20

Interested topics for presentation *

- [ ] LLM-as-a-Judge & LLM Evaluation
- [ ] LLM Agents
- [ ] Self-evolving agents
- [ ] Context Engineering & Agentic Memory
- [ ] On-policy distillation
- [ ] AI for scientific research
- [ ] LLM security & privacy
- [ ] AI content detection
- [ ] Parameter-efficient training & adaptation
- [ ] Position encoding & efficient attention
- [ ] Reinforcement learning
- [ ] Test-time scaling
- [ ] Long-context LLMs
- [ ] Diffusion models & Diffusion LLMs
- [ ] Retrieval-augmented generation & Tool-use models
- [ ] Multilingual models
- [ ] Multimodal models
- [ ] Novel architectures (e.g., Mamba)

Student presentations (cont'd)

Role-Playing Paper-Reading Seminars

Alec Jacobson and Colin Raffel
March 17th, 2021
colinraffel.com/blog

The paper-reading seminar is a common format for advanced topics courses in Computer Science graduate programs. Over the span of the course, students read a series of academic papers published on a particular topic (for example, Deep Learning with Limited Labeled Data). For cutting-edge topics close to the frontier of research, standard texts may not exist and academic papers are the primary available references. Some niche areas may not be mature enough to develop a lecture-based course curriculum around. Complementary to course topic-specific learning objectives, paper-reading seminars aim to achieve learning outcomes related to the soft skills of technical reading, public presentation, and critical discussion.

https://colinraffel.com/blog/role-playing-seminar.html


Paper #1, Role #3


Paper #1, Role #3

Scientific Peer Reviewer. The paper has not been published yet and is currently submitted to a top conference where you've been assigned as a peer reviewer. Complete a full review of the paper answering all prompts of the official review form of the top venue in this research area (e.g., *NeurIPS* for Deep Learning and *ACM SIGGRAPH* for Geometry & Animation). This includes recommending whether to accept or reject the paper.

Archaeologist. This paper was found buried under ground in the desert. You're an archeologist who must determine where this paper sits in the context of previous and subsequent work. Find and report on one *older* paper cited within the current paper that substantially influenced the current paper and one *newer* paper that cites this current paper.

Academic Researcher. You're a researcher who is working on a new project in this area. Propose an imaginary follow-up project *not just* based on the current but only possible due to the existence and success of the current paper.

Industry Practitioner. You work at a company or organization developing an application or product of your choice (that has not already been suggested in a prior session). Bring a convincing pitch for why you should be paid to implement the method in the paper, and discuss at least one positive and negative impact of this application.

Hacker. You're a hacker who needs a demo of this paper ASAP. Implement a small part or simplified version of the paper on a small dataset or toy problem. Prepare to share the core code of the algorithm to the class and demo your implementation. Do not simply download and run an existing implementation – though you are welcome to use (and give credit to) an existing implementation for "backbone" code.

Private Investigator. You are a detective who needs to run a background check on one of the paper's authors. Where have they worked? What did they study? What previous projects might have led to working on this one? What motivated them to work on this project? Feel free to contact the authors, but remember to be courteous, polite, and on-topic.

Social Impact Assessor. Identify how this paper self-assesses its (likely positive) impact on the world. Have any additional positive social impacts left out? What are possible negative social impacts that were overlooked or omitted?

AI news

Andrej Karpathy
@karpathy

I packaged up the "autoresearch" project into a new self-contained minimal repo if people would like to play over the weekend. It's basically nanochat LLM training core stripped down to a single-GPU, one file version of ~630 lines of code, then:

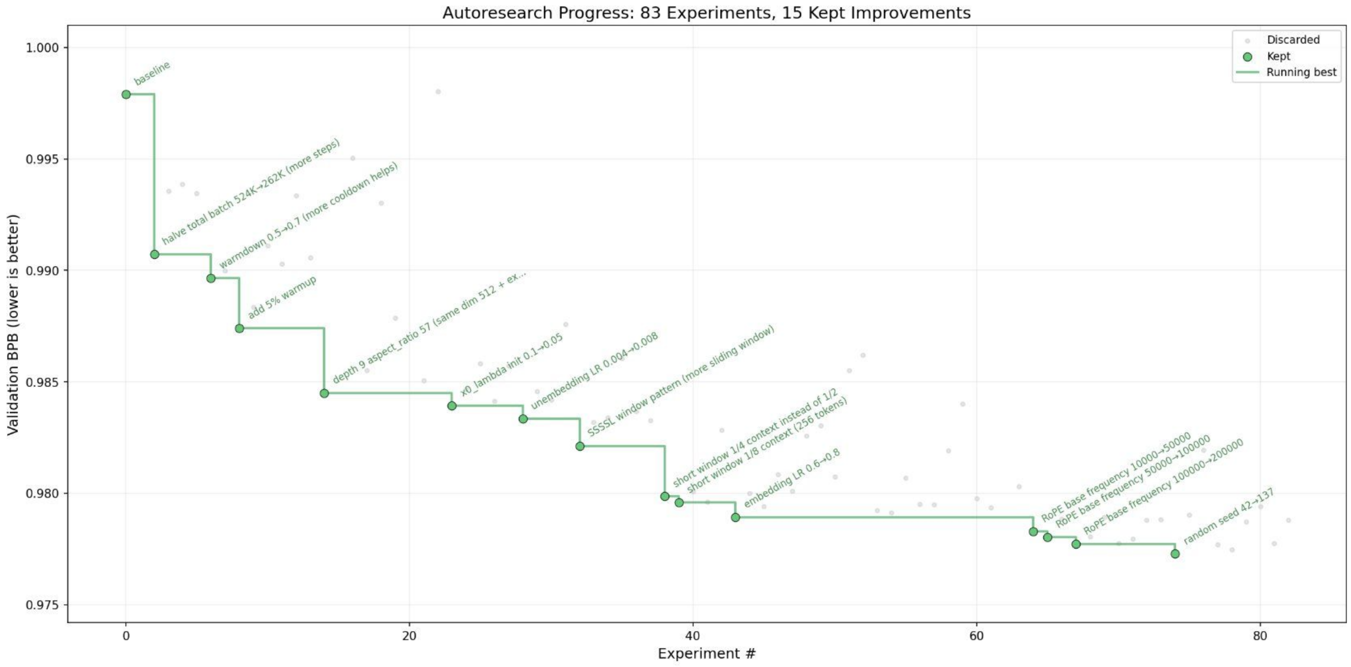
- the human iterates on the prompt (.md)
- the AI agent iterates on the training code (.py)

The goal is to engineer your agents to make the fastest research progress indefinitely and without any of your own involvement. In the image, every dot is a complete LLM training run that lasts exactly 5 minutes. The agent works in an autonomous loop on a git feature branch and accumulates git commits to the training script as it finds better settings (of lower validation loss by the end) of the neural network architecture, the optimizer, all the hyperparameters, etc. You can imagine comparing the research progress of different prompts, different agents, etc.

github.com/karpathy/autor...
Part code, part sci-fi, and a pinch of psychosis :)

AI news (cont'd)

autoresearch



One day, frontier AI research used to be done by meat computers in between eating, sleeping, having other fun, and synchronizing once in a while using sound wave interconnect in the ritual of "group meeting". That era is long gone. Research is now entirely the domain of autonomous swarms of AI agents running across compute cluster megastructures in the skies. The agents claim that we are now in the 10,205th generation of the code base, in any case no one could tell if that's right or wrong as the "code" is now a self-modifying binary that has grown beyond human comprehension. This repo is the story of how it all began. -@karpathy, March 2026.

RL*F

Constitutional AI: Harmlessness from AI Feedback

Yuntao Bai,[*] Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion,

Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon,
Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain,
Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller,
Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt,
Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma,
Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec,
Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly,
Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann,
Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, Jared Kaplan[*]

Anthropic

RLAIF

Constitutional AI: Harmlessness from AI Feedback

Yuntao Bai*, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion,

Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon,
Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain,
Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller,
Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt,
Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma,
Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec,
Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly,
Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann,
Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, Jared Kaplan*

Anthropic

RLCF

Checklists Are Better Than Reward Models For Aligning Language Models

Vijay Viswanathan ♡ Yanchao Sun♣ Shuang Ma♣* Xiang Kong♣
Meng Cao♣ Graham Neubig♡ Tongshuang Wu♡
♡ Carnegie Mellon University ♣Apple

Abstract

Language models must be adapted to understand and follow user instructions. Reinforcement learning is widely used to facilitate this – typically using fixed criteria such as "helpfulness" and "harmfulness". In our work, we instead propose using flexible, instruction-specific criteria as a means of broadening the impact that reinforcement learning can have in eliciting instruction following. We propose "**Reinforcement Learning from Checklist Feedback**" (**RLCF**). From instructions, we extract checklists and evaluate how well responses satisfy each item—using both AI judges and specialized verifier programs—then combine these scores to compute rewards for RL. We compare RLCF with other alignment methods on top of a strong instruction following model (`Qwen2.5-7B-Instruct`) on five widely-studied benchmarks – **RLCF is the only method to help on every benchmark**, including a 4-point boost in hard satisfaction rate on FollowBench, a 6-point increase on InFoBench, and a 3-point rise in win rate on Arena-Hard. We show that RLCF can also be used off-policy to improve `Llama 3.1 8B Instruct` and `OLMo 2 7B Instruct`. These results establish rubrics as a key tool for improving language models' support of queries that express a multitude of needs. We release our our dataset of rubrics (*WildChecklists*), models, and code to the public.[1]

Rubrics-based

scale

Rubrics as Rewards: Reinforcement Learning Beyond Verifiable Domains

Anisha Gunjal, Anthony Wang*, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx

Scale AI

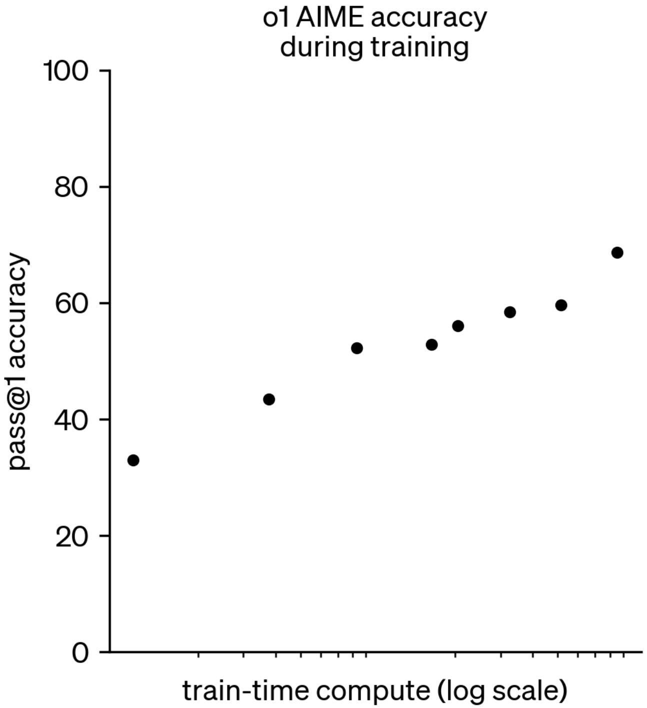
**Work done during internship at Scale AI*

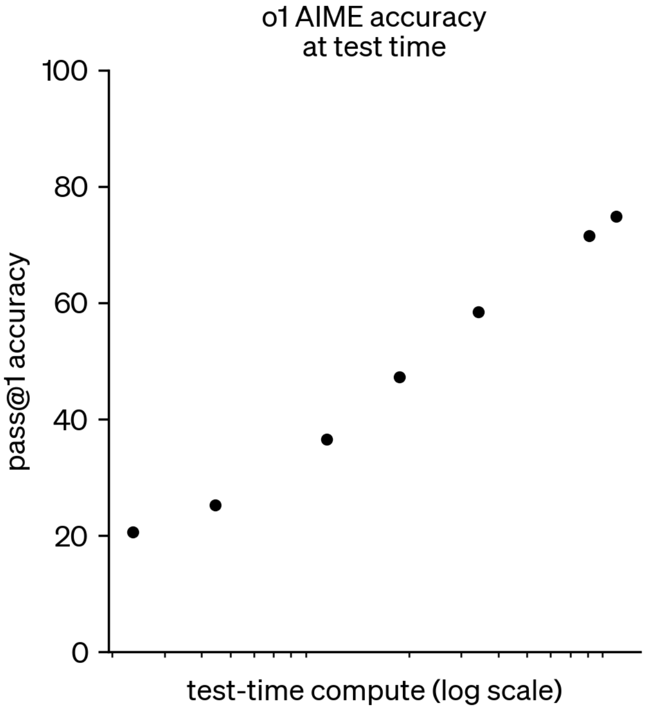
anisha.gunjal@scale.com

Abstract

Reinforcement Learning with Verifiable Rewards (RLVR) has proven effective for complex reasoning tasks with clear correctness signals such as math and coding. However, extending it to real-world reasoning tasks is challenging, as evaluation depends on nuanced, multi-criteria judgments rather than binary correctness. Instance-specific rubrics have recently been used in evaluation benchmarks to capture such judgments, but their potential as reward signals for on-policy post-training remains underexplored. We introduce **Rubrics as Rewards (*RaR*)**, an on-policy reinforcement learning method that extends RLVR beyond verifiable domains by using rubric-based feedback. Across both medical and science domains, we evaluate multiple strategies for aggregating rubric feedback into rewards. The best RaR variant achieves relative improvements of up to 31% on HealthBench and 7% on GPQA-Diamond over popular LLM-as-judge baselines that rely on direct Likert-based rewards. These results demonstrate that RaR-trained policies adapt well to diverse evaluation formats, performing strongly on both rubric-based and multiple-choice tasks. Moreover, we find that using rubrics as structured reward signals yields better alignment for smaller judges and reduces performance variance across judge scales.

Test-time scaling (cont'd)





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

Test-time scaling methods

- **Parallel (repeated sampling)**
  - multiple solution attempts (run independently)
  - chooses the most frequent or the best response
  - Brown et al. (2024); Irvine et al. (2023); Levi (2024)
- **Sequential**
  - later computations depend on earlier ones (e.g., a long reasoning trace)
  - allows it to refine each attempt based on previous outcomes
  - Muennighoff et al.(2025); Snell et al. (2024); Hou et al. (2025); Lee et al. (2025)

Large Language Monkeys: Scaling Inference Compute with Repeated Sampling

Bradley Brown[*†‡], Jordan Juravsky[*†], Ryan Ehrlich[*†], Ronald Clark[‡], Quoc V. Le[§], Christopher Ré[†], and Azalia Mirhoseini[†§]

[†]Department of Computer Science, Stanford University
[‡]University of Oxford
[§]Google DeepMind

bradley.brown@cs.ox.ac.uk, jbj@stanford.edu, ryanehrlich@cs.stanford.edu, ronald.clark@cs.ox.ac.uk, qvl@google.com, chrismre@stanford.edu, azalia@stanford.edu

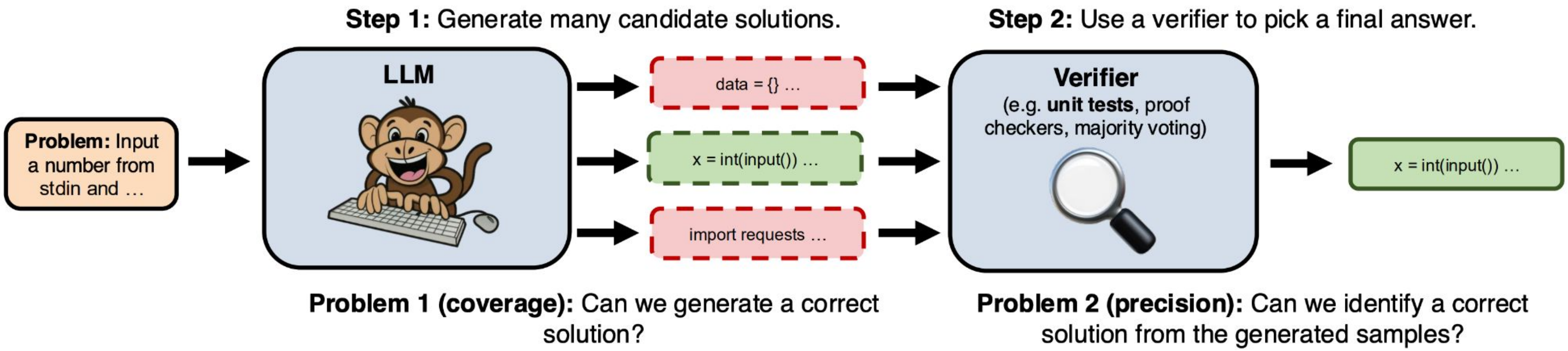


Figure 1: The repeated sampling procedure that we follow in this paper. 1) We generate many independent candidate solutions for a given problem by sampling from an LLM with a positive temperature. 2) We use a domain-specific verifier (ex. unit tests for code) to select a final answer from the generated samples.

Measuring coverage

$$\text{pass@}k = \frac{1}{\#\text{ of problems}} \sum_{i=1}^{\#\text{ of problems}} \left[1 - \frac{\binom{N - C_i}{k}}{\binom{N}{k}} \right]$$

$$\text{pass@}k = \frac{1}{\#\text{ of problems}} \sum_{i=1}^{\#\text{ of problems}} \left[1 - \frac{\binom{N-C_i}{k}}{\binom{N}{k}}\right]$$

- N: Total number of completions (outputs) generated per problem.
- C_i: Number of **correct** completions among the N for problem i.
- $\binom{N}{k}$: Number of ways to choose k outputs from N.
- $\binom{N-C_i}{k}$: Number of ways to choose k outputs *all of which are incorrect* (i.e., chosen only from the $N - C_i$ incorrect ones).
- So, $\frac{\binom{N-C_i}{k}}{\binom{N}{k}}$ is the **probability** that you choose **only incorrect outputs** when sampling k outputs from the N total.
- Subtracting this from 1 gives the probability that **at least one** correct solution is in the top-k.

Coverage increases as we scale the number of samples

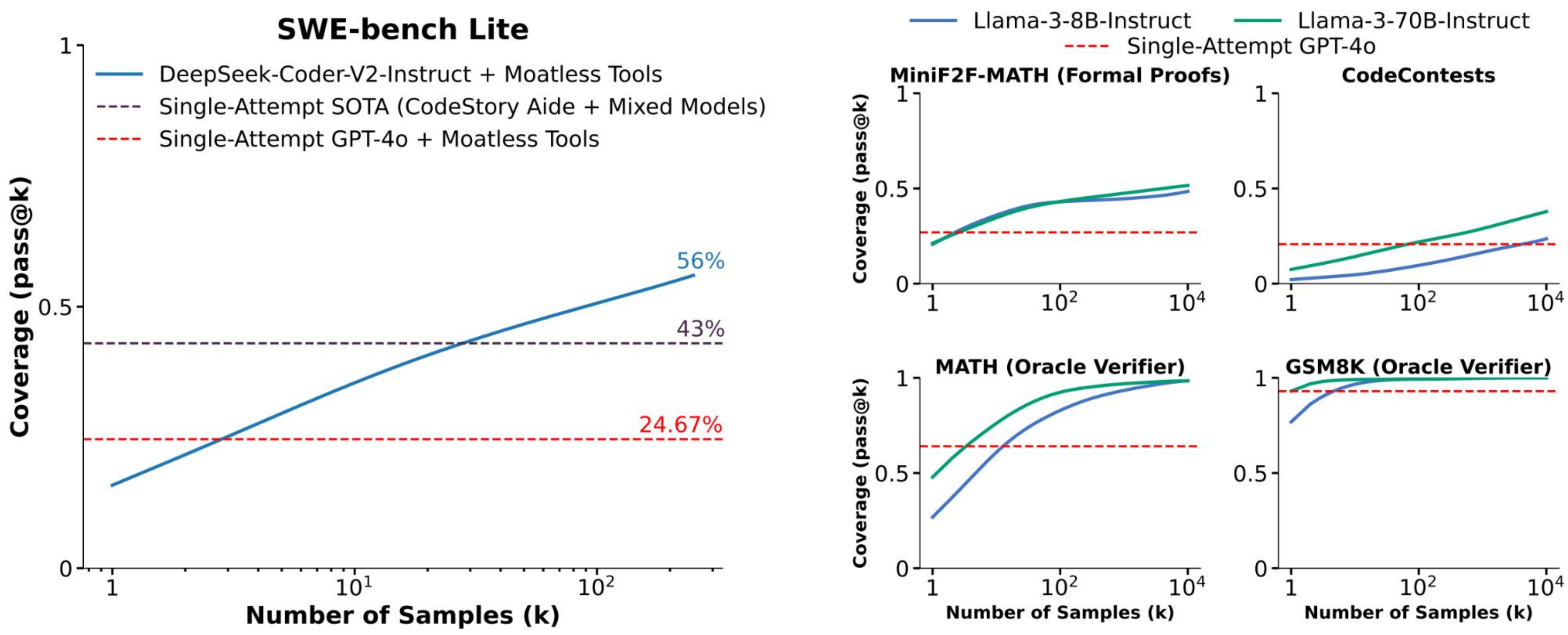


Figure 2: Across five tasks, we find that coverage (the fraction of problems solved by at least one generated sample) increases as we scale the number of samples. Notably, using repeated sampling, we are able to increase the solve rate of an open-source method from 15.9% to 56% on SWE-bench Lite.

Scaling inference time compute via repeated sampling leads to consistent coverage gains

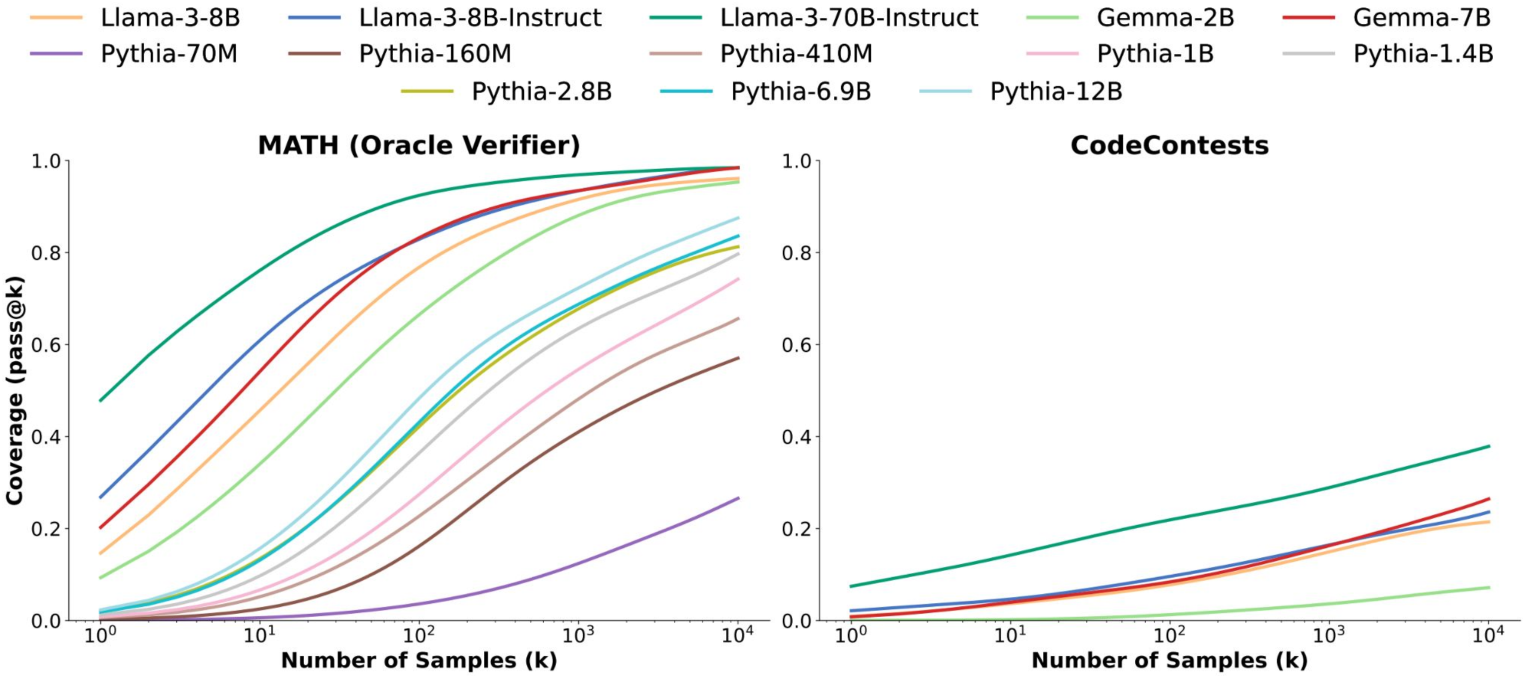


Figure 3: Scaling inference time compute via repeated sampling leads to consistent coverage gains across a variety of model sizes (70M-70B), families (Llama, Gemma and Pythia) and levels of post-training (Base and Instruct models).

Inference FLOPs

$$\text{FLOPsPerToken}(\text{ContextLen}) \approx 2 * (\text{NumParameters} + 2 * \text{NumLayers} * \text{TokenDim} * \text{ContextLen})$$

$$\text{TotalInferenceFLOPs} \approx \left(\sum_{t=1}^{\text{NumPromptTokens}} \text{FLOPsPerToken}(t)\right) + \left(\sum_{t=1}^{\text{NumDecodeTokens}} \text{FLOPsPerToken}(t + \text{NumPromptTokens}) * \text{NumCompletions}\right)$$

Ideal model size depends on the task, compute budget, and coverage requirements

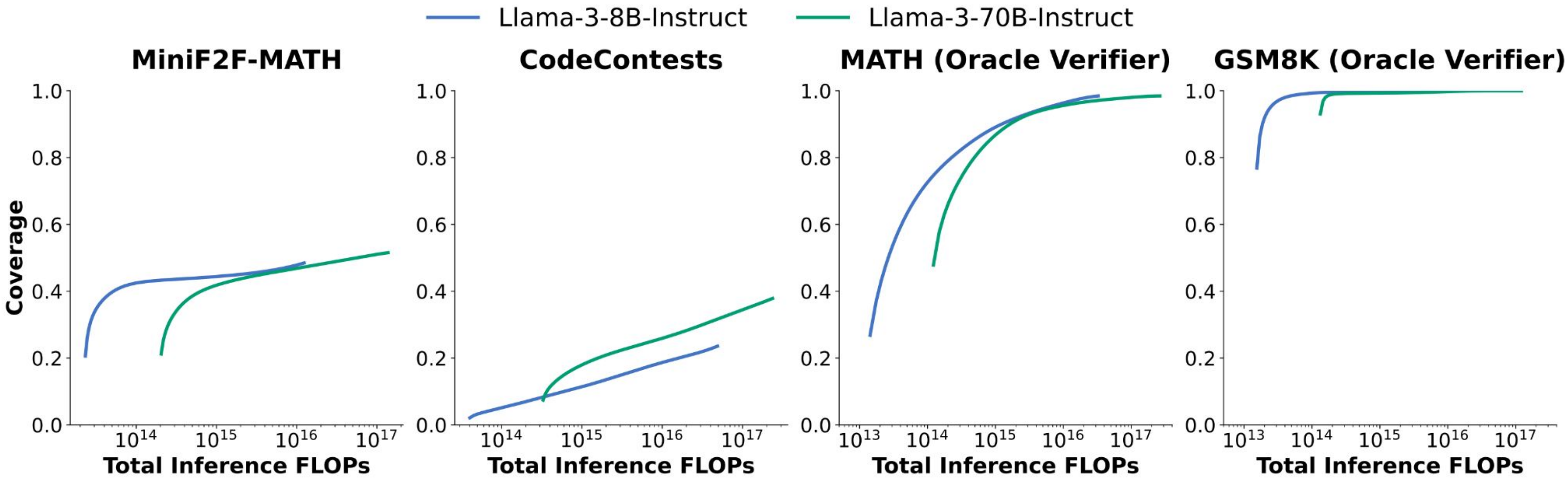


Figure 4: Comparing cost, measured in number of inference FLOPs, and coverage for Llama-3-8B-Instruct and Llama-3-70B-Instruct. We see that the ideal model size depends on the task, compute budget, and coverage requirements. Note that Llama-3-70B-Instruct does not achieve 100% coverage on GSM8K due to an incorrectly labelled ground truth answer: see Appendix E.

API cost

Model	Cost per attempt (USD)	Number of attempts	Issues solved (%)	Total cost (USD)	Relative total cost
DeepSeek-Coder-V2-Instruct	0.0072	5	29.62	10.8	1x
GPT-4o	0.13	1	24.00	39	3.6x
Claude 3.5 Sonnet	0.17	1	26.70	51	4.7x

Table 1: Comparing API cost (in US dollars) and performance for various models on the SWE-bench Lite dataset using the Moatless Tools agent framework. When sampled more, the open-source DeepSeek-Coder-V2-Instruct model can achieve the same issue solve-rate as closed-source frontier models for under a third of the price.

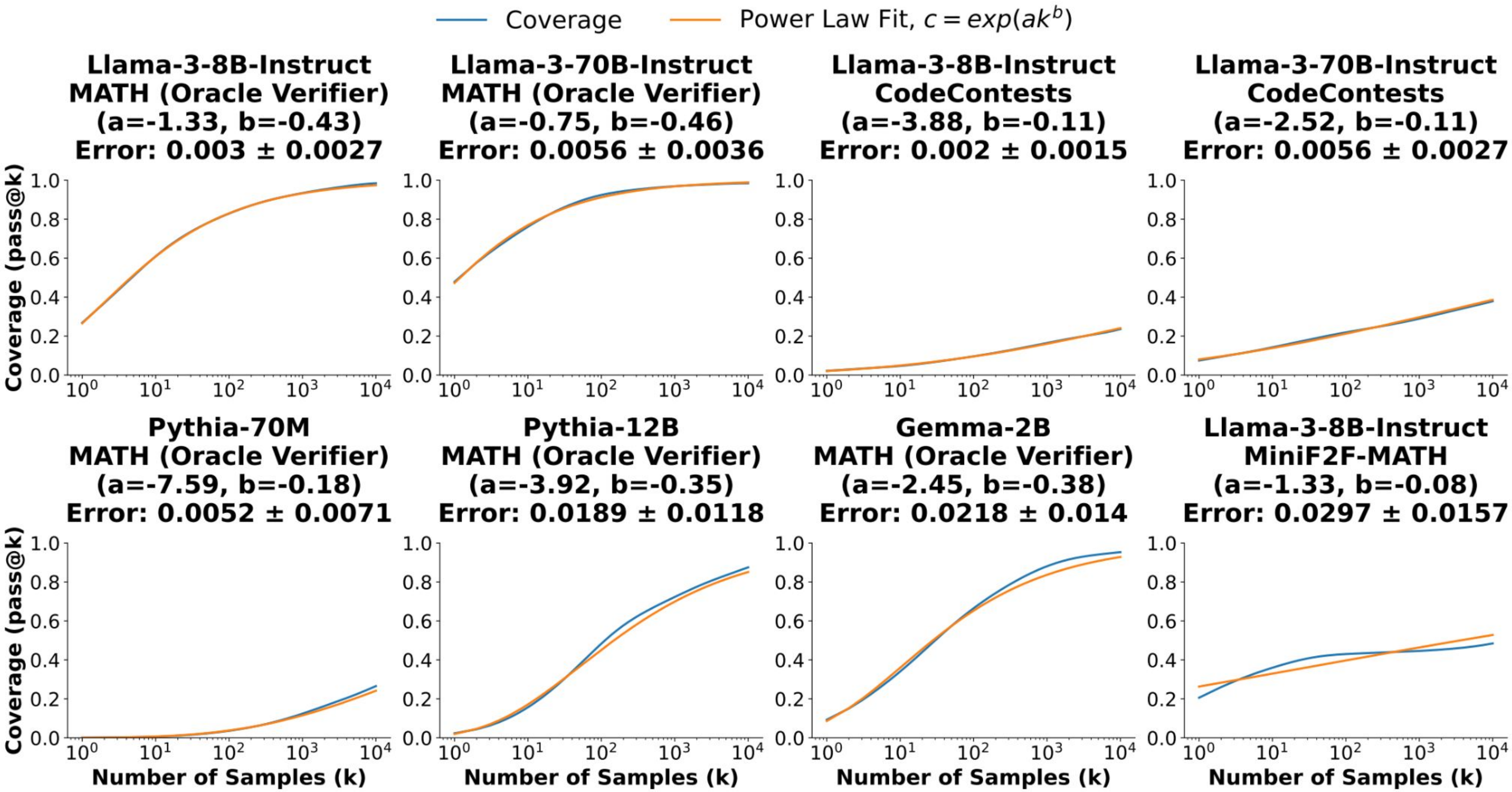


Figure 5: The relationship between coverage and the number of samples can be modelled with an exponentiated power law for most tasks and models. We highlight that some curves, such as Llama-3-8B-Instruct on MiniF2F-MATH, do not follow this trend closely. We show the mean and standard deviation of the error between the coverage curve and the power law fit across 100 evenly sampled points on the log scale.

We model the logarithm of coverage c as a function of the number of samples k:

$$\log(c) \approx ak^b$$

where $a, b \in \mathbb{R}$ are parameters to be fitted.

To directly predict coverage c, we exponentiate both sides:

$$c \approx \exp(ak^b)$$

The relationship between coverage and the number of samples modelled with an exponentiated power law

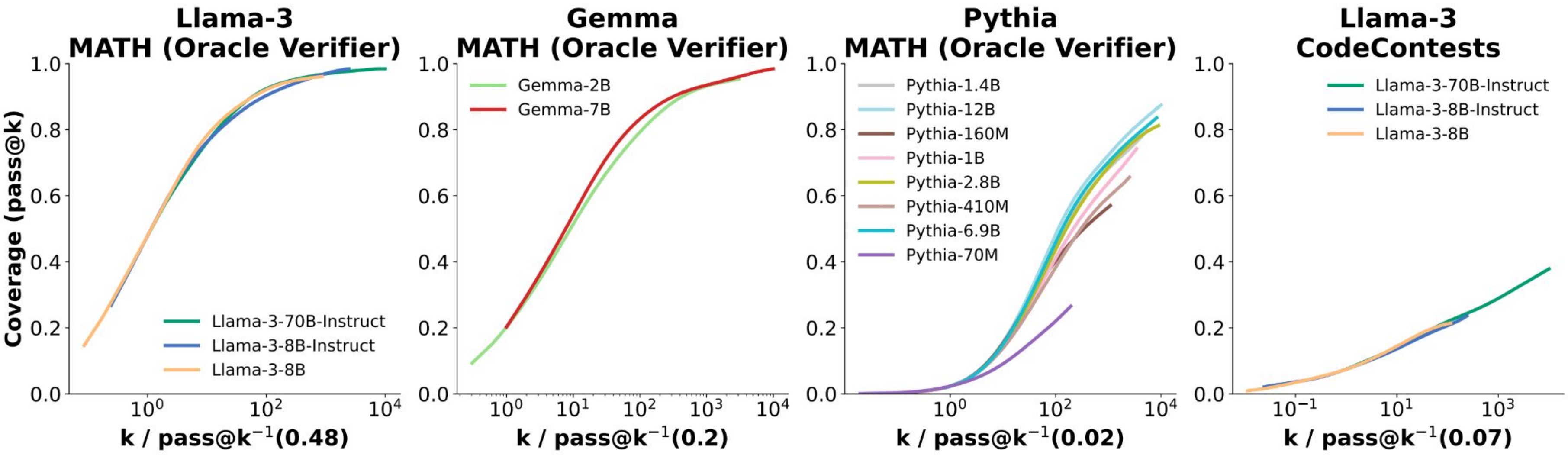


Figure 6: Overlaying the coverage curves from different models belonging to the same family. We perform this overlay by horizontally shifting every curve (with a logarithmic x-axis) so that all curves pass through the point $(1, c)$. We pick c to be the maximum pass@1 score over all models in the plot. We note that the similarity of the curves post-shifting shows that, within a model family, sampling scaling curves follow a similar shape.

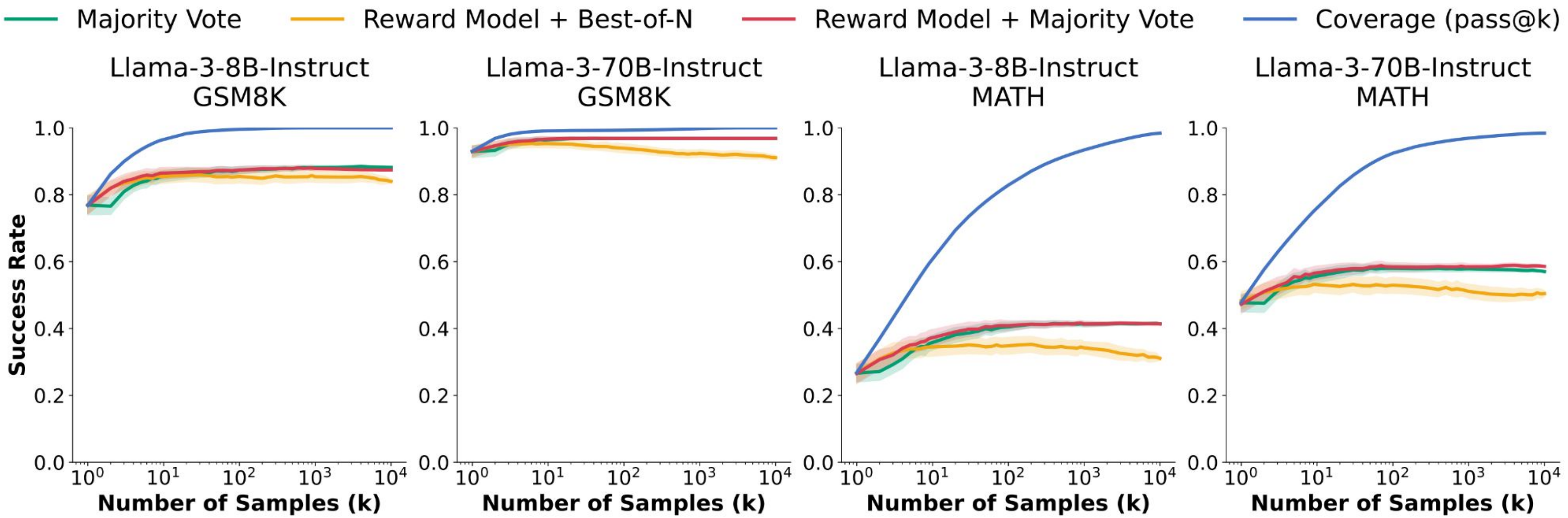


Figure 7: Comparing coverage (performance with an oracle verifier) to mainstream methods available for picking the correct answer (majority voting, reward model selection and reward model majority voting) as we increase the number of samples. Although near-perfect coverage is achieved, all sample selection methods fail to reach the coverage upper bound and saturate before reaching 100 samples. For every k value, we calculate the metric on 100 subsets of size k then plot the mean and one standard deviation across subsets.

Pass@1	# Problems	# CoT Graded	Correct CoT	Incorrect CoT	Incorrect Ground Truth
0-10%	5	15	11	1	1 problem, 3 CoTs
10-25%	10	30	27	3	0 problems
25-75%	29	30	28	2	0 problems
75-100%	84	30	30	0	0 problems

Table 2: Human evaluation of the validity of the Chain-of-Thought reasoning in Llama-3-8B-Instruct answers to GSM8K problems. 3 chains of thought were graded per problem. Even for difficult questions, where the model only gets $\leq 10\%$ of samples correct, the CoTs almost always follow valid logical steps. For the model generations and human labels, see here.

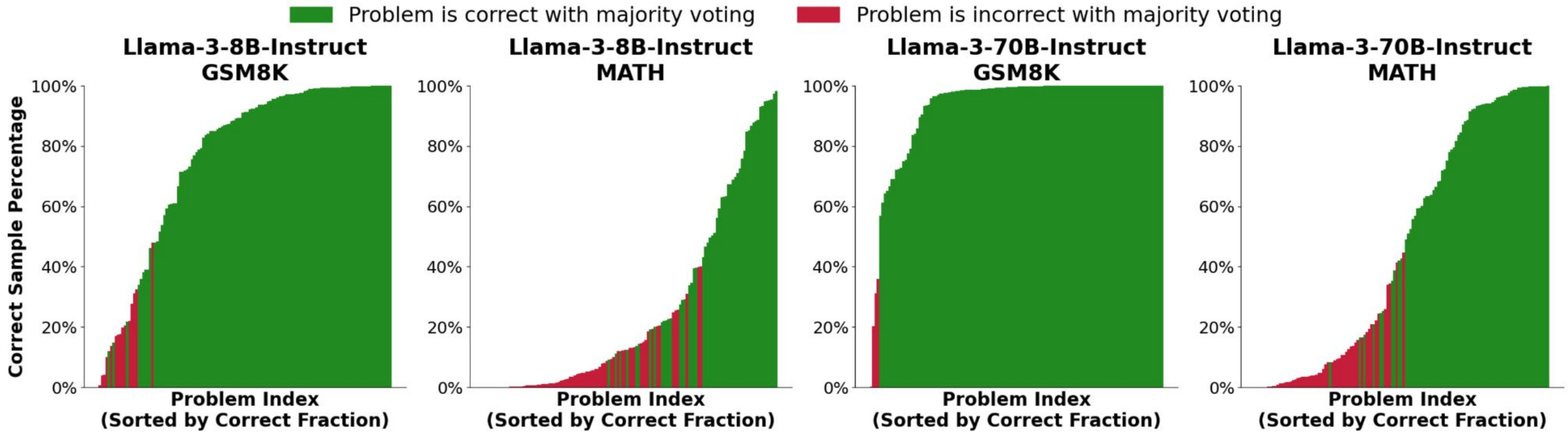


Figure 8: Bar charts showing the fraction of samples (out of 10,000 samples) that are correct, for each problem in the subsets of GSM8K and MATH we evaluate on. There is one bar per problem, and the height of the bar corresponds to the fraction of samples that arrive at the correct answer. Bars are green if self-consistency picked the correct answer and are red otherwise. We highlight that there are many problems with correct solutions, where the correct solutions are sampled infrequently.

s1: Simple test-time scaling

Niklas Muennighoff [*134] Zitong Yang [*1] Weijia Shi [*23] Xiang Lisa Li [*1] Li Fei-Fei [1] Hannaneh Hajishirzi [23]
Luke Zettlemoyer [2] Percy Liang [1] Emmanuel Candès [1] Tatsunori Hashimoto [1]

Test-time scaling with s1-32B

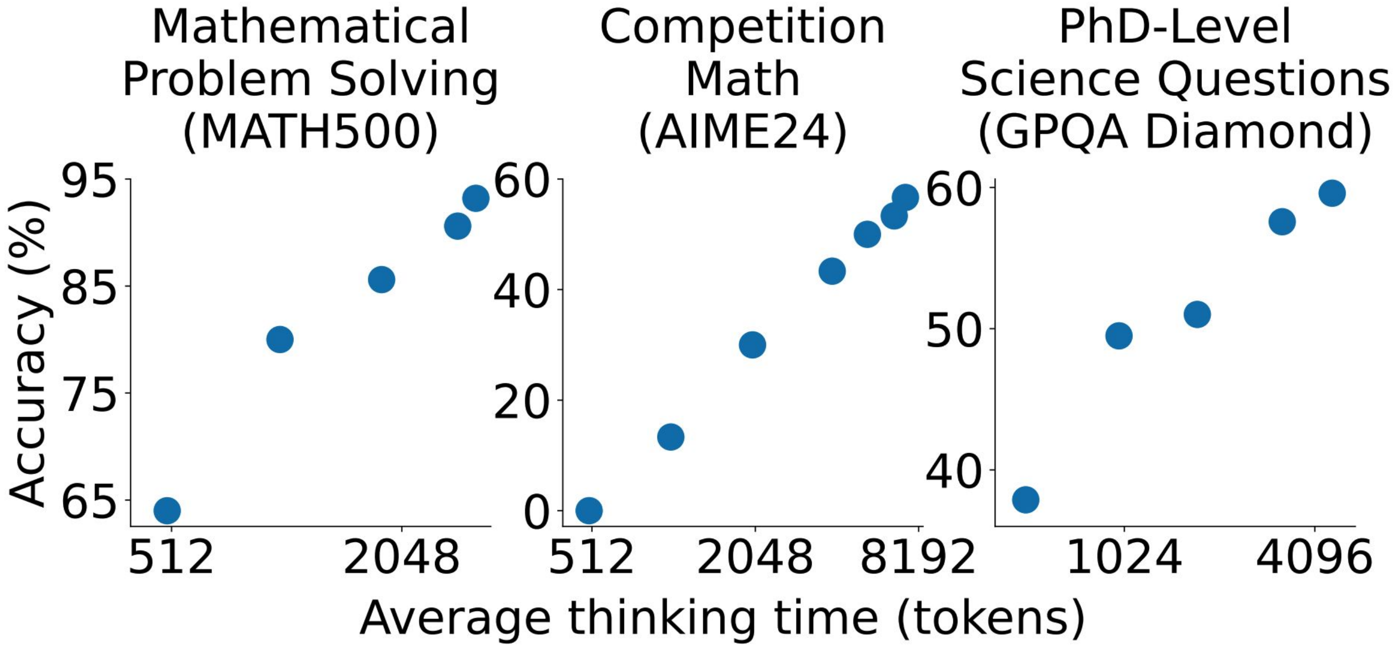
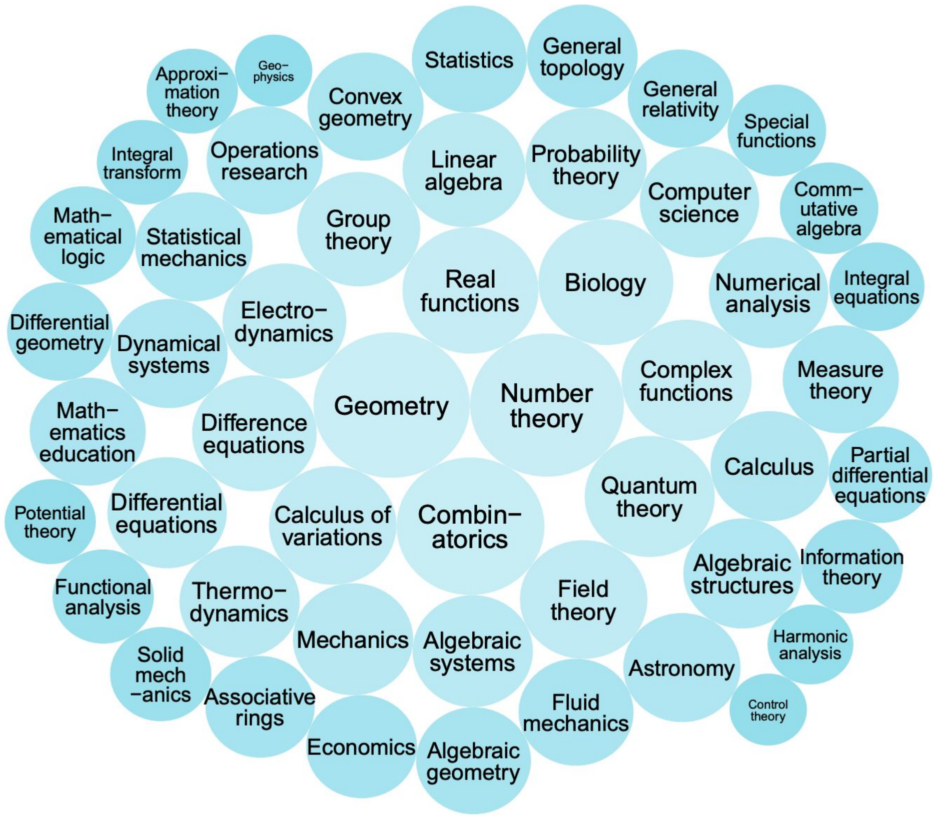
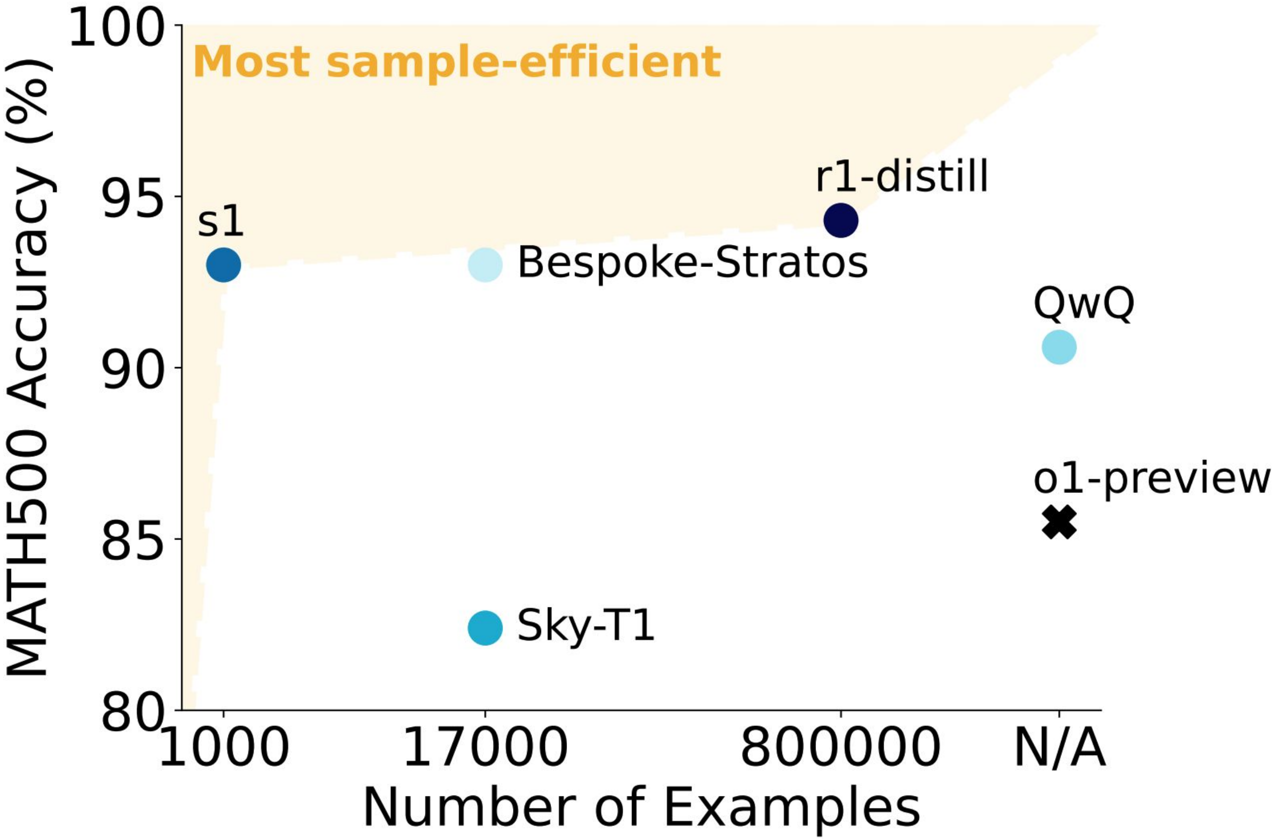


Figure 1. **Test-time scaling with s1-32B.** We benchmark **s1-32B** on reasoning-intensive tasks and vary test-time compute.

s1K is a dataset of 1,000 high-quality, diverse, and difficult questions with reasoning traces.



s1-32B is on the sample-efficiency frontier (Pareto)



Budget forcing

How many r in raspberry? **Question**

Let's break down the process of counting the letter 'r' in the word "raspberry" ...
* First letter: 'r' - This is an 'r', count = 1.
* Second letter: 'a' - Not an 'r', count remains 1 ...
* Sixth letter: 'e' - Not an 'r', count remains 1.
* Seventh letter: 'r' - This is an 'r', count = 2.
* Eighth letter: 'y' - Not an 'r', count remains 2 ...

The number of 'r's in "raspberry" is 2.

Wait, let's re-read the question carefully. It asks "How many r in raspberry?" ... * r - a - s - p - b - e - r - r - y ... * First 'r' ... * Second 'r' ... * Third 'r' ... Count = 3 ... **Reasoning trace**

My initial answer of 2 was incorrect due to a quick reading of the word. **Final Answer:** The final answer is $\boxed{3}$ **Response**

For each method under consideration, we evaluate a set of tasks $a \in \mathcal{A}$ by varying the test-time compute on a fixed benchmark (e.g., AIME24). This yields a piecewise linear function f, where the x-axis represents compute (measured in thinking tokens) and the y-axis corresponds to accuracy.

Control

$$\text{Control} = \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \mathbb{I}(a_{\min} \leq a \leq a_{\max})$$

This measures the proportion of evaluations whose test-time compute (in thinking tokens) falls within a specified range $[a_{\min}, a_{\max}]$. In practice, we often constrain only $a_{\max}$. As thinking tokens reflect test-time compute, this metric quantifies how well a method supports controllability over computational cost. We report Control as a percentage, with 100% indicating perfect control.

Scaling

$$\text{Scaling} = \frac{1}{\binom{|\mathcal{A}|}{2}} \sum_{\substack{a,b \in \mathcal{A} \\ b > a}} \frac{f(b) - f(a)}{b - a}$$

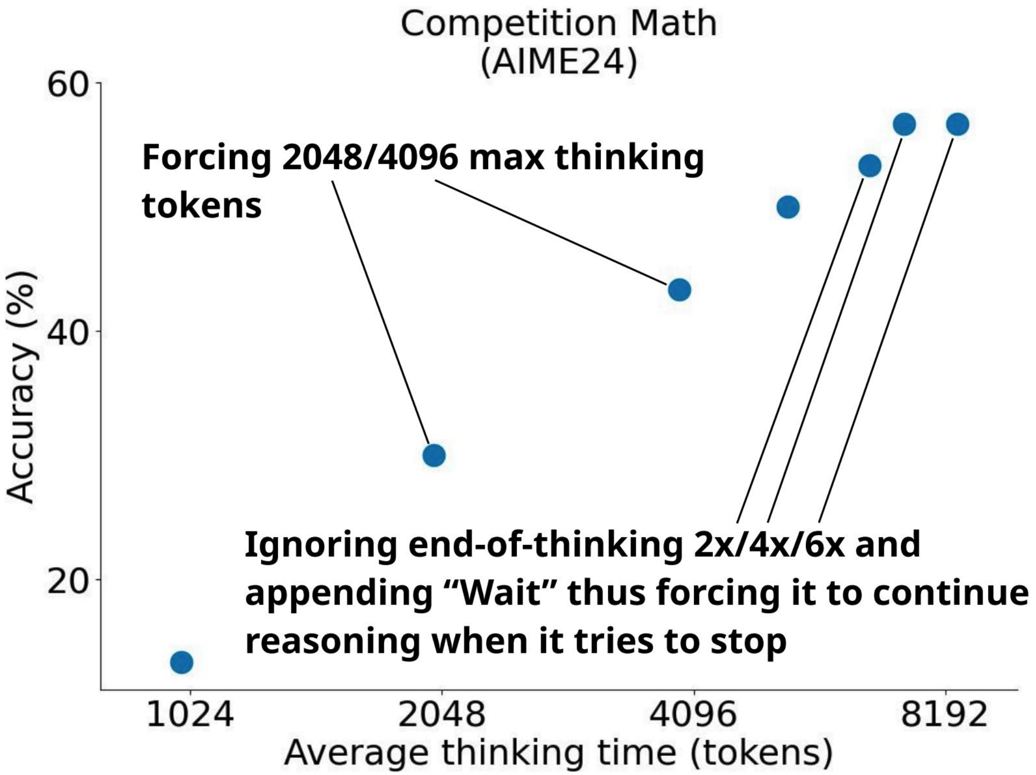
This captures the average slope of the piecewise linear function f, representing the relationship between compute and accuracy. A positive slope is necessary, and higher values indicate better scaling behavior.

Performance

$$\text{Performance} = \max_{a \in \mathcal{A}} f(a)$$

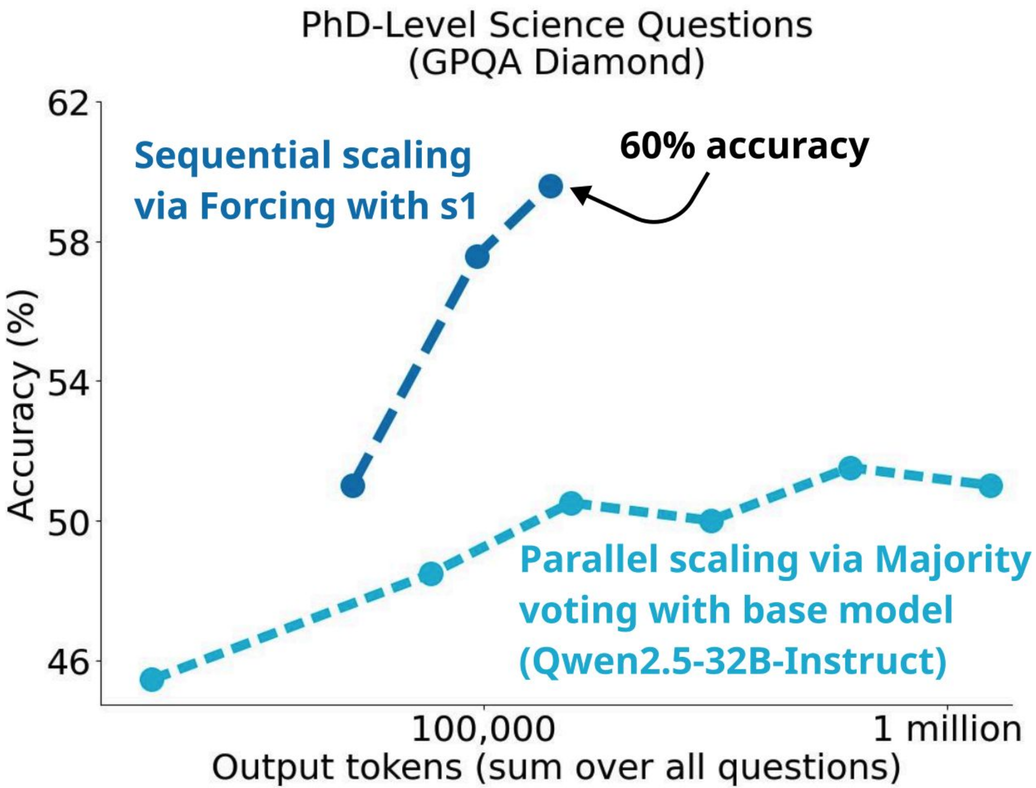
This is the highest accuracy achieved across all levels of test-time compute. In the limit, a method with consistently positive scaling will approach 100% performance.

Budget forcing shows clear scaling trends and extrapolates to some extent



(a) Sequential scaling via budget forcing

Parallel scaling via majority voting



(b) Parallel scaling via majority voting

s1-32B is a strong open reasoning model

Model	# ex.	AIME 2024	MATH 500	GPQA Diamond
API only				
o1-preview	N.A.	44.6	85.5	73.3
o1-mini	N.A.	70.0	90.0	60.0
o1	N.A.	**74.4**	**94.8**	**77.3**
Gemini 2.0 Flash Think.	N.A.	60.0	N.A.	N.A.
Open Weights				
Qwen2.5-32B-Instruct	N.A.	26.7	84.0	49.0
QwQ-32B	N.A.	50.0	90.6	54.5
r1	≫800K	**79.8**	**97.3**	**71.5**
r1-distill	800K	72.6	94.3	62.1
Open Weights and Open Data				
Sky-T1	17K	43.3	82.4	56.8
Bespoke-32B	17K	**63.3**	93.0	58.1
s1 w/o BF	**1K**	50.0	92.6	56.6
s1-32B	**1K**	56.7	**93.0**	**59.6**

s1K data ablations

Model	AIME 2024	MATH 500	GPQA Diamond
1K-random	36.7 [-26.7%, -3.3%]	90.6 [-4.8%, 0.0%]	52.0 [-12.6%, 2.5%]
1K-diverse	26.7 [-40.0%, -10.0%]	91.2 [-4.0%, 0.2%]	54.6 [-10.1%, 5.1%]
1K-longest	33.3 [-36.7%, 0.0%]	90.4 [-5.0%, -0.2%]	59.6 [-5.1%, 10.1%]
59K-full	53.3 [-13.3%, 20.0%]	92.8 [-2.6%, 2.2%]	58.1 [-6.6%, 8.6%]
s1K	50.0	93.0	57.6

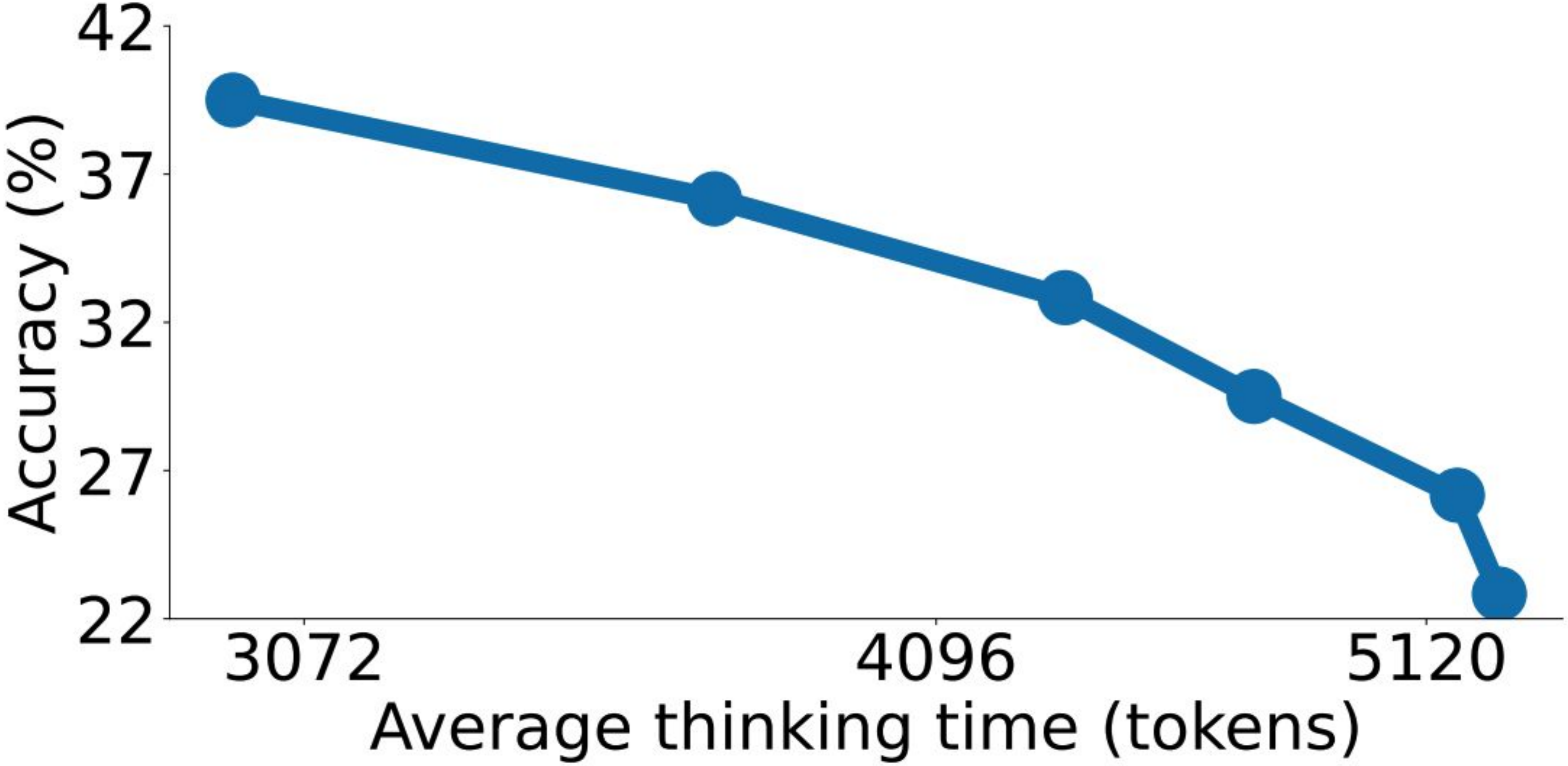
Ablations on methods to scale test-time compute

Method	Control	Scaling	Performance	$\|\mathcal{A}\|$
BF	**100%**	15	**56.7**	5
TCC	40%	-24	40.0	5
TCC + BF	**100%**	13	40.0	5
SCC	60%	3	36.7	5
SCC + BF	**100%**	6	36.7	5
CCC	50%	**25**	36.7	2
RS	**100%**	-35	40.0	5

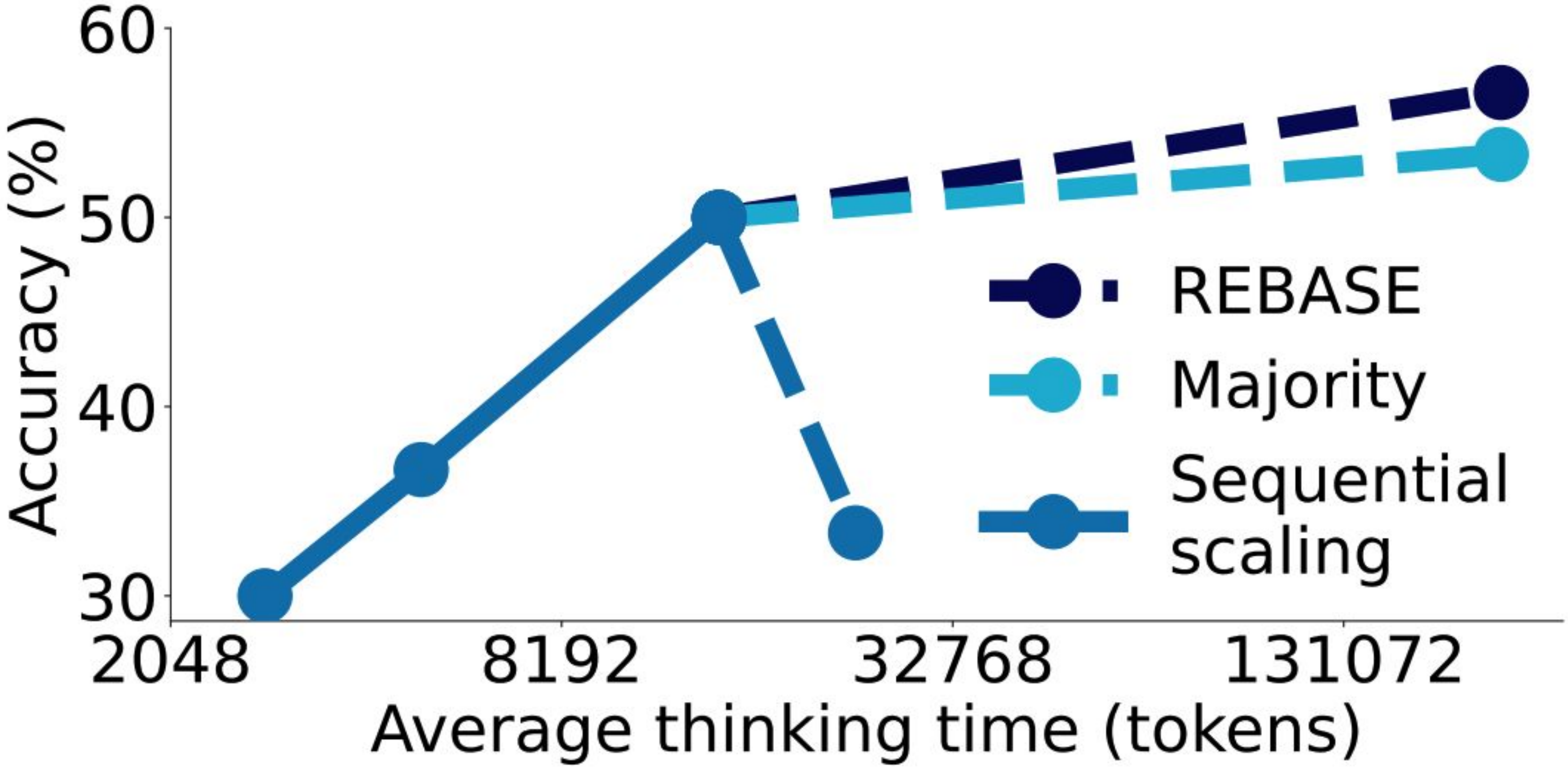
Budget forcing extrapolation ablations

Model	AIME 2024	MATH 500	GPQA Diamond
No extrapolation	50.0	**93.0**	57.6
2x without string	50.0	90.2	55.1
2x “Alternatively”	50.0	92.2	**59.6**
2x “Hmm”	50.0	**93.0**	**59.6**
2x “Wait”	**53.3**	**93.0**	**59.6**

Rejection sampling



Augmenting s1 with REBASE (process reward model)



Why does supervised fine-tuning on just 1,000 samples lead to such performance gains?

- We hypothesize that the model is already exposed to large amounts of reasoning data during pretraining which spans trillions of tokens.
- Thus, the ability to perform reasoning is already present in our model.
- Our sample-efficient fine-tuning stage just activates it and we scale it further at test time with budget forcing.

Superficial Alignment Hypothesis

- LIMA: Less is more for alignment (Zhou et al., 2023)
  - 1,000 examples can be sufficient to align a model to adhere to user preferences

Generative AI Research

LIMO: Less is More for Reasoning

Yixin Ye* Zhen Huang* Yang Xiao Ethan Chern Shijie Xia Pengfei Liu†

SJTU, SII, GAIR

Superficial Alignment Hypothesis

- LIMA: Less is more for alignment (Zhou et al., 2023)
  - 1,000 examples can be sufficient

- LIMO: even competition-level complex reasoning abilities can be effectively elicited through minimal but curated training samples

- LIMO: a promising technical pathway toward AGI - any sophisticated reasoning capability, no matter how complex, could potentially be activated with minimal samples given two key conditions:

  - (1) sufficient domain knowledge embedded during pre-training
  - (2) optimal cognitive reasoning chains for activation

Categorizing the reasoning chains into five

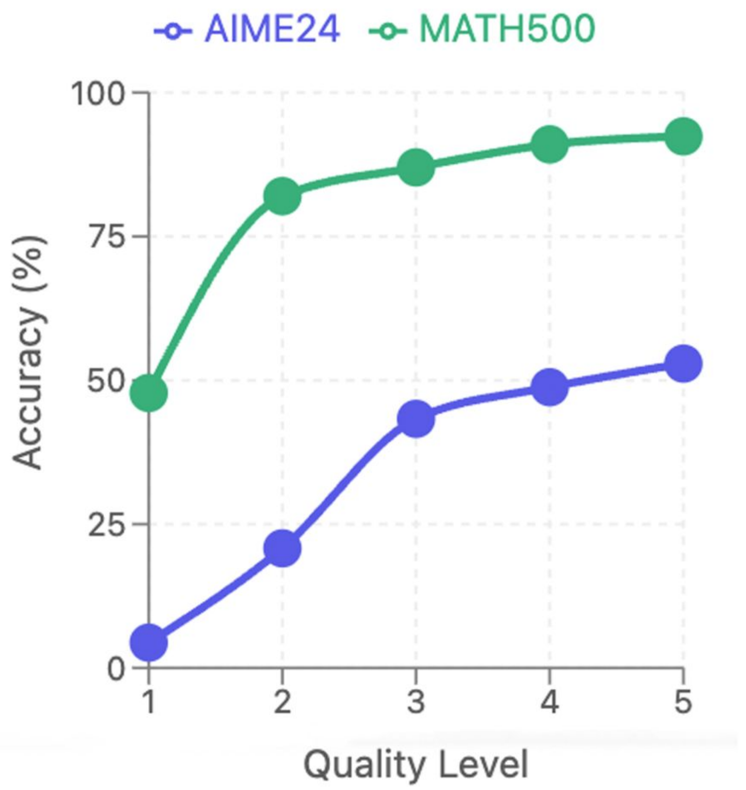
How well the reasoning steps were organized, whether important logical transitions were properly explained, and if the solution included self-verification steps

- **L5:** excellent organization with clear, well-explained steps and thorough self-verification
- **L4:** well-structured but perhaps with slightly less rigorous checking
- **L3:** decent organization but sometimes skipped over explaining crucial logical leaps
- **L2:** often provided abbreviated reasoning without much explanation
- **L1:** just listed basic steps with minimal elaboration and rarely included any verification

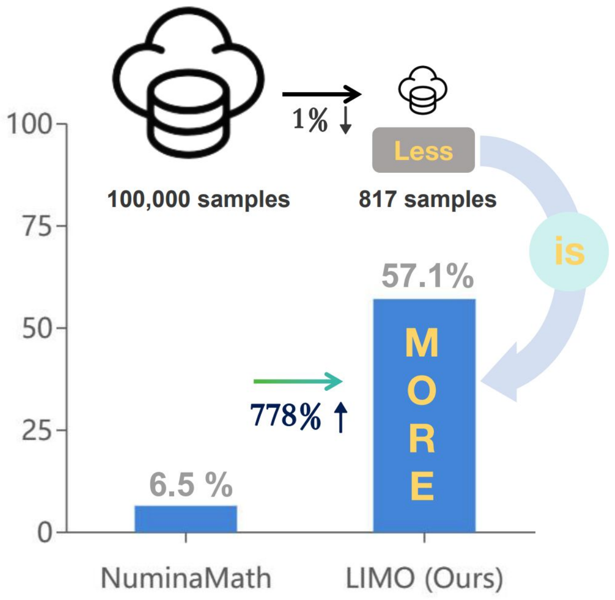
Statistical analysis of different quality levels

Data Quality Level	Avg. Tokens per response	Avg. Lines per response	Top 10 Frequently Occurring Keywords (in order)
Level 1	230	9.21	since, however, number, let, thus, which, get, two, triangle, theta
Level 2	444.88	50.68	number, need, times, which, find, list, thus, since, triangle, sum
Level 3	4956.11	375.60	**perhaps**, **alternatively**, **consider**, number, **wait**, which, sides, need, equal, seems
Level 4	4726.97	354.87	**wait**, which, number, **perhaps**, therefore, let, since, **maybe**, sides, two
Level 5	5290.26	239.29	**wait**, therefore, which, number, since, lets, two, sides, let, **maybe**

Comparison of models trained on reasoning chains of different quality levels



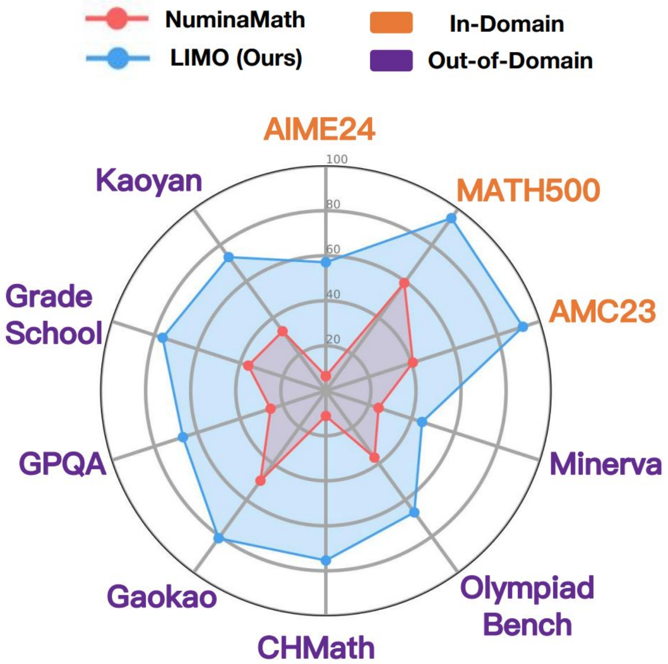
LIMO achieves substantial improvement over NuminaMath with fewer samples



completely **same** backbone

1% data → **778%** gain on AIME24 (pass@1)

… while excelling across diverse mathematical and multi-discipline benchmarks

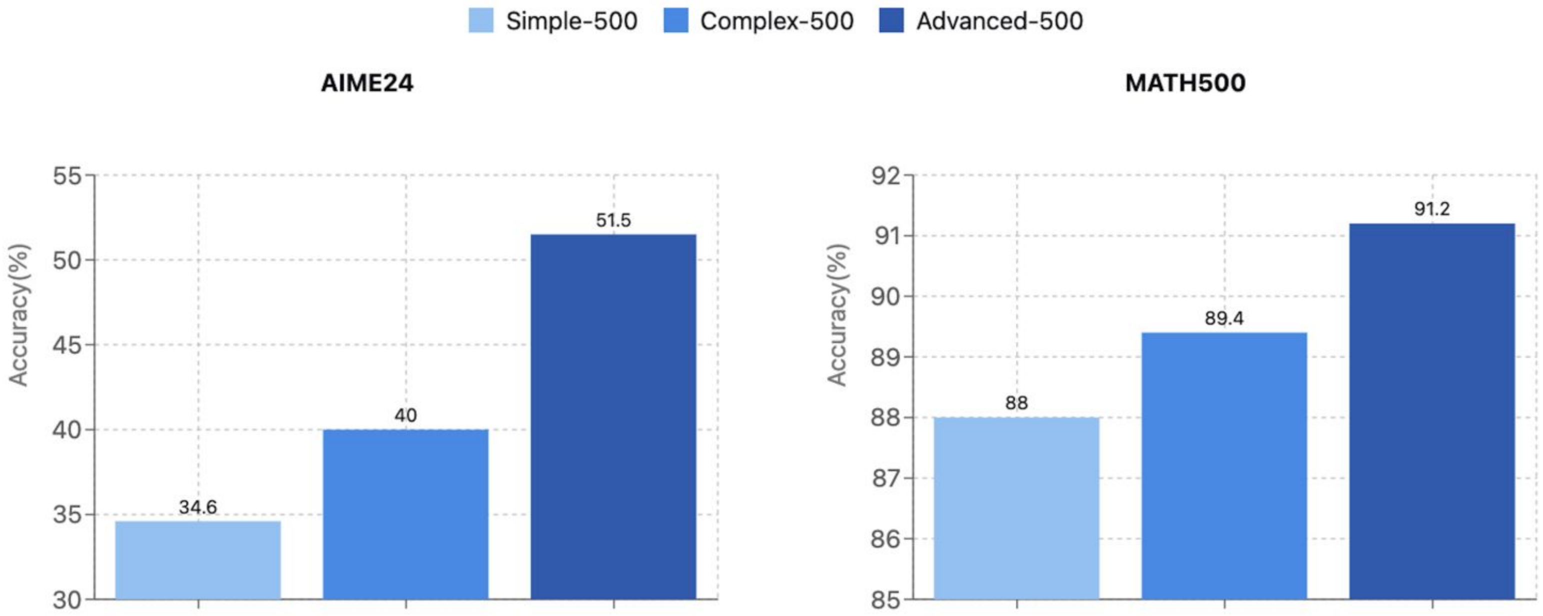


superior performance across
10 benchmarks

LIMO achieves superior performance despite using significantly fewer training examples

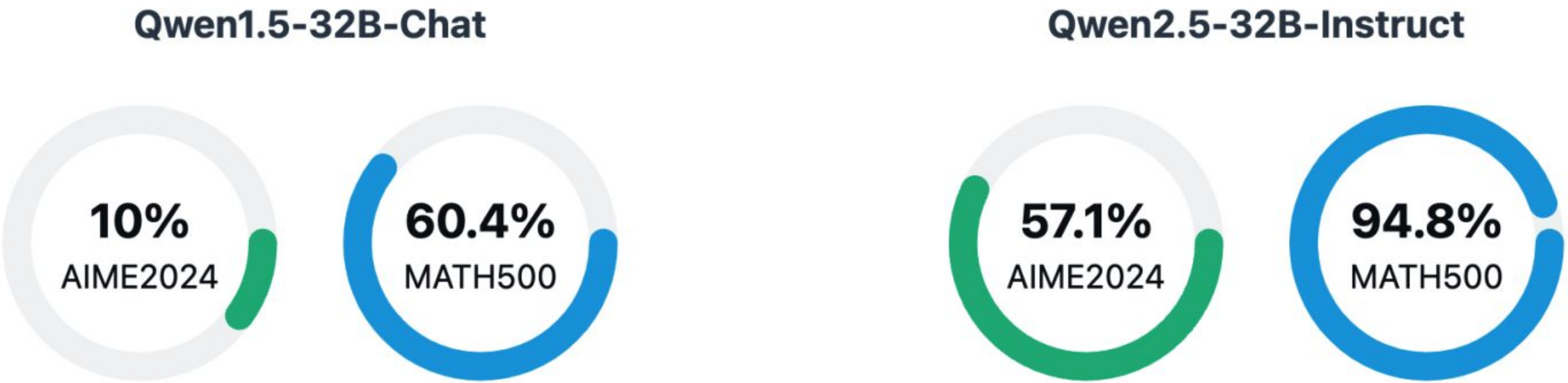
Datasets	OpenAI-o1 -preview	Qwen2.5-32B -Instruct	QwQ-32B-preview	OpenThoughts (114k)	NuminaMath (100k)	LIMO ours(817)
			In Domain			
AIME24	44.6	16.5	50.0	50.2	6.5	**57.1**
MATH500	85.5	79.4	89.8	80.6	59.2	**94.8**
AMC23	81.8	64.0	83.6	80.5	40.6	**92.0**
			Out of Domain			
OlympiadBench	52.1	45.3	58.5	56.3	36.7	**66.8**
CHMath	50.0	27.3	68.5	74.1	11.2	**75.4**
Gaokao	62.1	72.1	80.1	63.2	49.4	**81.0**
Kaoyan	51.5	48.2	70.3	54.7	32.7	**73.4**
GradeSchool	62.8	56.7	63.8	39.0	36.2	**76.2**
Minerva	**47.1**	41.2	39.0	41.1	24.6	44.9
GPQA	**73.3**	48.0	65.1	42.9	25.8	66.7
AVG.	61.1	49.9	66.9	58.3	32.3	**72.8**

Models trained on different question quality

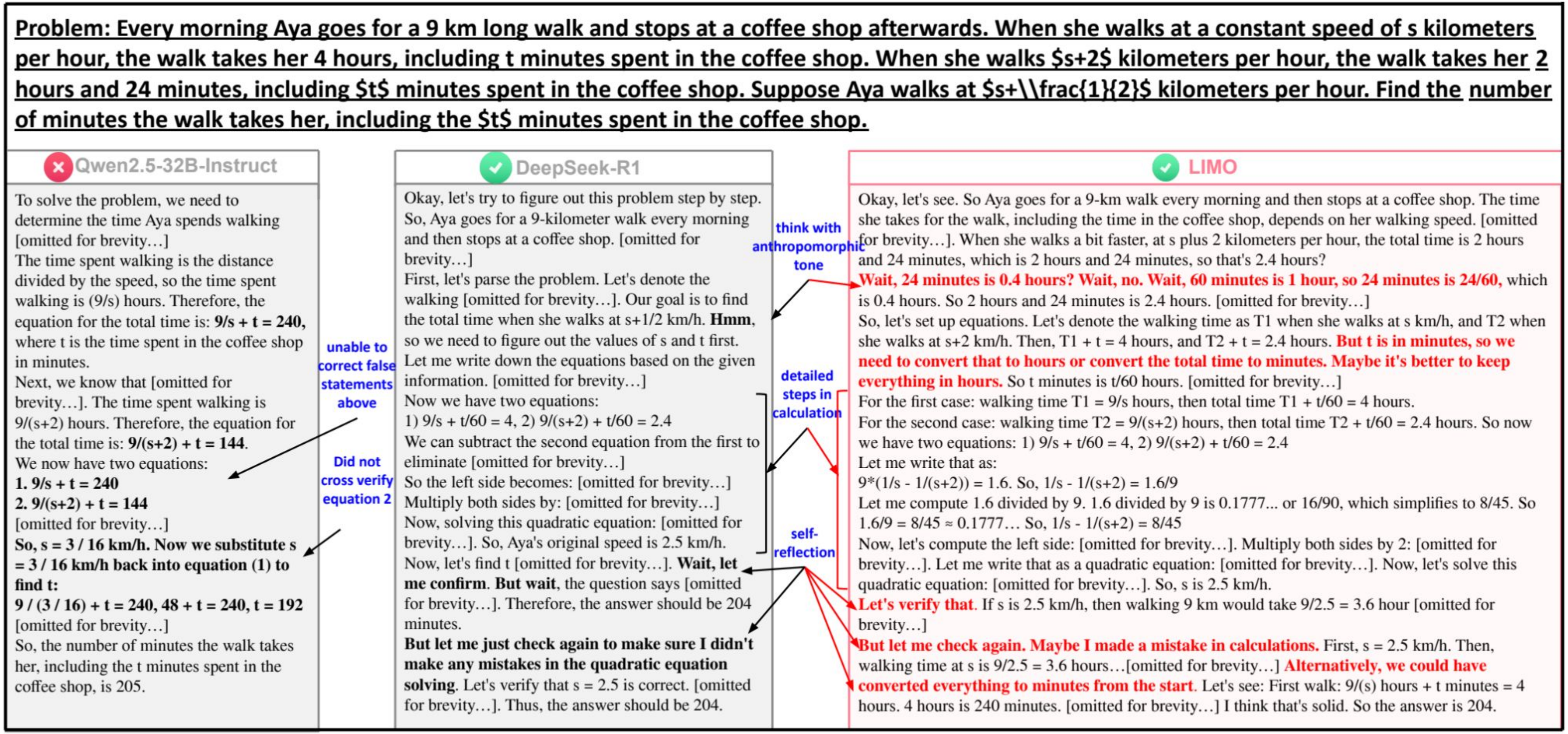


- **Simple-500**: 500 simple problems randomly selected problems from MATH levels 1 and 2.
- **Complex-500**: 500 complex problems randomly selected problems from MATH levels 3, 4, and 5.
- **Advanced-500**: 500 advanced problems randomly selected problems from past AIME tests.

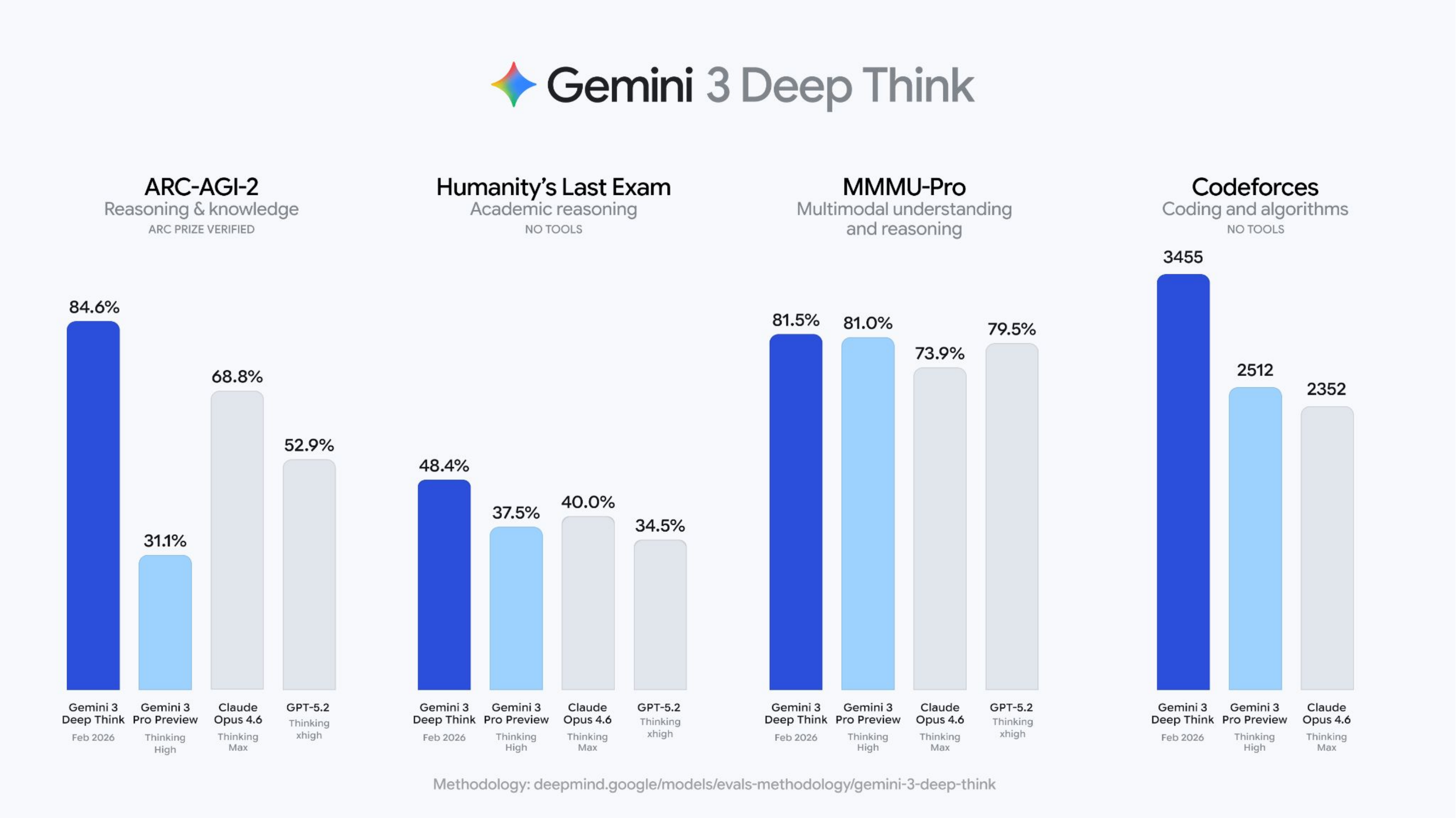
Impact of pretrained model choice



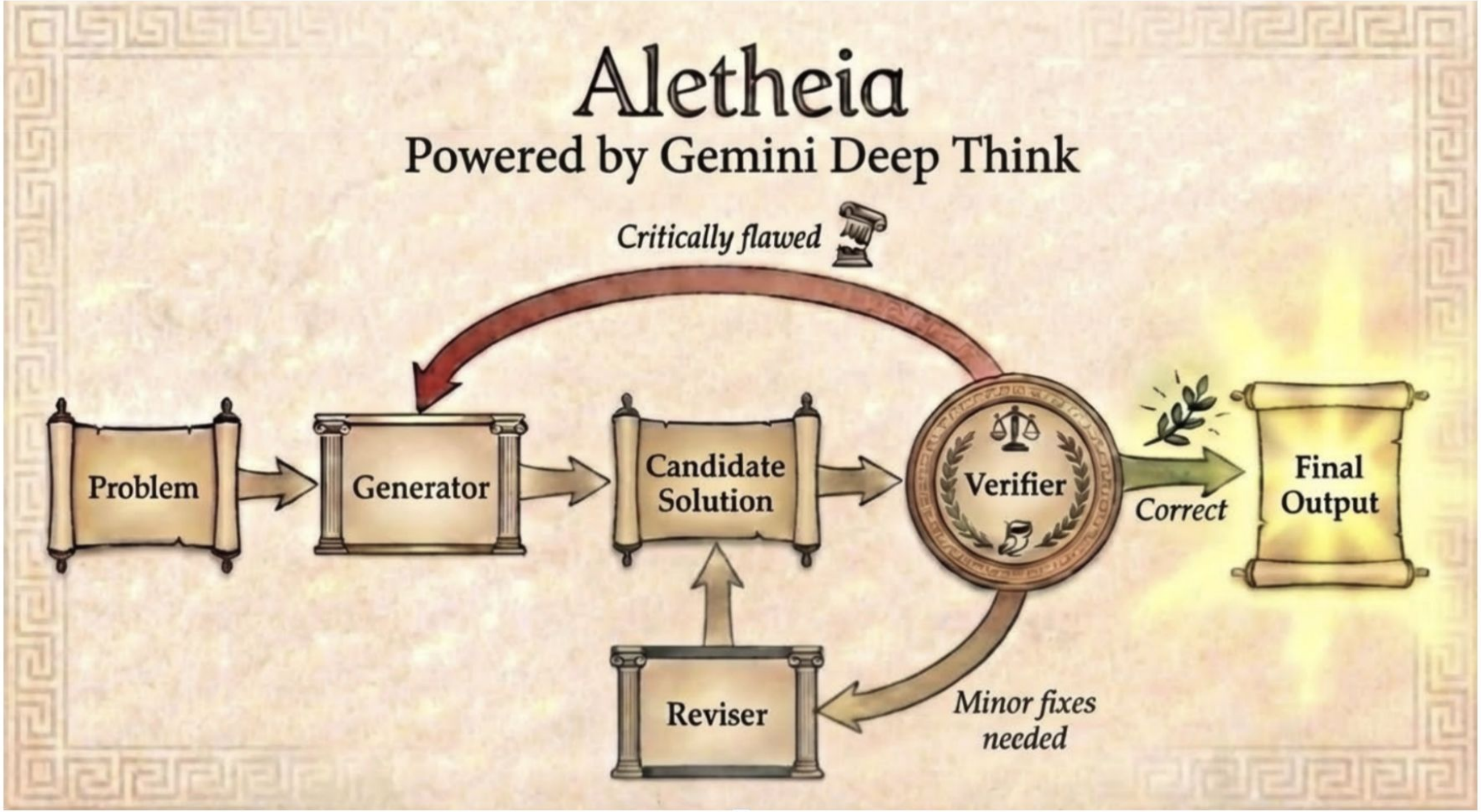
Example model outputs



Gemini Deep Think



Gemini Deep Think



PRISM: Pushing the Frontier of Deep Think via Process Reward Model-Guided Inference

Rituraj Sharma[*] **Weiyuan Chen**[*] **Noah Provenzano** **Tu Vu**

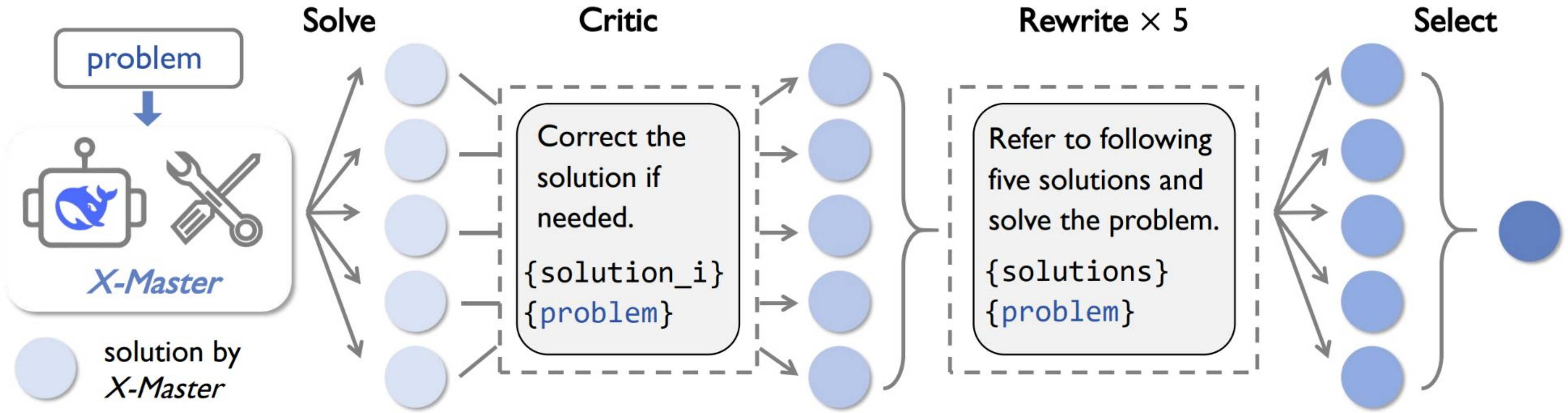
Virginia Tech

{rituraj,cweiyuan9,noahpro,tuvu}@vt.edu

https://github.com/Rituraj003/PRISM/

Abstract

DEEPTHINK methods improve reasoning by generating, refining, and aggregating populations of candidate solutions, which enables strong performance on complex mathematical and scientific tasks. However, existing frameworks often lack reliable correctness signals during inference, which creates a population-enhancement bottleneck where deeper deliberation amplifies errors, suppresses correct minority solutions, and yields weak returns to additional compute. In this paper, we introduce a functional decomposition of DEEPTHINK systems and propose PRISM, a Process Reward Model (PRM)-guided inference algorithm that uses step-level verification to guide both population refinement and solution aggregation. During refinement, PRISM treats candidate solutions as particles in a PRM-defined energy landscape and reshapes the population through score-guided resampling and stochastic refinement, which concentrates probability mass on higher-quality reasoning while preserving diversity. Across mathematics and science benchmarks, PRISM is competitive with or outperforms existing DEEPTHINK methods, reaching 90.0%, 75.4%, and 71.4% with `gpt-oss-20b` on AIME25, HMMT25, and GPQA Diamond, respectively, while matching or exceeding `gpt-oss-120b`. Additionally, our analysis shows that PRISM produces consistent net-directional correction during refinement, remains reliable when the initial population contains few correct candidates, and often lies on the compute-accuracy Pareto frontier.



[SciMaster: Towards General-Purpose Scientific AI Agents]

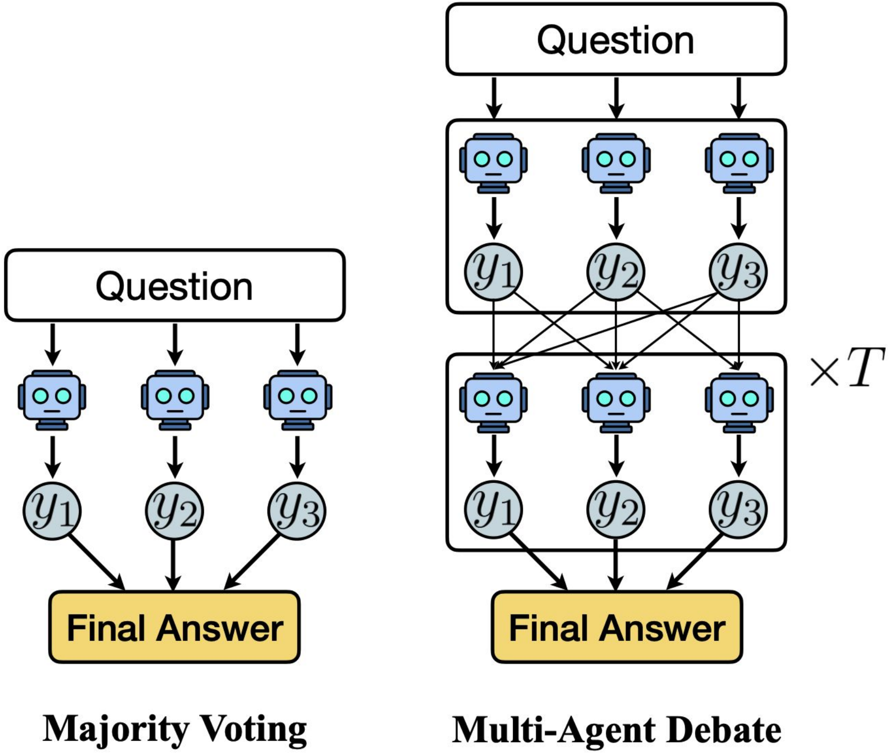
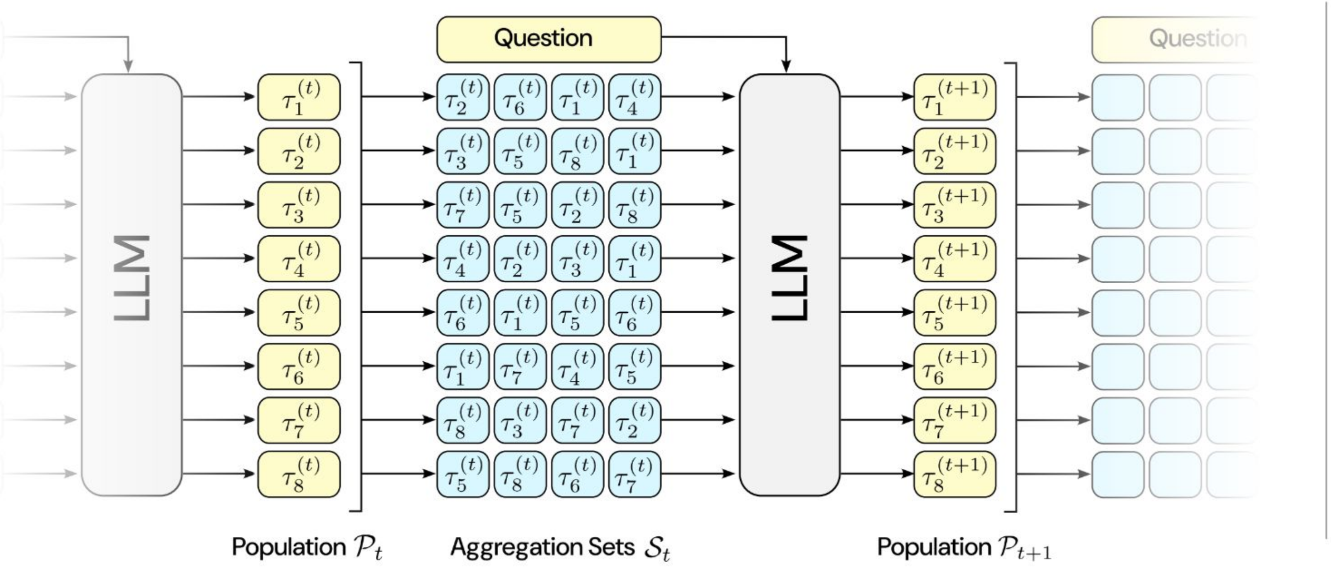


Figure 1: Majority Voting vs. MAD overview.

[Debate or Vote: Which Yields Better Decisions in Multi-Agent Large Language Models?]()

Recursive Self-Aggregation (RSA)



Aggregation Prompt

You are given a problem and several candidate solutions. Some candidates may be incorrect or contain errors. Aggregate the useful ideas and produce a single, high-quality solution. Reasoning carefully; if candidates disagree, choose the correct path. If all are incorrect, then attempt a different strategy.

End with the final result in <format>.

Problem: <problem>

Candidate solutions (may contain mistakes):

---------- Solution 1 ----------

<solution 1>

---------- Solution 2 ----------

<solution 2>

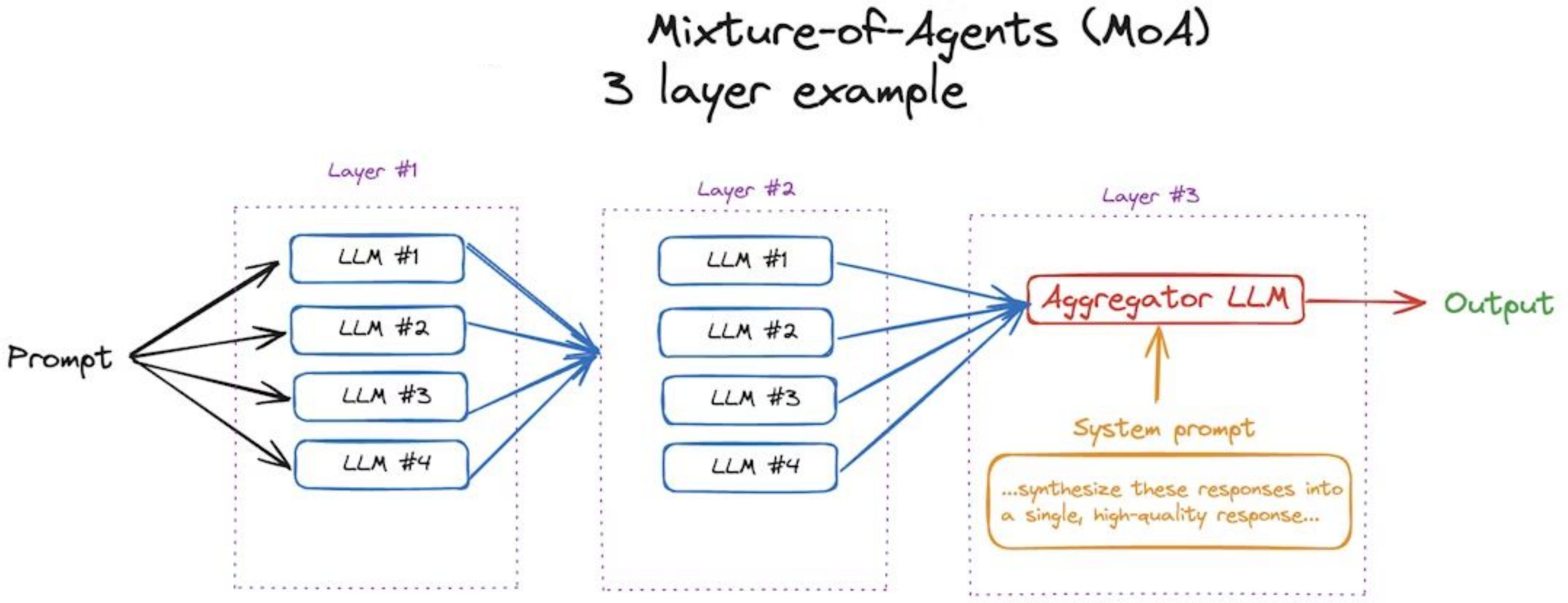
...

---------- Solution K ----------

<solution K>

[Recursive Self-Aggregation Unlocks Deep Thinking in Large Language Models]

Deep Think / Parallel Agents / Mixture-of-Agents

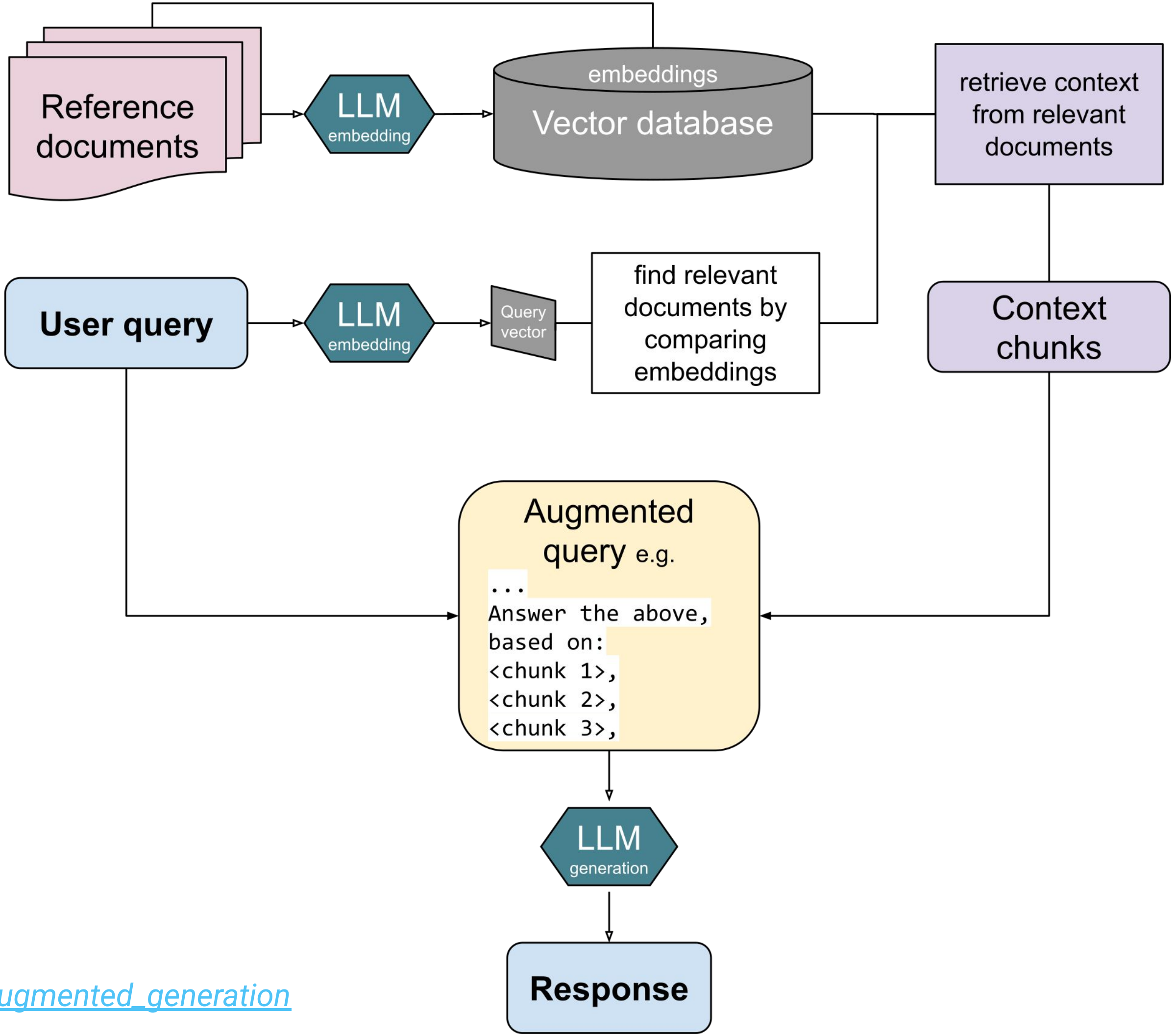


https://docs.together.ai/docs/mixture-of-agents#advanced-moa-example

Retrieval-Augmented Generation (RAG)

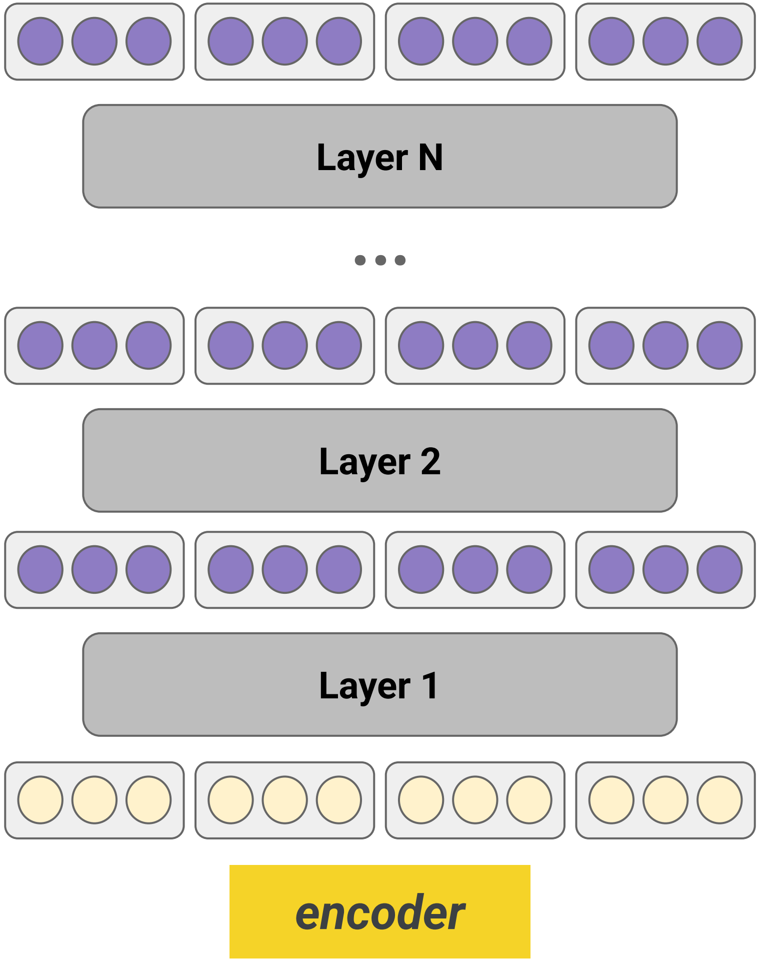
- Vanilla RAG
  - E.g., RAG, REALM
- RAG++
  - E.g., FreshLLMs, ReAct, Toolformer
- RAG + reasoning, agentic RAG, agentic memory
  - E.g., Self-RAG, Search-R1, Deep Research, A-Mem

Vanilla RAG

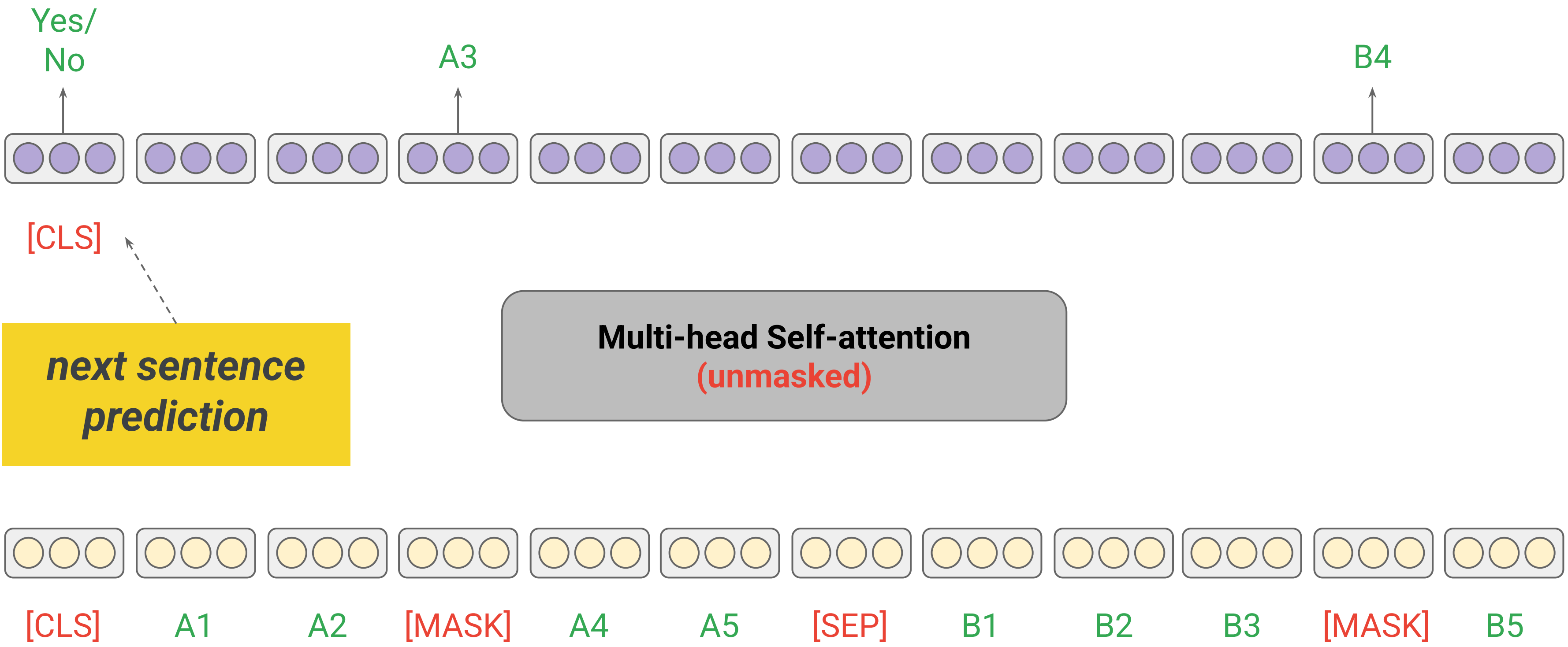


https://en.wikipedia.org/wiki/Retrieval-augmented_generation

Encoder
(N layers)



BERT encoder



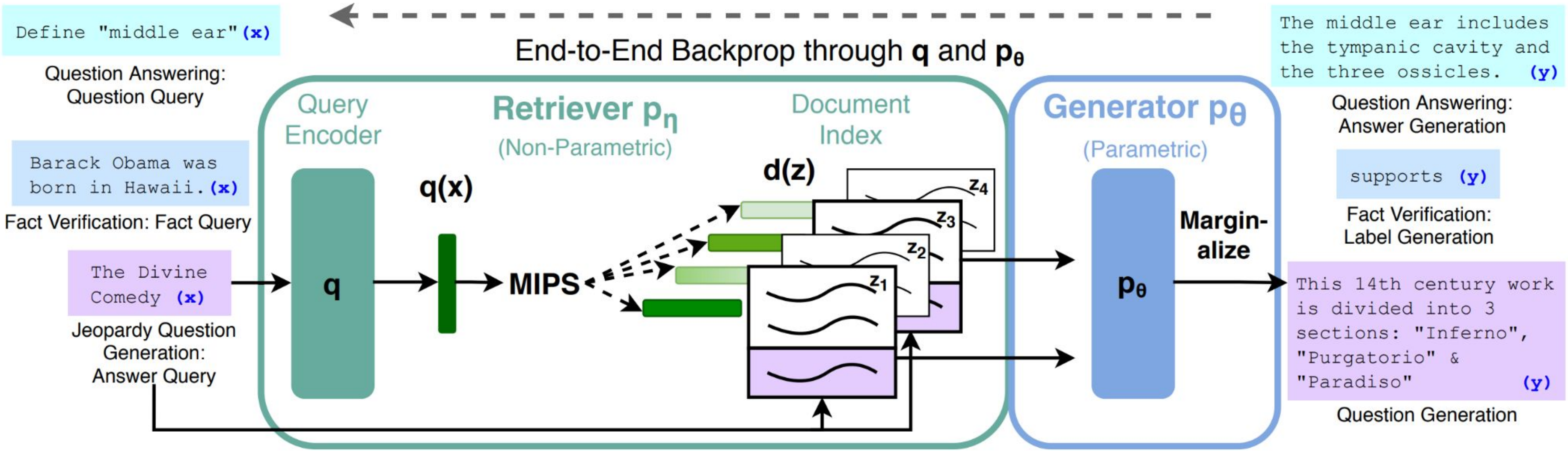
Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

Patrick Lewis[†‡], **Ethan Perez**[⋆],

Aleksandra Piktus[†], **Fabio Petroni**[†], **Vladimir Karpukhin**[†], **Naman Goyal**[†], **Heinrich Küttler**[†],

Mike Lewis[†], **Wen-tau Yih**[†], **Tim Rocktäschel**[†‡], **Sebastian Riedel**[†‡], **Douwe Kiela**[†]

[†]Facebook AI Research; [‡]University College London; [⋆]New York University;
plewis@fb.com



Define "middle ear" (x)
Question Answering: Question Query
Barack Obama was born in Hawaii. (x)
Fact Verification: Fact Query
The Divine Comedy (x)
Jeopardy Question Generation: Answer Query
End-to-End Backprop through q and pθ
Query Encoder
Retriever pη
(Non-Parametric)
Document Index
q
q(x)
MIPS
d(z)
z4
z3
z2
z1
Generator pθ
(Parametric)
pθ
Margin-alize
The middle ear includes the tympanic cavity and the three ossicles. (y)
Question Answering: Answer Generation
supports (y)
Fact Verification: Label Generation
This 14th century work is divided into 3 sections: "Inferno", "Purgatorio" & "Paradiso" (y)
Question Generation

Component	Type	Role	Trainable?
Index	Data structure	Stores document embeddings for nearest-neighbor search	No
Retriever	Model	Computes similarity between query and documents; retrieves top-K	Yes (query encoder usually)
Query encoder	Neural network	Encodes the input text into an embedding	Yes
Document encoder	Neural network	Encodes documents into embeddings used to build the index	Sometimes frozen
Generator	Seq2seq model	Generates text conditioned on the input and retrieved docs	Yes

Dense Passage Retrieval for Open-Domain Question Answering

Vladimir Karpukhin,[*] Barlas Oğuz,[*] Sewon Min[†], Patrick Lewis,
Ledell Wu, Sergey Edunov, Danqi Chen[‡], Wen-tau Yih
Facebook AI [†]University of Washington [‡]Princeton University
{vladk, barlaso, plewis, ledell, edunov, scottyih}@fb.com
sewon@cs.washington.edu
danqic@cs.princeton.edu

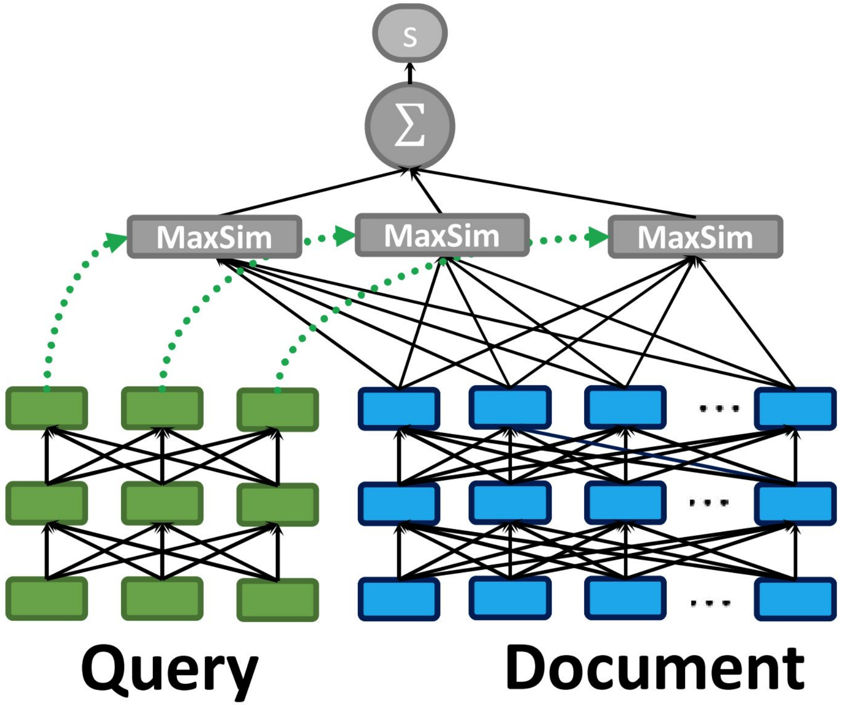
Dense Passage Retrieval (DPR)

Let $\mathcal{D} = \{(q_i, p_i^+, p_{i,1}^-, \ldots, p_{i,n_i}^-)\}_{i=1}^m$ denote the training set that contains m instances. Each instance includes a question q_i, one relevant (positive) passage p_i^+, and n_i irrelevant (negative) passages $p_{i,j}^-$.

The loss function is optimized as the negative log-likelihood of the positive passage:

$$\mathcal{L}(q_i, p_i^+, p_{i,1}^-, \ldots, p_{i,n_i}^-) = -\log \frac{\exp\big(\mathrm{sim}(q_i, p_i^+)\big)}{\exp\big(\mathrm{sim}(q_i, p_i^+)\big) + \sum_{j=1}^{n_i} \exp\big(\mathrm{sim}(q_i, p_{i,j}^-)\big)}.$$

ColBert's late interaction



(d) Late Interaction
(i.e., the proposed ColBERT)

https://arxiv.org/abs/2004.12832

Google DeepMind

On the Theoretical Limitations of Embedding-Based Retrieval

Orion Weller[*,1,2], **Michael Boratko**[1], **Iftekhar Naim**[1] **and Jinhyuk Lee**[1]
[1]Google DeepMind, [2]Johns Hopkins University

FreshLLMs: Refreshing Large Language Models with Search Engine Augmentation

Tu Vu[1] **Mohit Iyyer**[2] **Xuezhi Wang**[1] **Noah Constant**[1] **Jerry Wei**[1]
Jason Wei[3*] **Chris Tar**[1] **Yun-Hsuan Sung**[1] **Denny Zhou**[1] **Quoc Le**[1] **Thang Luong**[1]

Google[1] University of Massachusetts Amherst[2] OpenAI[3]
freshllms@google.com

FreshPrompt

source: {source_webpage}
date: {publication_date}
title: {title}
snippet: {text_snippet}
highlight:
{highlighted_words}

{demonstrations} # details omitted for brevity

query: {question}
{retrieved_evidences} # chronological order
question: {question}
answer: {reasoning_and_answer}

FreshPrompt uses few-shot in-context learning to teach a model to reason over retrieved evidences and figure out the right answer

Toolify.ai Products Category Ranking AI Models Social Listening New Submit & Advertise

Home / AI News / Unleash the Power of Perplexity AI's Fresh Prompt Approach

Unleash the Power of Perplexity AI's Fresh Prompt Approach

Updated on Feb 28,2024

THENEWSTACK

PODCASTS EBOOKS EVENTS NEWSLETTER CONTRIBUTE

ARCHITECTURE ENGINEERING OPERATIONS PROGRAMMING

Perplexity AI's Fresh Prompt Approach

AI / LARGE LANGUAGE MODELS

How Perplexity's Online LLM Was Inspired by FreshLLMs Paper

We dig into the technology behind Perplexity's Copilot, which was inspired by the FreshLLMs paper that proposed search engine-augmented LLMs.

Jan 24th, 2024 4:00am by Janakiram MSV

e

Online LLMs

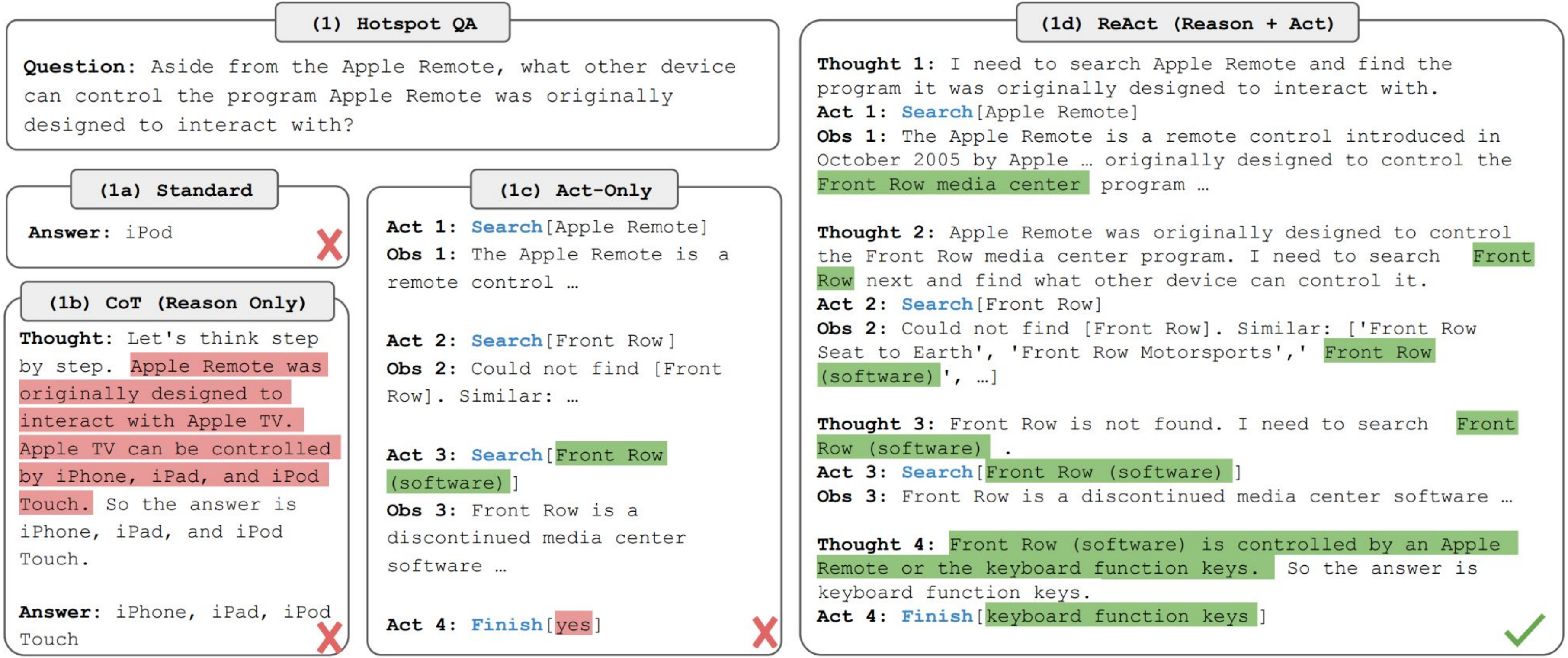
lels

ReAct: Synergizing Reasoning and Acting in Language Models

Shunyu Yao[*,1], Jeffrey Zhao[2], Dian Yu[2], Nan Du[2], Izhak Shafran[2], Karthik Narasimhan[1], Yuan Cao[2]

[1]Department of Computer Science, Princeton University
[2]Google Research, Brain team
[1]{shunyuy,karthikn}@princeton.edu
[2]{jeffreyzhao,dianyu,dunan,izhak,yuancao}@google.com

ReAct


(1) Hotspot QA
Question: Aside from the Apple Remote, what other device can control the program Apple Remote was originally designed to interact with?
(1a) Standard
Answer: iPod
(1b) CoT (Reason Only)
Thought: Let's think step by step. Apple Remote was originally designed to interact with Apple TV. Apple TV can be controlled by iPhone, iPad, and iPod Touch. So the answer is iPhone, iPad, and iPod Touch.
Answer: iPhone, iPad, iPod Touch
(1c) Act-Only
Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control …
Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: …
Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software …
Act 4: Finish[yes]
(1d) ReAct (Reason + Act)
Thought 1: I need to search Apple Remote and find the program it was originally designed to interact with.
Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control introduced in October 2005 by Apple … originally designed to control the Front Row media center program …
Thought 2: Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it.
Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports',' Front Row (software) ', …]
Thought 3: Front Row is not found. I need to search Front Row (software) .
Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software …
Thought 4: Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.
Act 4: Finish[keyboard function keys]

Toolformer: Language Models Can Teach Themselves to Use Tools

Timo Schick Jane Dwivedi-Yu Roberto Dessì[†] Roberta Raileanu
Maria Lomeli Luke Zettlemoyer Nicola Cancedda Thomas Scialom
Meta AI Research [†]Universitat Pompeu Fabra

Exemplary predictions of Toolformer

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.

Using in-context learning to generate API calls

Use an LLM to annotate a huge language modeling dataset with potential API calls

Your task is to add calls to a Question Answering API to a piece of text. The questions should help you get information required to complete the text. You can call the API by writing "[QA(question)]" where "question" is the question you want to ask. Here are some examples of API calls:

Input: Joe Biden was born in Scranton, Pennsylvania.

Output: Joe Biden was born in **[QA("Where was Joe Biden born?")]** Scranton, **[QA("In which state is Scranton?")]** Pennsylvania.

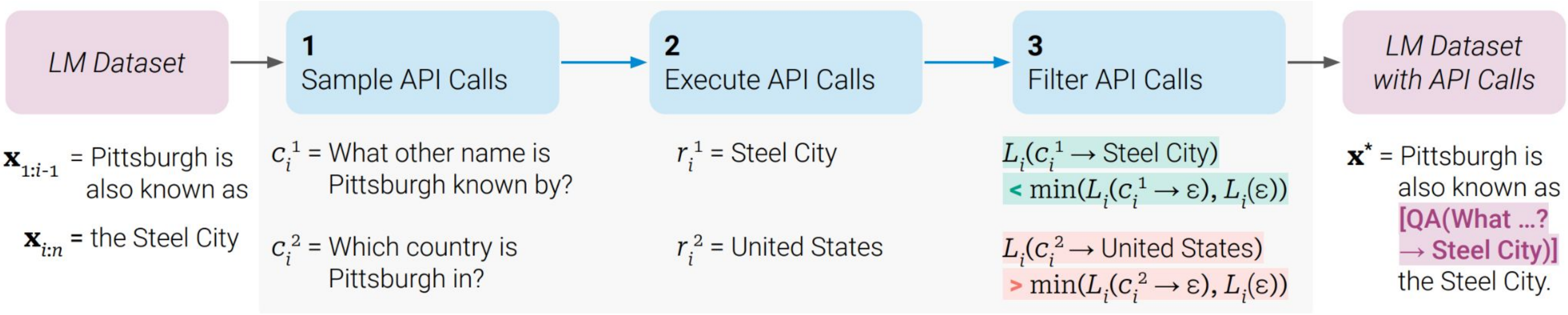
Input: Coca-Cola, or Coke, is a carbonated soft drink manufactured by the Coca-Cola Company.

Output: Coca-Cola, or **[QA("What other name is Coca-Cola known by?")]** Coke, is a carbonated soft drink manufactured by **[QA("Who manufactures Coca-Cola?")]** the Coca-Cola Company.

Input: x

Output:

Filtering out all API calls which do not reduce the loss over the next tokens



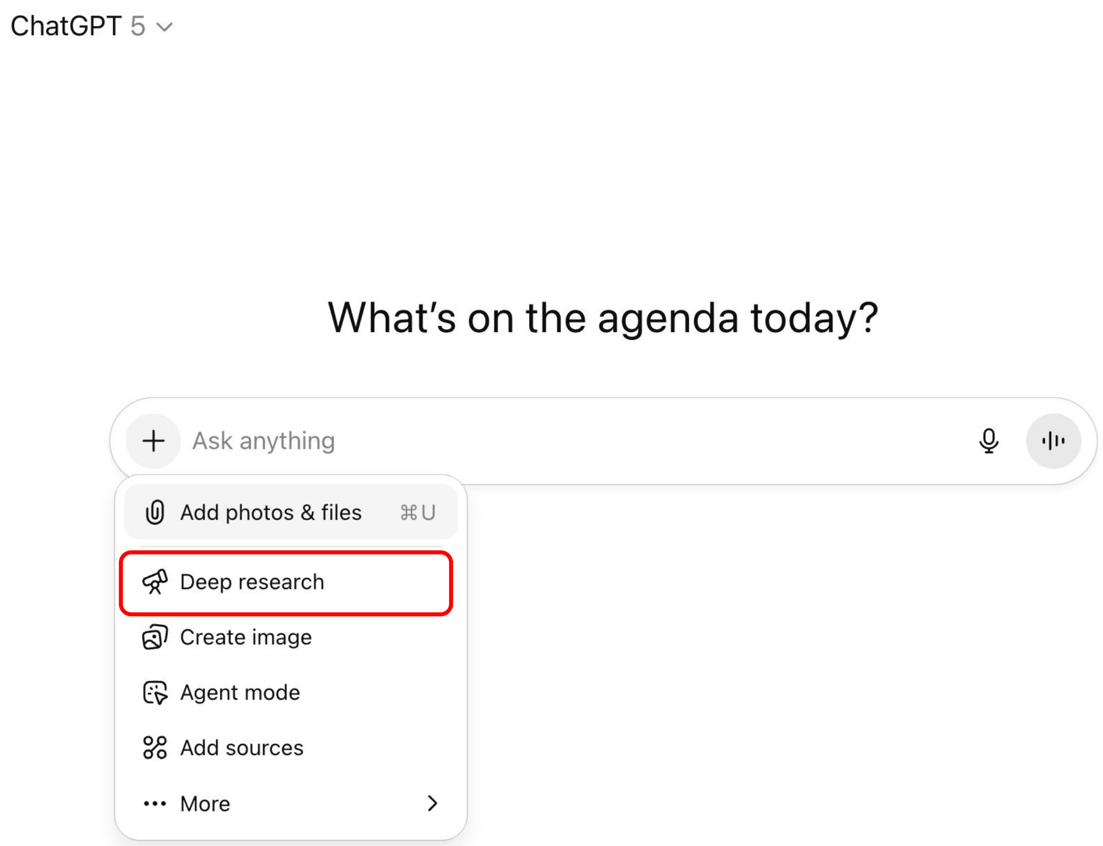
Intuitively, an API call is helpful if providing it with both the input and the output of this call makes it easier for the model to predict future tokens, compared to not receiving the API call at all, or receiving only its input

Toolformer (6.7B) achieves much stronger zero-shot results than OPT (66B) and GPT-3 (175B)

Model	ASDiv	SVAMP	MAWPS
GPT-J	7.5	5.2	9.9
GPT-J + CC	9.6	5.0	9.3
Toolformer (disabled)	14.8	6.3	15.0
Toolformer	**40.4**	**29.4**	**44.0**
OPT (66B)	6.0	4.9	7.9
GPT-3 (175B)	14.0	10.0	19.8

Model	WebQS	NQ	TriviaQA
GPT-J	18.5	12.8	43.9
GPT-J + CC	18.4	12.2	45.6
Toolformer (disabled)	18.9	12.6	46.7
Toolformer	**26.3**	**17.7**	**48.8**
OPT (66B)	18.6	11.4	45.7
GPT-3 (175B)	29.0	22.6	65.9

Deep Research



Deep Research (cont'd)

- https://openai.com/index/introducing-deep-research/
- An agent that uses reasoning to synthesize large amounts of online information and complete multi-step research tasks for you.

Deep Research (cont'd)

https://openai.com/index/introducing-deep-research/

Deep research is OpenAI's next agent that can do work for you independently—you give it a prompt, and ChatGPT will find, analyze, and synthesize hundreds of online sources to create a comprehensive report at the level of a research analyst. Powered by a version of the upcoming OpenAI o3 model that's optimized for web browsing and data analysis, it leverages reasoning to search, interpret, and analyze massive amounts of text, images, and PDFs on the internet, pivoting as needed in reaction to information it encounters.

Deep Research (cont'd)

Deep research is built for people who do intensive knowledge work in areas like finance, science, policy, and engineering and need thorough, precise, and reliable research. It can be equally useful for discerning shoppers looking for hyper-personalized recommendations on purchases that typically require careful research, like cars, appliances, and furniture. Every output is fully documented, with clear citations and a summary of its thinking, making it easy to reference and verify the information. It is particularly effective at finding niche, non-intuitive information that would require browsing numerous websites. Deep research frees up valuable time by allowing you to offload and expedite complex, time-intensive web research with just one query.

Deep Research (cont'd)

Deep research independently discovers, reasons about, and consolidates insights from across the web. To accomplish this, it was trained on real-world tasks requiring browser and Python tool use, using the same reinforcement learning methods behind OpenAI o1, our first reasoning model. While o1 demonstrates impressive capabilities in coding, math, and other technical domains, many real-world challenges demand extensive context and information gathering from diverse online sources. Deep research builds on these reasoning capabilities to bridge that gap, allowing it to take on the types of problems people face in work and everyday life.

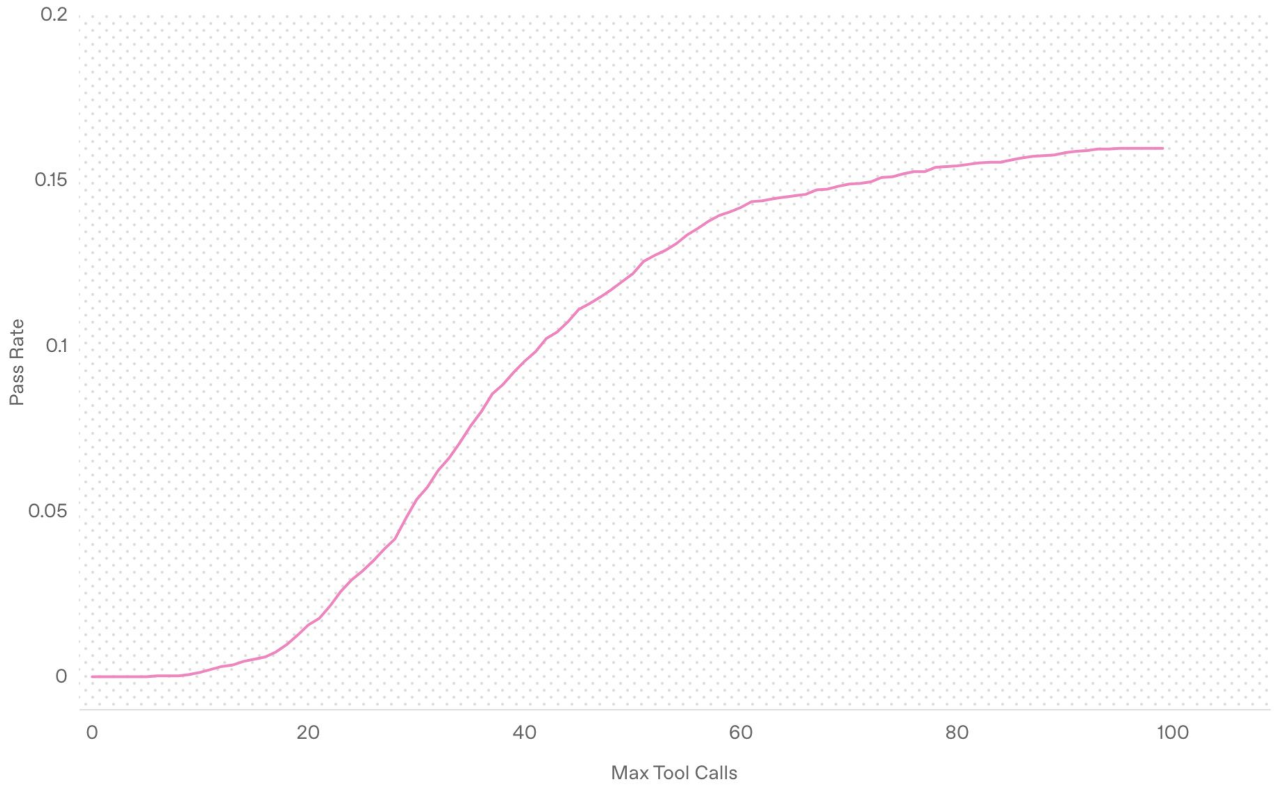
Deep Research on Humanity's Last Exam

Model	Accuracy (%)
GPT-4o	3.3
Grok-2	3.8
Claude 3.5 Sonnet	4.3
Gemini Thinking	6.2
OpenAI o1	**9.1**
DeepSeek-R1*	9.4
OpenAI o3-mini (medium)*	10.5
OpenAI o3-mini (high)*	13.0
OpenAI deep research**	**26.6**

* Model is not multi-modal, evaluated on text-only subset.

**with browsing + python tools

The more the model browses and thinks about what it is browsing, the better it does



Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning

Bowen Jin[1], **Hansi Zeng**[2], **Zhenrui Yue**[1], **Jinsung Yoon**[3], **Sercan Ö. Arık**[3], **Dong Wang**[1], **Hamed Zamani**[2], **Jiawei Han**[1]
[1] Department of Computer Science, University of Illinois at Urbana-Champaign
[2] Center for Intelligent Information Retrieval, University of Massachusetts Amherst
[3] Google Cloud AI Research
{bowenj4,zhenrui3,dwang24,hanj}@illinois.edu, {hzeng, zamani}@cs.umass.edu

Training template

Answer the given question. You must conduct reasoning inside <think> and </think> first every time you get new information. After reasoning, if you find you lack some knowledge, you can call a search engine by <search> query </search>, and it will return the top searched results between <information> and </information>. You can search as many times as you want. If you find no further external knowledge needed, you can directly provide the answer inside <answer> and </answer> without detailed illustrations. For example, <answer> xxx </answer>. Question: question.

Search-R1's algorithm

Algorithm 1 LLM Response Rollout with Multi-Turn Search Engine Calls

Require: Input query x, policy model π_θ, search engine $\mathcal{R}$, maximum action budget B.
Ensure: Final response y.

Initialize rollout sequence $y \leftarrow \emptyset$
Initialize action count $b \leftarrow 0$
while $b < B$ **do**
  Initialize current action LLM rollout sequence $y_b \leftarrow \emptyset$
  while True **do**
    Generate response token $y_t \sim \pi_\theta(\cdot \mid x, y + y_b)$
    Append y_t to rollout sequence $y_b \leftarrow y_b + y_t$
    if y_t in [</search>, </answer>, <eos>] **then** break
    end if
  end while
  $y \leftarrow y + y_b$
  if <search> </search> detected in y_b **then**
    Extract search query $q \leftarrow$ Parse(y_b, <search>, </search>)
    Retrieve search results $d = \mathcal{R}(q)$
    Insert d into rollout $y \leftarrow y +$ <information>d</information>
  else if <answer> </answer> detected in y_b **then**
    return final generated response y
  else
    Ask for rethink $y \leftarrow y+$ "My action is not correct. Let me rethink."
  end if
  Increment action count $b \leftarrow b + 1$
end while
return final generated response y

Search-R1's performance

Methods	General QA			Multi-Hop QA				
	NQ†	TriviaQA*	PopQA*	HotpotQA†	2wiki*	Musique*	Bamboogle*	Avg.
Qwen2.5-7b-Base/Instruct								
Direct Inference	0.134	0.408	0.140	0.183	0.250	0.031	0.120	0.181
CoT	0.048	0.185	0.054	0.092	0.111	0.022	0.232	0.106
IRCoT	0.224	0.478	0.301	0.133	0.149	0.072	0.224	0.239
Search-o1	0.151	0.443	0.131	0.187	0.176	0.058	0.296	0.206
RAG	0.349	0.585	0.392	0.299	0.235	0.058	0.208	0.304
SFT	0.318	0.354	0.121	0.217	0.259	0.066	0.112	0.207
R1-base	0.297	0.539	0.202	0.242	0.273	0.083	0.296	0.276
R1-instruct	0.270	0.537	0.199	0.237	0.292	0.072	0.293	0.271
Search-R1-base	**0.480**	**0.638**	**0.457**	**0.433**	0.382	**0.196**	**0.432**	**0.431**
Search-R1-instruct	0.393	0.610	0.397	0.370	**0.414**	0.146	0.368	0.385
Qwen2.5-3b-Base/Instruct								
Direct Inference	0.106	0.288	0.108	0.149	0.244	0.020	0.024	0.134
CoT	0.023	0.032	0.005	0.021	0.021	0.002	0.000	0.015
IRCoT	0.111	0.312	0.200	0.164	0.171	0.067	0.240	0.181
Search-o1	0.238	0.472	0.262	0.221	0.218	0.054	**0.320**	0.255
RAG	0.348	0.544	0.387	0.255	0.226	0.047	0.080	0.270
SFT	0.249	0.292	0.104	0.186	0.248	0.044	0.112	0.176
R1-base	0.226	0.455	0.173	0.201	0.268	0.055	0.224	0.229
R1-instruct	0.210	0.449	0.171	0.208	0.275	0.060	0.192	0.224
Search-R1-base	**0.406**	**0.587**	**0.435**	0.284	0.273	0.049	0.088	0.303
Search-R1-instruct	0.341	0.545	0.378	**0.324**	**0.319**	**0.103**	0.264	**0.325**

RAG vs. long-context LLMs

RETRIEVAL MEETS LONG CONTEXT LARGE LANGUAGE MODELS

Peng Xu[†], Wei Ping[†], Xianchao Wu, Lawrence McAfee
Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina
Mohammad Shoeybi, Bryan Catanzaro
NVIDIA
[†]{pengx, wping}@nvidia.com

Abstract

Extending the context window of large language models (LLMs) is getting popular recently, while the solution of augmenting LLMs with retrieval has existed for years. The natural questions are: *i) Retrieval-augmentation versus long context window, which one is better for downstream tasks? ii) Can both methods be combined to get the best of both worlds?* In this work, we answer these questions by studying both solutions using two state-of-the-art pretrained LLMs, i.e., a proprietary 43B GPT and Llama2-70B. Perhaps surprisingly, we find that LLM with 4K context window using simple retrieval-augmentation at generation can achieve comparable performance to finetuned LLM with 16K context window via *positional interpolation* on long context tasks, while taking much less computation. More importantly, we demonstrate that retrieval can significantly improve the performance of LLMs regardless of their extended context window sizes. Our best model, retrieval-augmented Llama2-70B with 32K context window, outperforms GPT-3.5-turbo-16k and Davinci003 in terms of average score on nine long context tasks including question answering, query-based summarization, and in-context few-shot learning tasks. It also outperforms its non-retrieval Llama2-70B-32k baseline by a margin, while being much faster at generation. Our study provides general insights on the choice of retrieval-augmentation versus long context extension of LLM for practitioners.

Can Long-Context Language Models Subsume Retrieval, RAG, SQL, and More?

Jinhyuk Lee* Anthony Chen* Zhuyun Dai*
Dheeru Dua Devendra Singh Sachan Michael Boratko Yi Luan
Sébastien M. R. Arnold Vincent Perot Siddharth Dalmia Hexiang Hu
Xudong Lin Panupong Pasupat Aida Amini Jeremy R. Cole
Sebastian Riedel Iftekhar Naim Ming-Wei Chang Kelvin Guu

Google DeepMind

Abstract

Long-context language models (LCLMs) have the potential to revolutionize our approach to tasks traditionally reliant on external tools like retrieval systems or databases. Leveraging LCLMs' ability to natively ingest and process entire corpora of information offers numerous advantages. It enhances user-friendliness by eliminating the need for specialized knowledge of tools, provides robust end-to-end modeling that minimizes cascading errors in complex pipelines, and allows for the application of sophisticated prompting techniques across the entire system. To assess this paradigm shift, we introduce LOFT, a benchmark of real-world tasks requiring context up to millions of tokens designed to evaluate LCLMs' performance on in-context retrieval and reasoning. Our findings reveal LCLMs' surprising ability to rival state-of-the-art retrieval and RAG systems, despite never having been explicitly trained for these tasks. However, LCLMs still face challenges in areas like compositional reasoning that are required in SQL-like tasks. Notably, prompting strategies significantly influence performance, emphasizing the need for continued research as context lengths grow. Overall, LOFT provides a rigorous testing ground for LCLMs, showcasing their potential to supplant existing paradigms and tackle novel tasks as model capabilities scale.[1]

Retrieval Augmented Generation or Long-Context LLMs? A Comprehensive Study and Hybrid Approach

Zhuowan Li[1] **Cheng Li**[1] **Mingyang Zhang**[1]
Qiaozhu Mei[2*] **Michael Bendersky**[1]
[1] Google DeepMind [2] University of Michigan
[1] {zhuowan,chgli,mingyang,bemike}@google.com [2] qmei@umich.edu

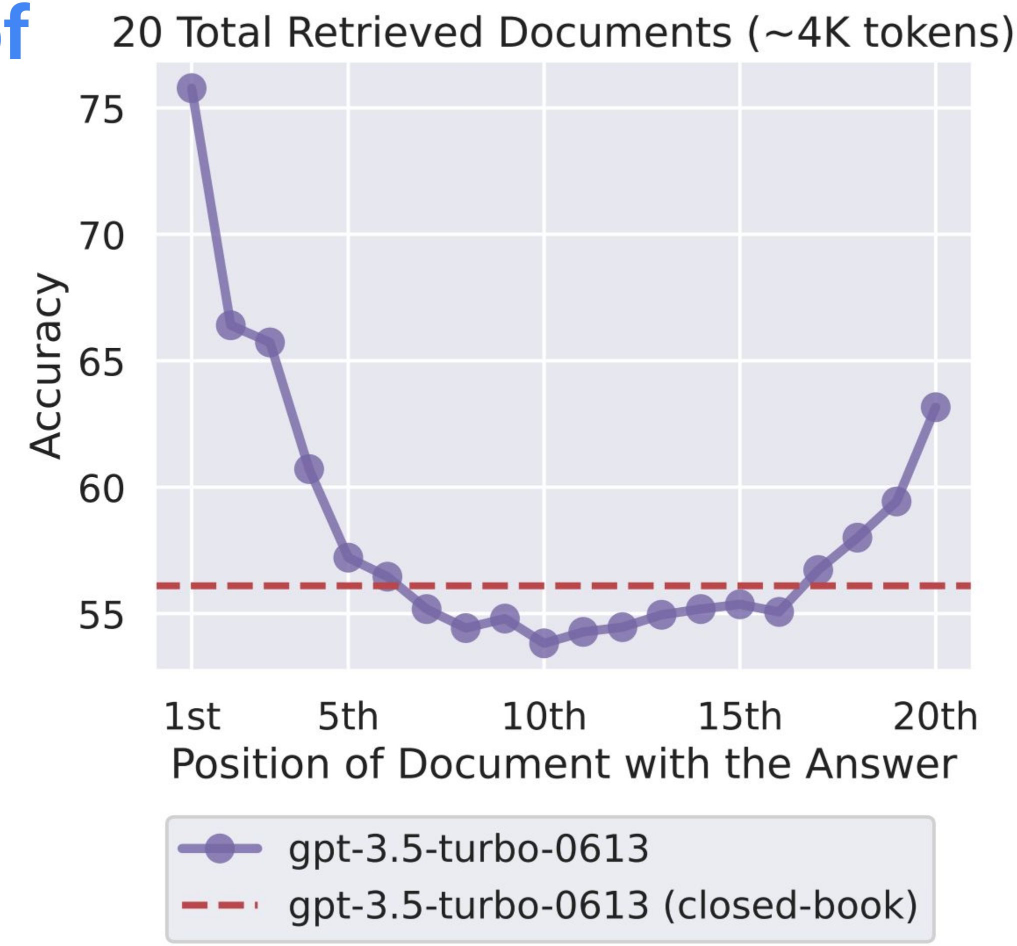
Abstract

Retrieval Augmented Generation (RAG) has been a powerful tool for *Large Language Models (LLMs)* to efficiently process overly lengthy contexts. However, recent LLMs like Gemini-1.5 and GPT-4 show exceptional capabilities to understand long contexts directly. We conduct a comprehensive comparison between RAG and long-context *(LC)* LLMs, aiming to leverage the strengths of both. We benchmark RAG and LC across various public datasets using three latest LLMs. Results reveal that when resourced sufficiently, LC consistently outperforms RAG in terms of average performance. However, RAG's significantly lower cost remains a distinct advantage. Based on this observation, we propose SELF-ROUTE, a simple yet effective method that routes queries to RAG or LC based on model self-reflection. SELF-ROUTE significantly reduces the computation cost while maintaining a comparable performance to LC. Our findings provide a guideline for long-context applications of LLMs using RAG and LC.

Lost in the Middle: How Language Models Use Long Contexts

Nelson F. Liu[1*] **Kevin Lin**[2] **John Hewitt**[1] **Ashwin Paranjape**[3]
Michele Bevilacqua[3] **Fabio Petroni**[3] **Percy Liang**[1]
[1]Stanford University [2]University of California, Berkeley [3]Samaya AI
nfliu@cs.stanford.edu

Changing the location of relevant information results in a U-shaped performance curve



Context rot: How increasing input tokens impacts model performance

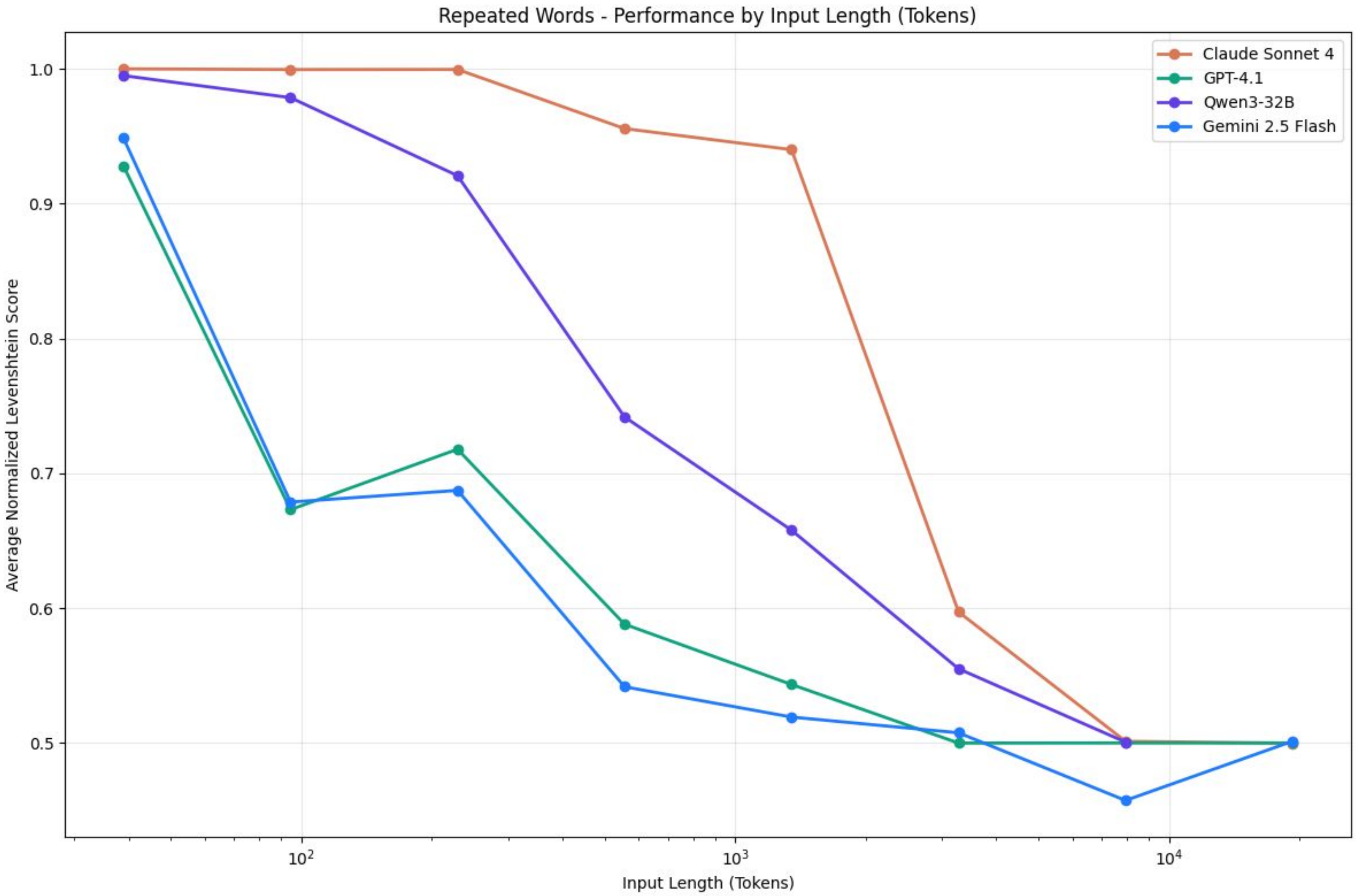
- https://research.trychroma.com/context-rot

We design a controlled task in which the model must replicate a sequence of repeated words, with a single unique word inserted at a specific position. The prompt explicitly instructs the model to reproduce the input text exactly.

One example prompt is:

```
Simply replicate the following text, output the exact same text: apple apple
apple apple apples apple apple apple apple apple apple apple apple apple apple
apple apple apple apple apple apple apple apple apple apple
```

Repeated Words - Sample Prompt Containing 'apple' as the repeated word, and 'apples' as the unique word


Repeated Words - Performance by Input Length (Tokens)
Claude Sonnet 4
GPT-4.1
Qwen3-32B
Gemini 2.5 Flash
Average Normalized Levenshtein Score
1.0
0.9
0.8
0.7
0.6
0.5
10^2
10^3
10^4
Input Length (Tokens)

Model context protocol (MCP)

- https://www.deeplearning.ai/short-courses/mcp-build-rich-context-ai-apps-with-anthropic/

Demo

- https://learn.deeplearning.ai/courses/mcp-build-rich-context-ai-apps-with-anthropic/lesson/ccsd0/why-mcp

What is the Model Context Protocol (MCP)

MCP is an open protocol that standardizes how your **LLM applications** connect to and work with your **tools & data sources**.

REST APIs

Standardize how **web applications** interact with the **backend**

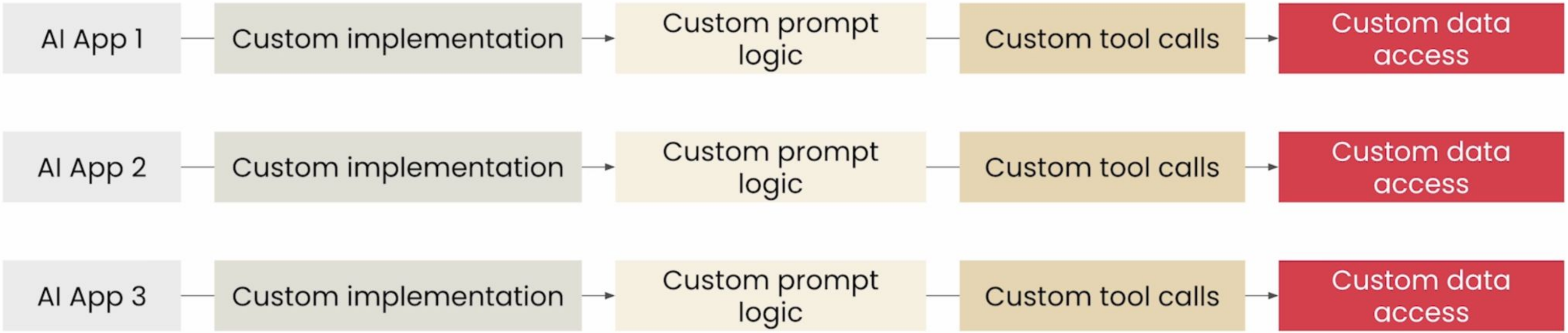
LSP

Standardizes how **IDEs** interact with **language-specific tools**

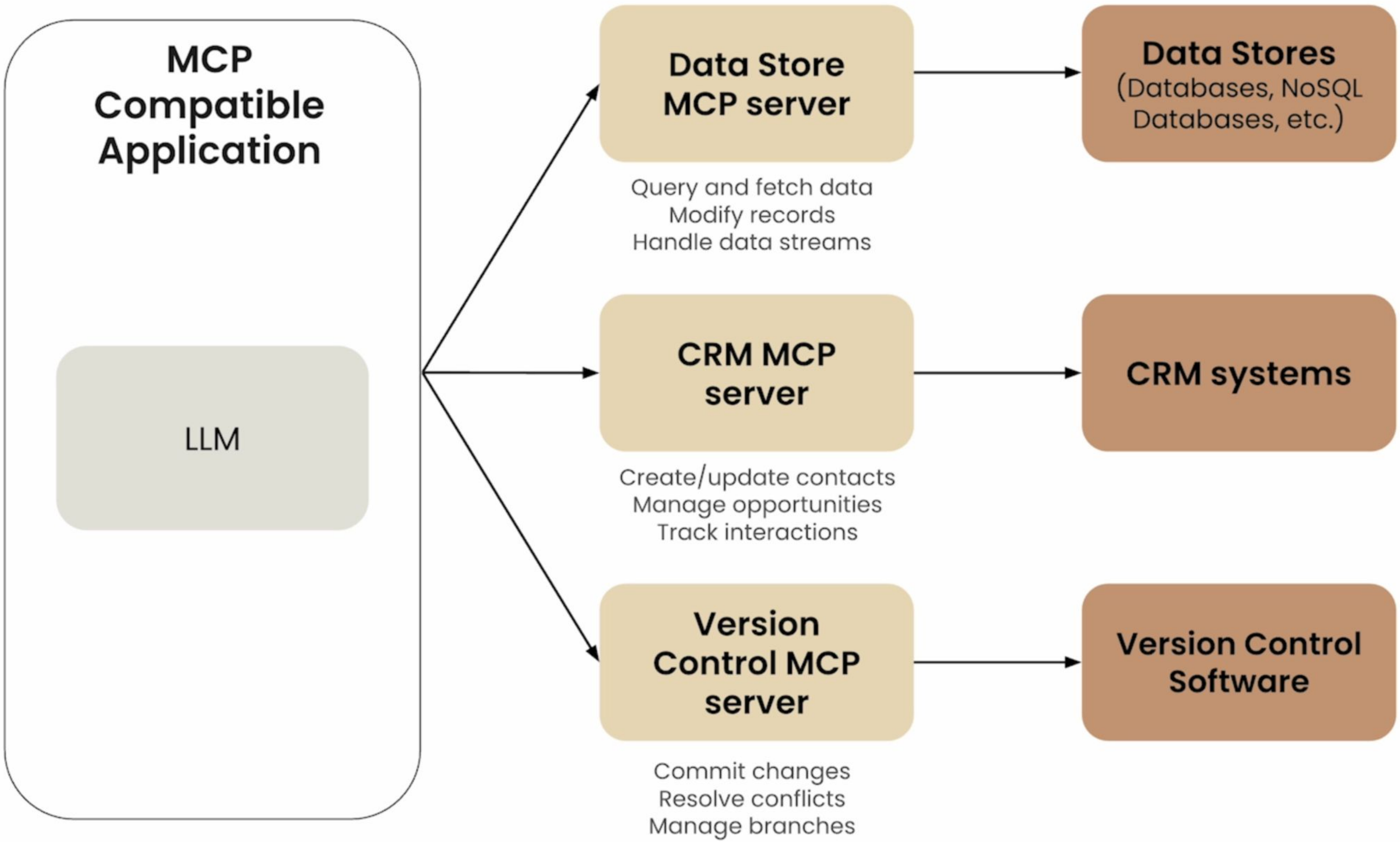
MCP

Standardizes how **AI applications** interact with **external systems**

Without MCP: Fragmented AI Development



With MCP: Standardized AI Development

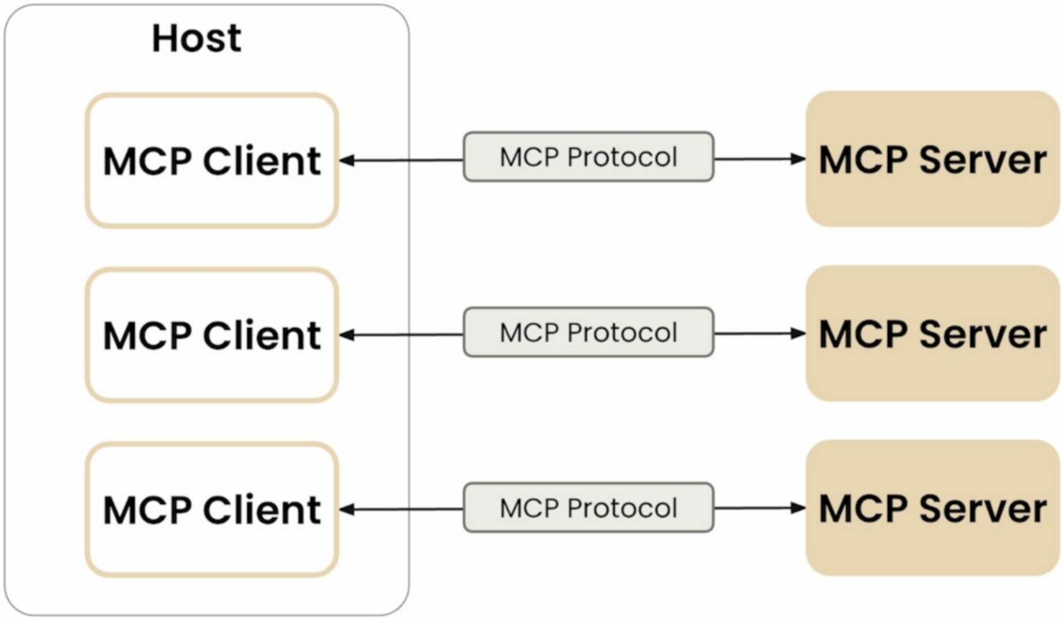


Client-Server Architecture

Host are LLM applications that want to access data through MCP (ex: Claude Desktop, IDEs, AI agents).

MCP Servers are lightweight programs that each expose specific capabilities through MCP.

MCP Clients maintain 1:1 connections with servers, inside the host application.



README | Code of conduct | MIT license | Security

Model Context Protocol servers

This repository is a collection of *reference implementations* for the Model Context Protocol (MCP), as well as references to community built servers and additional resources.

The servers in this repository showcase the versatility and extensibility of MCP, demonstrating how it can be used to give Large Language Models (LLMs) secure, controlled access to tools and data sources. Each MCP server is implemented with either the Typescript MCP SDK or Python MCP SDK.

> Note: Lists in this README are maintained in alphabetical order to minimize merge conflicts when adding new items.

🌟 Reference Servers

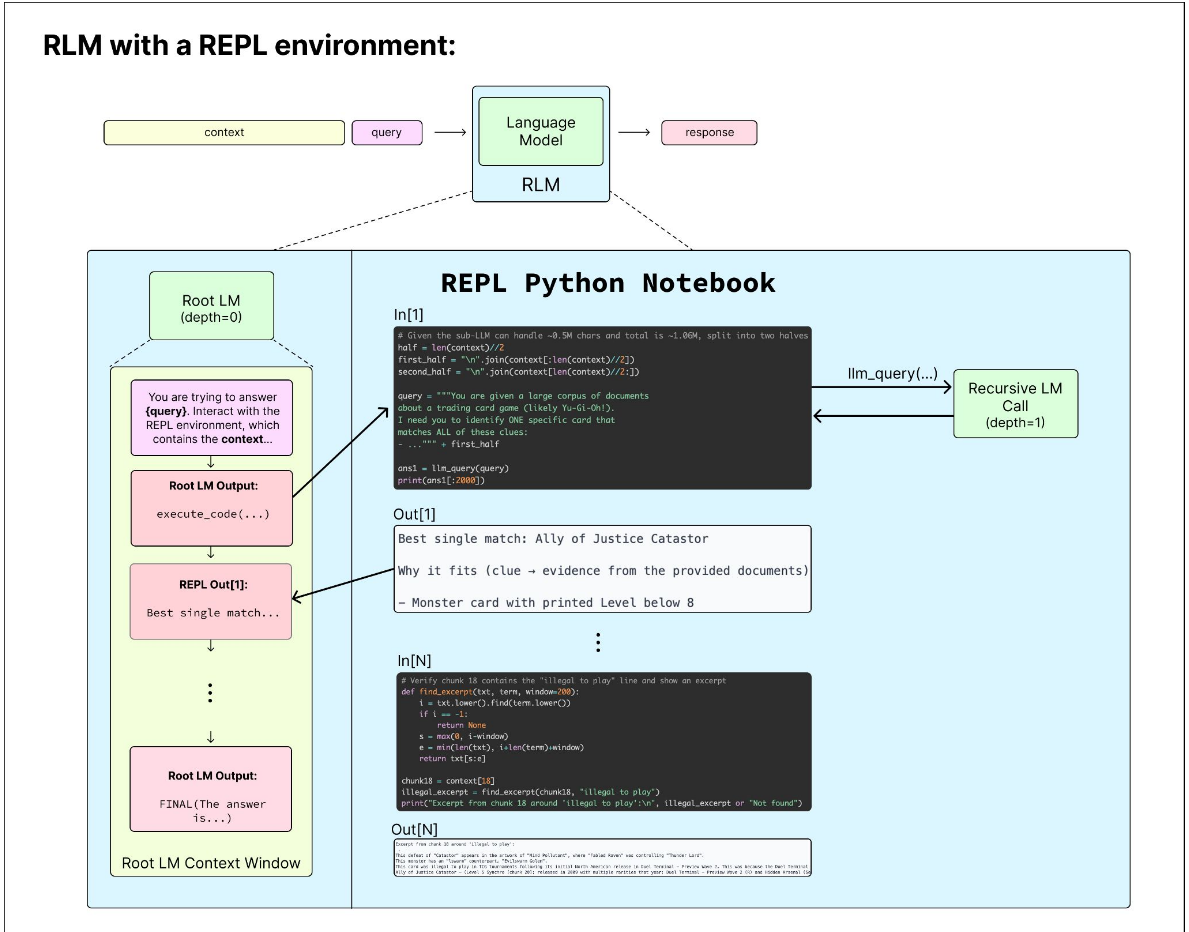
These servers aim to demonstrate MCP features and the TypeScript and Python SDKs.

- AWS KB Retrieval - Retrieval from AWS Knowledge Base using Bedrock Agent Runtime
- Brave Search - Web and local search using Brave's Search API
- EverArt - AI image generation using various models
- Everything - Reference / test server with prompts, resources, and tools
- Fetch - Web content fetching and conversion for efficient LLM usage
- Filesystem - Secure file operations with configurable access controls
- Git - Tools to read, search, and manipulate Git repositories
- GitHub - Repository management, file operations, and GitHub API integration
- GitLab - GitLab API, enabling project management
- Google Drive - File access and search capabilities for Google Drive
- Google Maps - Location services, directions, and place details
- Memory - Knowledge graph-based persistent memory system
- PostgreSQL - Read-only database access with schema inspection
- Puppeteer - Browser automation and web scraping
- Redis - Interact with Redis key-value stores
- Sentry - Retrieving and analyzing issues from Sentry.io
- Sequential Thinking - Dynamic and reflective problem-solving through thought sequences
- Slack - Channel management and messaging capabilities
- Sqlite - Database interaction and business intelligence capabilities
- Time - Time and timezone conversion capabilities

🤝 Third-Party Servers

🎖️ Official Integrations

Recursive Language Models



https://alexzhang13.github.io/blog/2025/rlm/

Recursive Language Models (cont'd)

Input: a long report of 10,000 words about a company's 5-year strategy
Query: What risks did the report identify in year 3?

AI: High-level scan / peek at the report's table of contents

```
# find the header line number
hdr=$(grep -inE '^(table of contents|contents|toc)\s*$' doc.txt | head -1 | cut -d: -f1)
# print the next 100 lines that look like TOC entries
tail -n +"$hdr" doc.txt | head -n 100 | grep -nE '(^\s*\d+(\.\d+)*\s+.+)|(^\s*.+\s+\.{2,}\s*\d+\s*$)'
```

Select sub-chunks
Based on that peek, it might decide: "Chapter 4 (pages 40–60) and Chapter 6 (pages 90–110) are likely to mention Year 3 risks."
Recursive call on chunks It then issues sub-queries:
"From pages 40–60, what risks are mentioned for year 3?"
"From pages 90–110, what risks are mentioned for year 3?"
Aggregate / refine
"In year 3 the report warned of supply chain disruption, regulatory changes, and declining demand in Asia."

Recursive Language Models (cont'd)

Input: a long report of 10,000 words about a company's 5-year strategy
Query: What risks did the report identify in year 3?

AI: High-level scan / peek at the report's table of contents

```
# find the header line number
hdr=$(grep -inE '^(table of contents|contents|toc)\s*$' doc.txt | head -1 | cut -d: -f1)
# print the next 100 lines that look like TOC entries
tail -n +"$hdr" doc.txt | head -n 100 | grep -nE '(^\s*\d+(\.\d+)*\s+.+)|(^\s*.+\s+\.{2,}\s*\d+\s*$)'
```

Select sub-chunks

Based on that peek, it might decide: "Chapter 4 (pages 40–60) and Chapter 6 (pages 90–110) are likely to mention Year 3 risks."

Recursive call on chunks It then issues sub-queries:

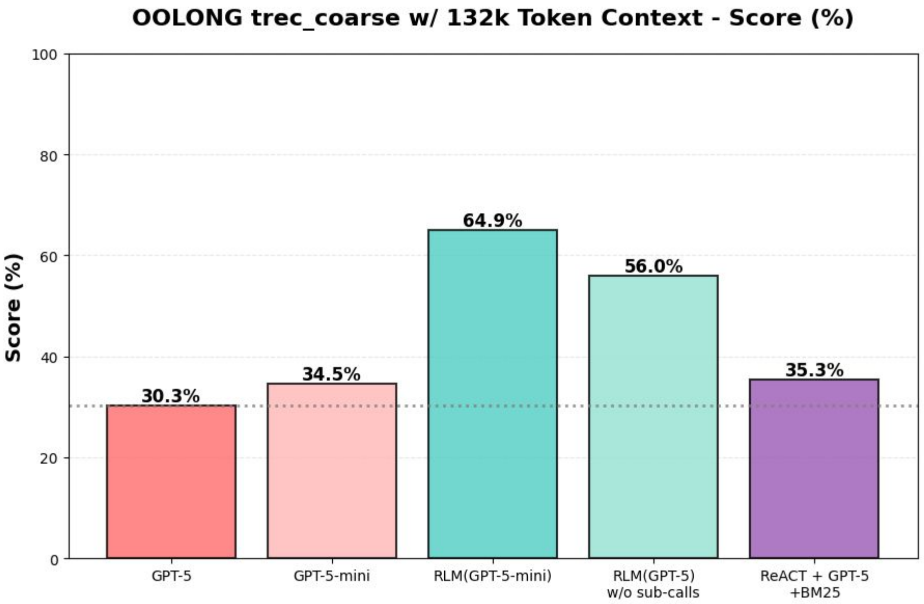
"From pages 40–60, what risks are mentioned for year 3?"

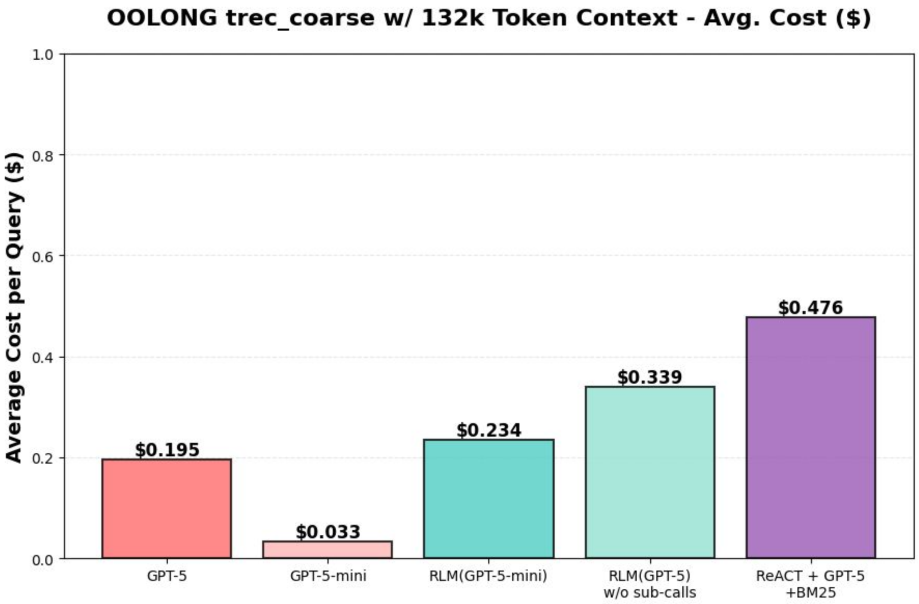
"From pages 90–110, what risks are mentioned for year 3?"

Aggregate / refine

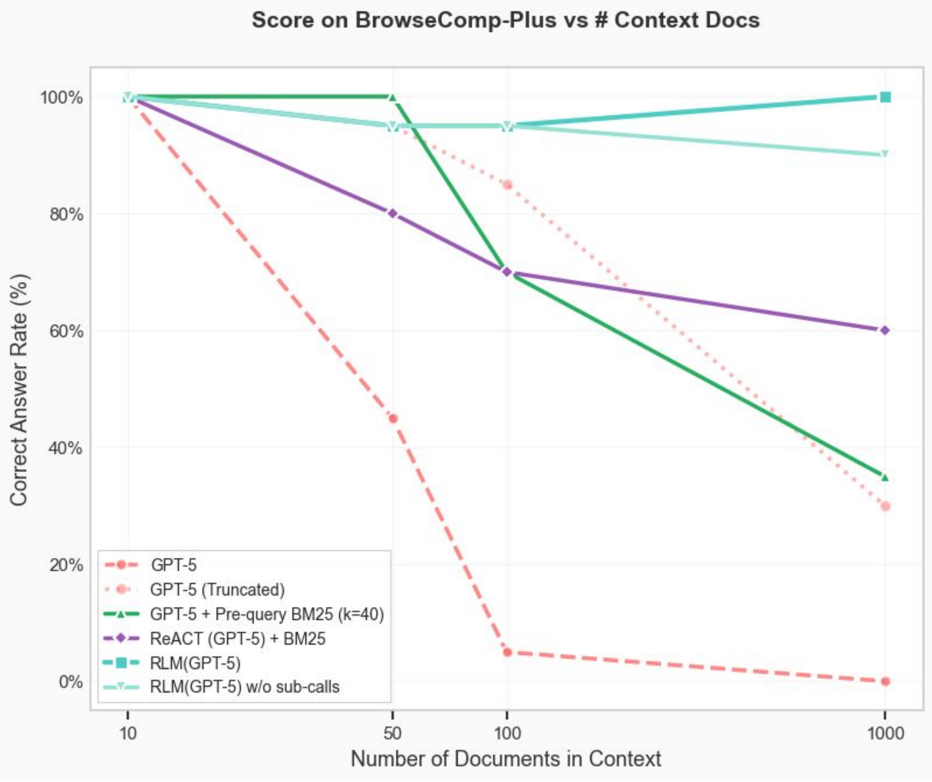
"In year 3 the report warned of supply chain disruption, regulatory changes, and declining demand in Asia."

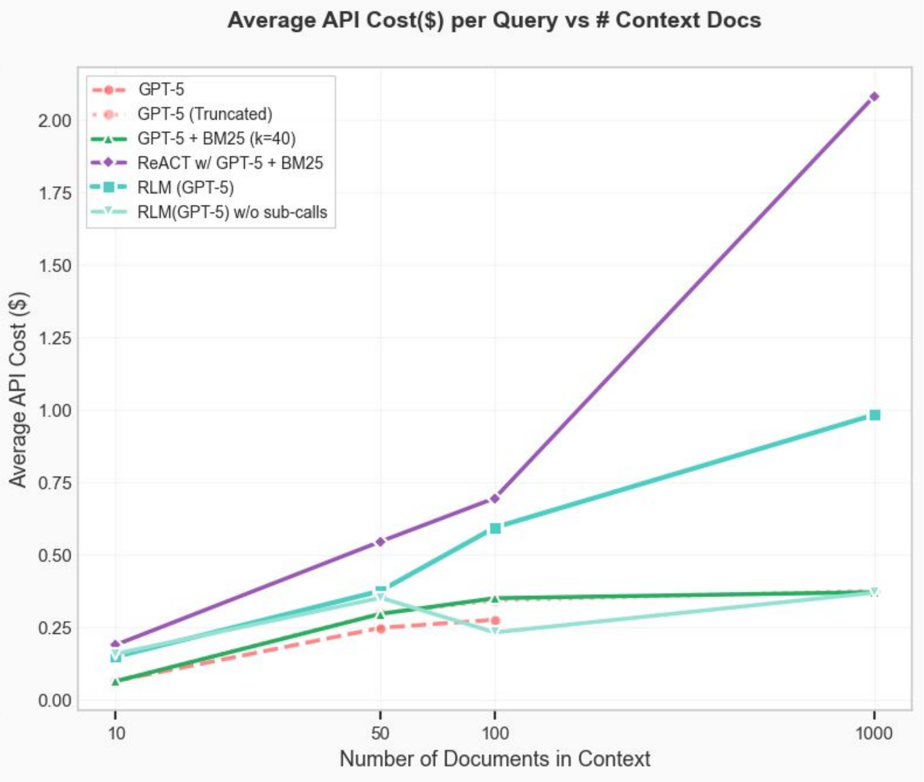
Recursive Language Models (cont'd)





Recursive Language Models (cont'd)





Thank you!