Large Language Models

Runpeng Dai

The University of North Carolina at Chapel Hill

Content

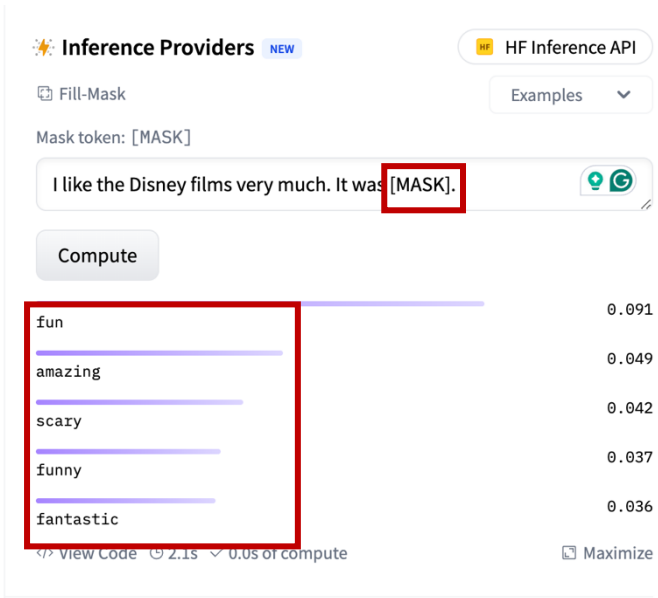
1. Introduction to Large Language Models

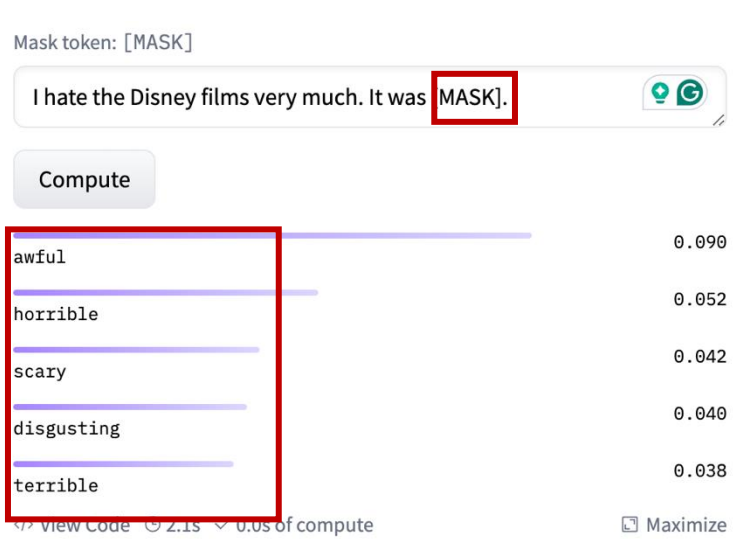
2. Popular Large Language Model Architectures

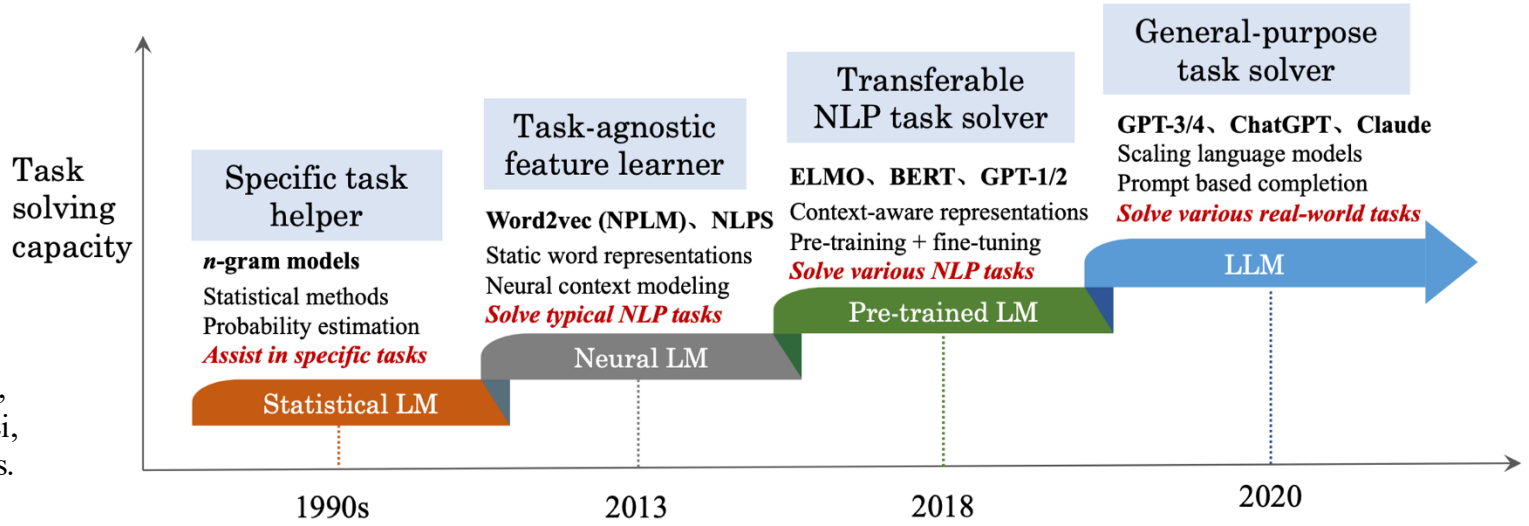
3. Make LLMs More Suitable for Your Downstream Application

Language Models >>Large LMs (LLMs)

- **Definition:** A language model aims to predict the probability of the occurrence of a token or a sequence of tokens.
- The probability prediction of a language model is closely related to **context** and **corpus** information.
- Language model is **not a new technical** concept specially for LLMs, but has evolved with the advance of artificial intelligence over the decades.
- **Definition:** LLMs have billions of parameters, trained on massive corpora.





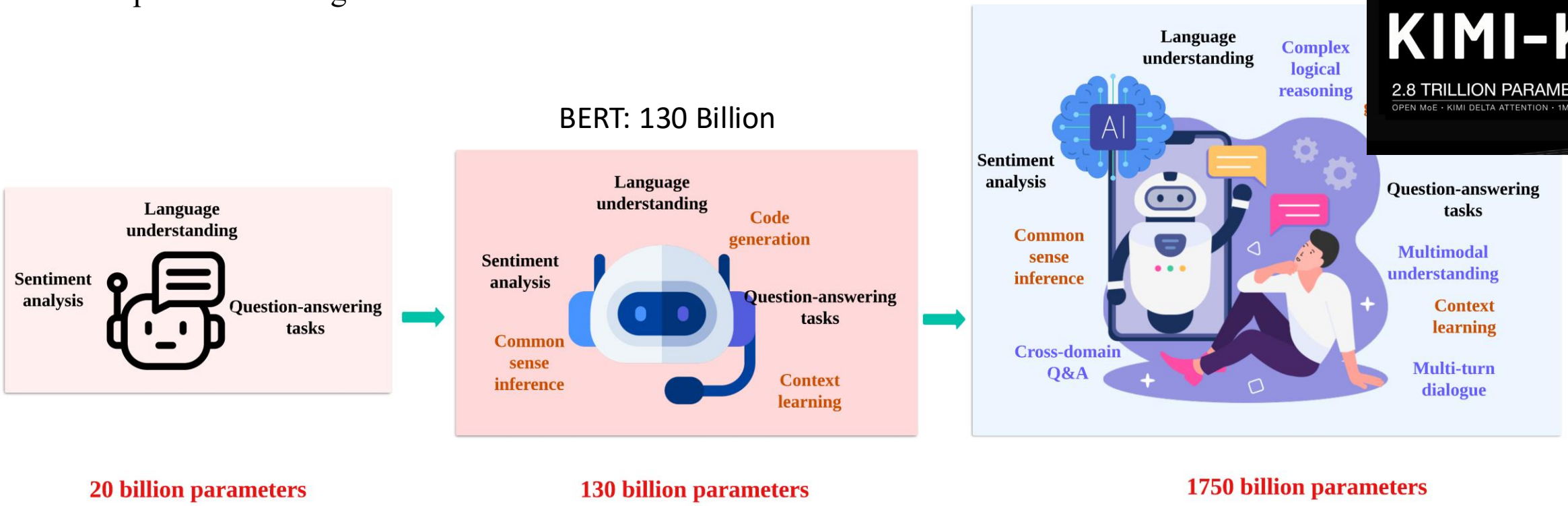


Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., . . . Wen, J. (2023). A Survey of Large Language Models. *ArXiv*. https://arxiv.org/abs/2303.18223

Evolution brings new abilities

Key Point: With the continuous iteration and updates of LLMs, the range of problems they can solve has become increasingly rich, demonstrating some new abilities.

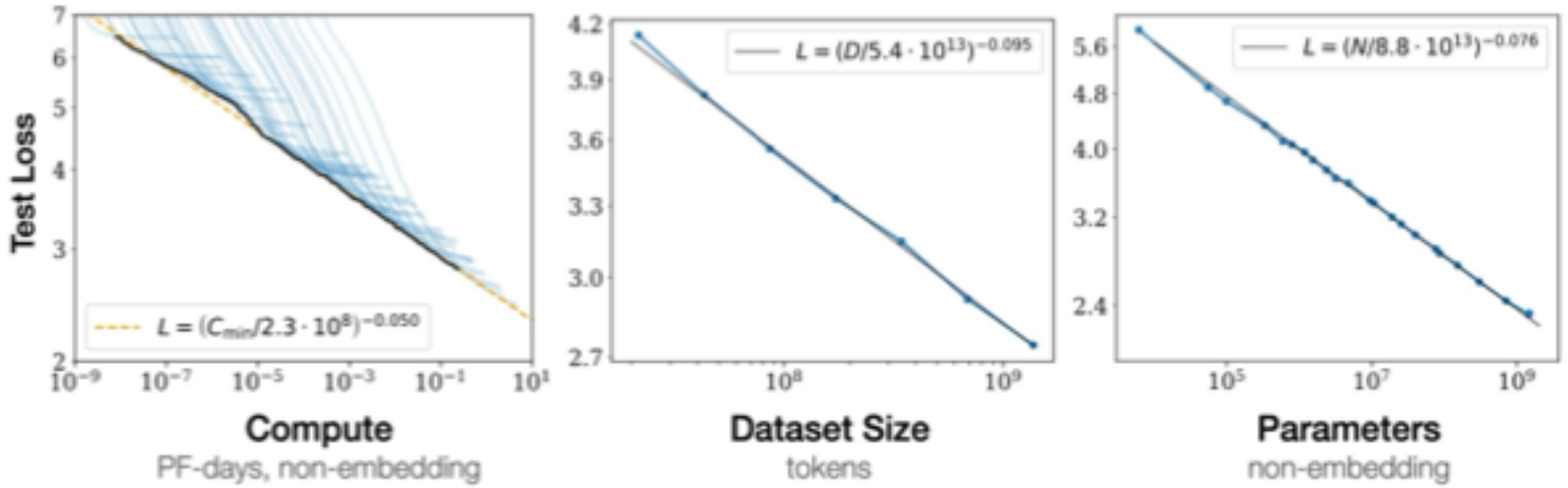
- **Growing Complexity:** Each new LLM version shows improvements in reasoning, creativity, and context handling.
- **Emergent Properties:** Larger, more diverse training corpora lead to surprising capabilities (e.g., zero-shot translation, chain-of-thought prompting).
- **Broadening Applications:** Beyond text generation, LLMs now assist in code completion, legal drafting, and even scientific problem-solving.



Scaling Law

OpenAI **Kaplan et al. (2020)**

- Three quantities dominate performance: N = # parameters, D = # tokens, C = #FLOPS
- Model shape doesn't matter very much.
- Performance improves as long as we increase both N and D.
- Training loss curves follow predictable power laws.



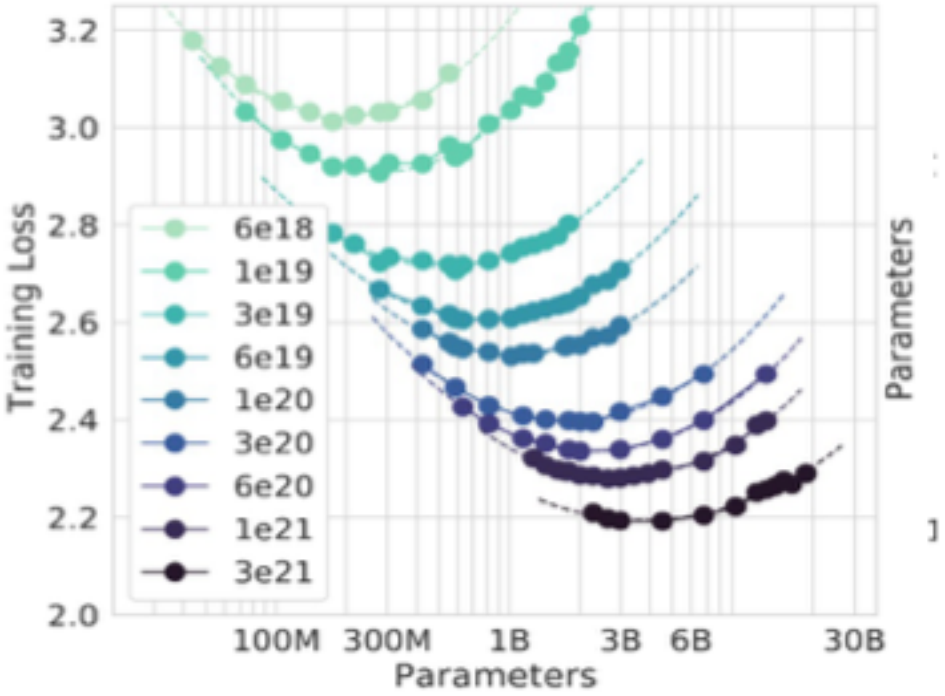
$$L(N, D) = \left[\left(\frac{N_c}{N}\right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_c}{D}\right]^{\alpha_D}$$

Every time we increase the model size 8x, we only need to increase the data by roughly 5x to avoid a penalty.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). Scaling Laws for Neural Language Models. *ArXiv*. https://arxiv.org/abs/2001.08361

Scaling Law

DeepMind Hoffman et al. (2022)



- The big shift for Chinchilla was a dramatic increase in the number of tokens.
- Everyone had been using way too little data.
- Increasing the amount of data and decreasing the model size.
- → Same computational budget but get much better performance!

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., Driessche, G. V., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., . . . Sifre, L. (2022). Training Compute-Optimal Large Language Models. *ArXiv*. https://arxiv.org/abs/2203.15556

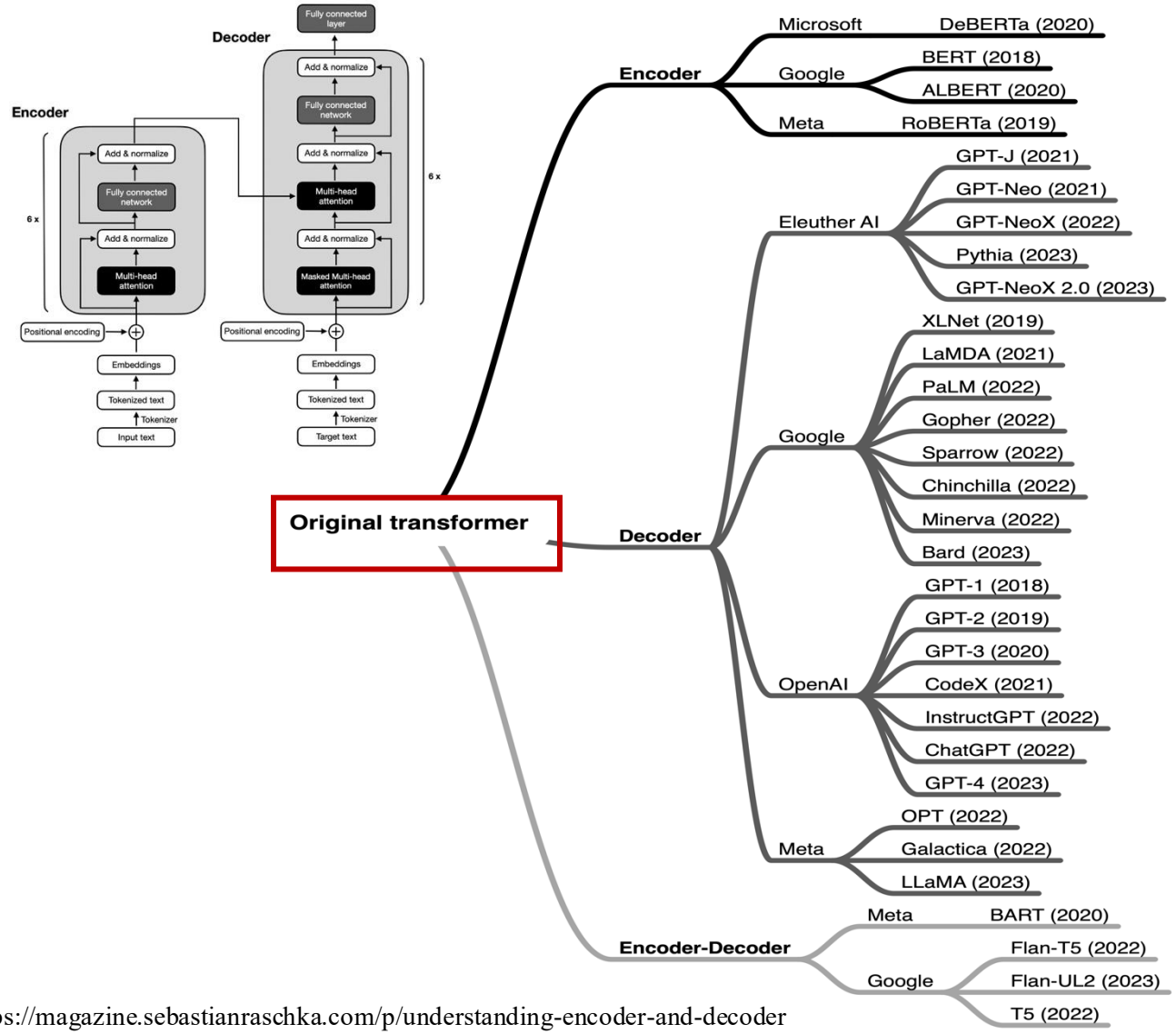
Content

1. Introduction to Large Language Models

2. Popular Large Language Model Architectures

3. Make LLMs More Suitable for Your Downstream Application

The core of LLMs --- Transformer



Transformer models have diversified into **Encoder, Decoder, and Encoder-Decoder branches**, driven by advancements from companies like Microsoft, Google, Meta, OpenAI, and Eleuther AI.

- ❖ Encoder models like BERT focus on language understanding.
- ❖ Decoder models like GPT are aimed at generation.
- ❖ Encoder-decoder models.
- ❖ Diffusion LLMs.

https://magazine.sebastianraschka.com/p/understanding-encoder-and-decoder

Encoder-only models —— BERT

What is BERT?

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained language model with an encoder-only architecture.

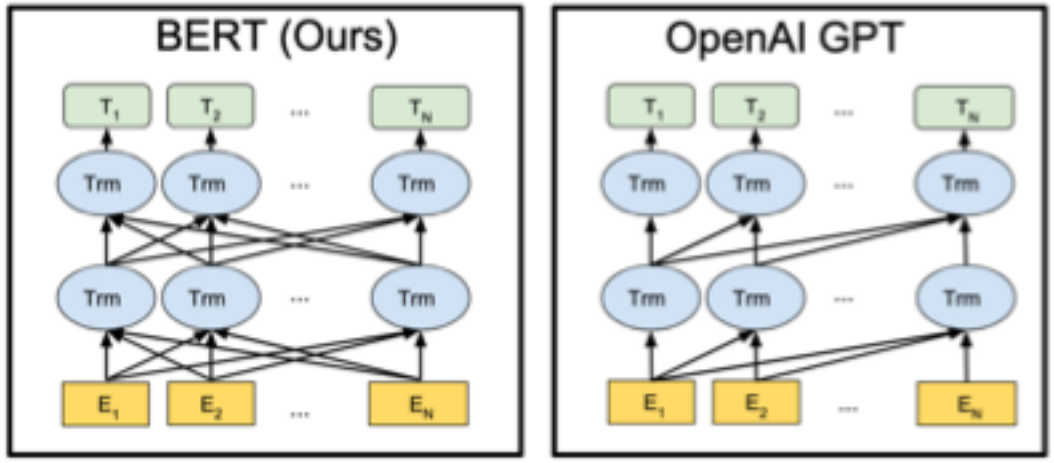
- **Bidirectional Encoding**: Uses bidirectional attention to understand **context from both directions**.
- Suitable for **understanding-based tasks** like classification and question answering.

E.g. The cat ran over the street, because it got startled

Difference with Decoder only (GPT)

Uses unidirectional attention to predict the **next word** in a sequence, only have context from previous words.(Causal language model)

Suitable for **generation-based tasks** like text completion and conversation



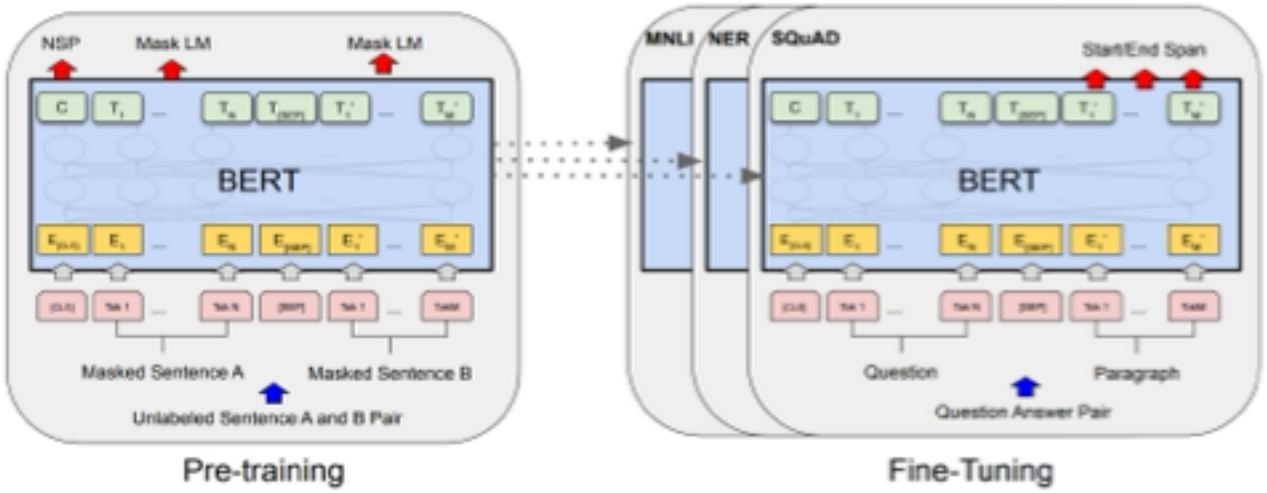
BERT uses a **bidirectional Transformer**.
OpenAI GPT uses a **left-to-right Transformer**.

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv*. https://arxiv.org/abs/1810.04805

Pretraining Finetuning paradigm

Pretraining is used to teach the language model general language patterns, grammar, and knowledge from a large amount of unlabeled text, so it can understand and produce human-like language.

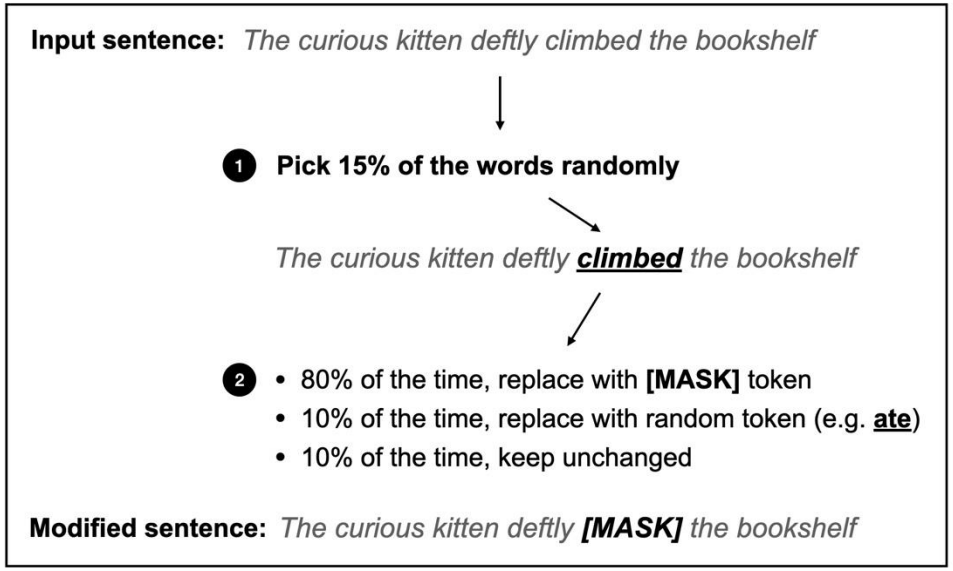
Finetuning (post) adapts the pretrained model to a specific task or domain with labeled dataset to specialize at down stream tasks or to align well with Human preference.



BERT Training Methods —— Pre-training

➢ Task1: Masked Language Model (MLM)

- **Randomly masks the input words** and predicts the original words.
- 80% of masked tokens are replaced with [MASK], 10% are replaced with a random word, and 10% remain unchanged to reduce pre-training and fine-tuning mismatch.



➢ Task2: Next Sentence Prediction (NSP)

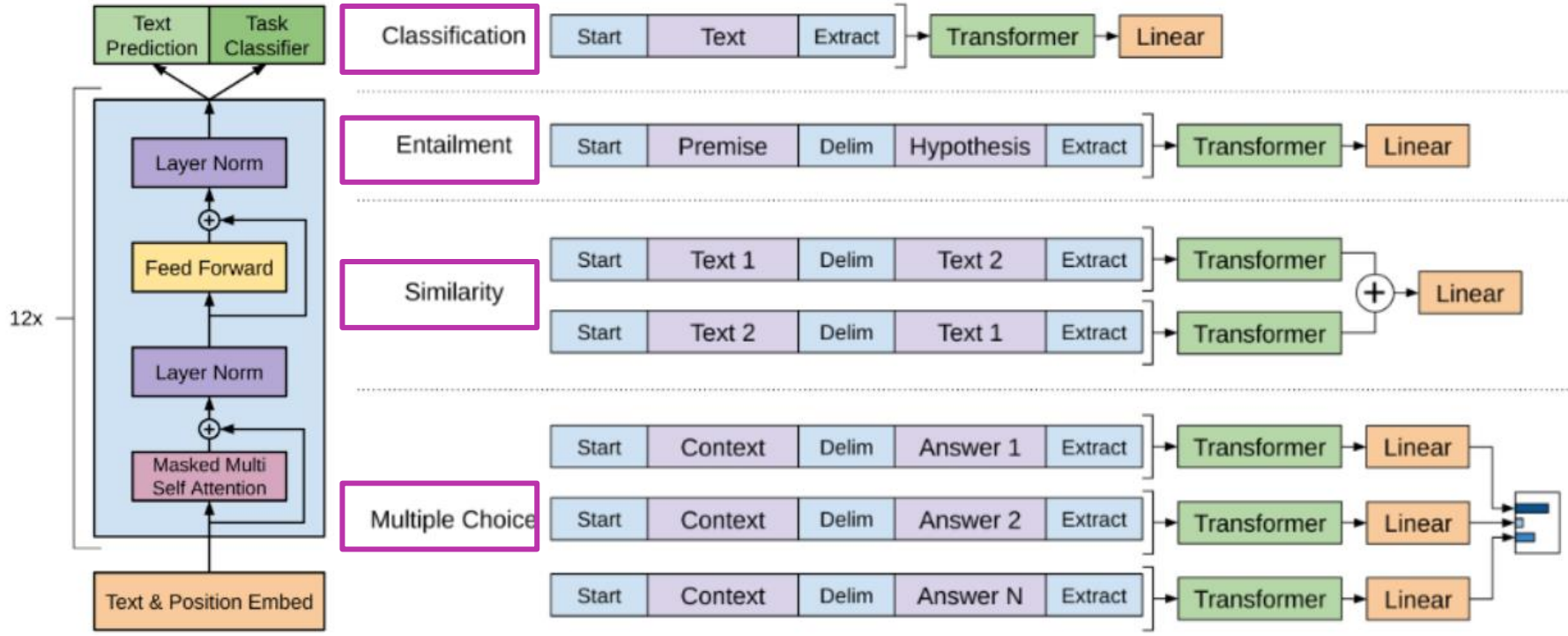
Predicts whether two sentences are consecutive: 50% of examples are actual consecutive sentences (**IsNext**), 50% of examples are randomly chosen, non-consecutive sentences (**NotNext**).

Helps the model learn textual coherence and is particularly useful for tasks like **Question Answering (QA)** and **Natural Language Inference (NLI)**.

https://magazine.sebastianraschka.com/p/understanding-encoder-and-decoder

BERT Training Methods —— Fine-tuning

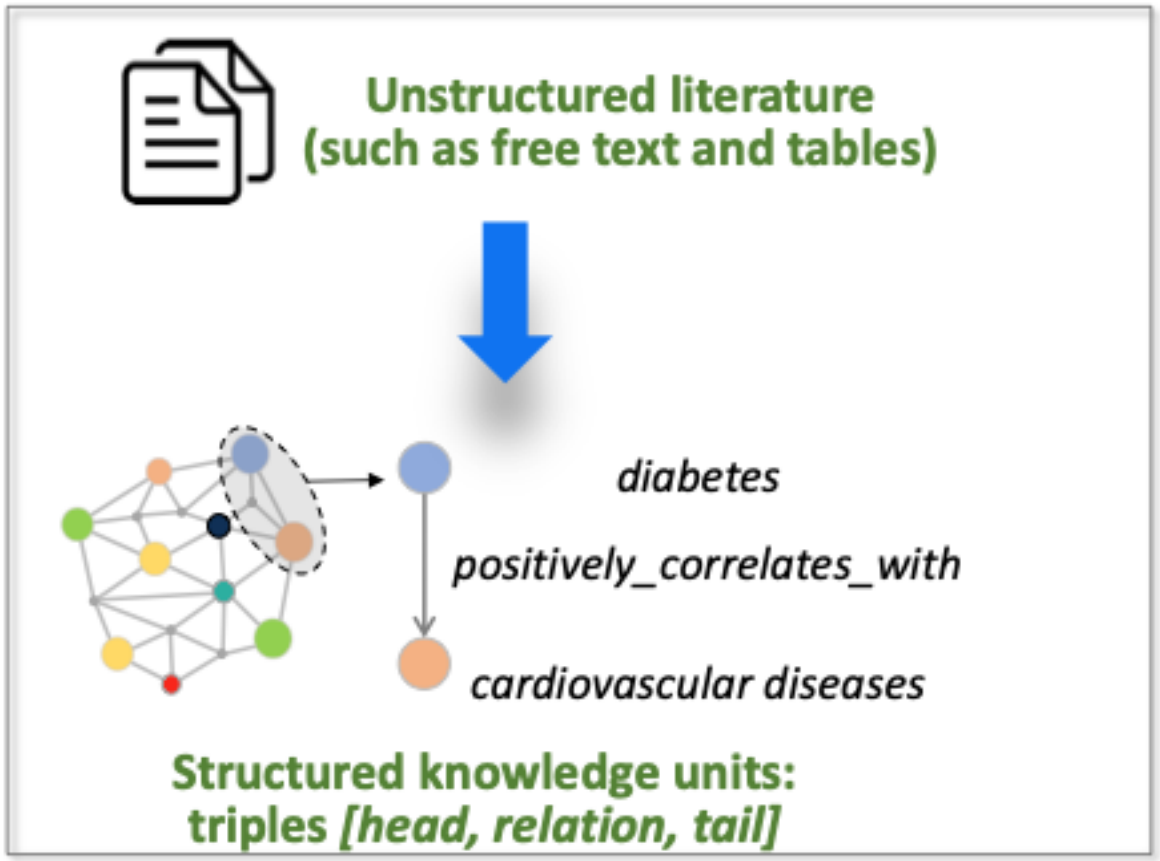
Same architecture, different tasks: A simple output layer is added for tasks such as classification, question answering (QA), and Named Entity Recognition (NER).



(i) Transformer architecture and training objectives used in this work. (ii) Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

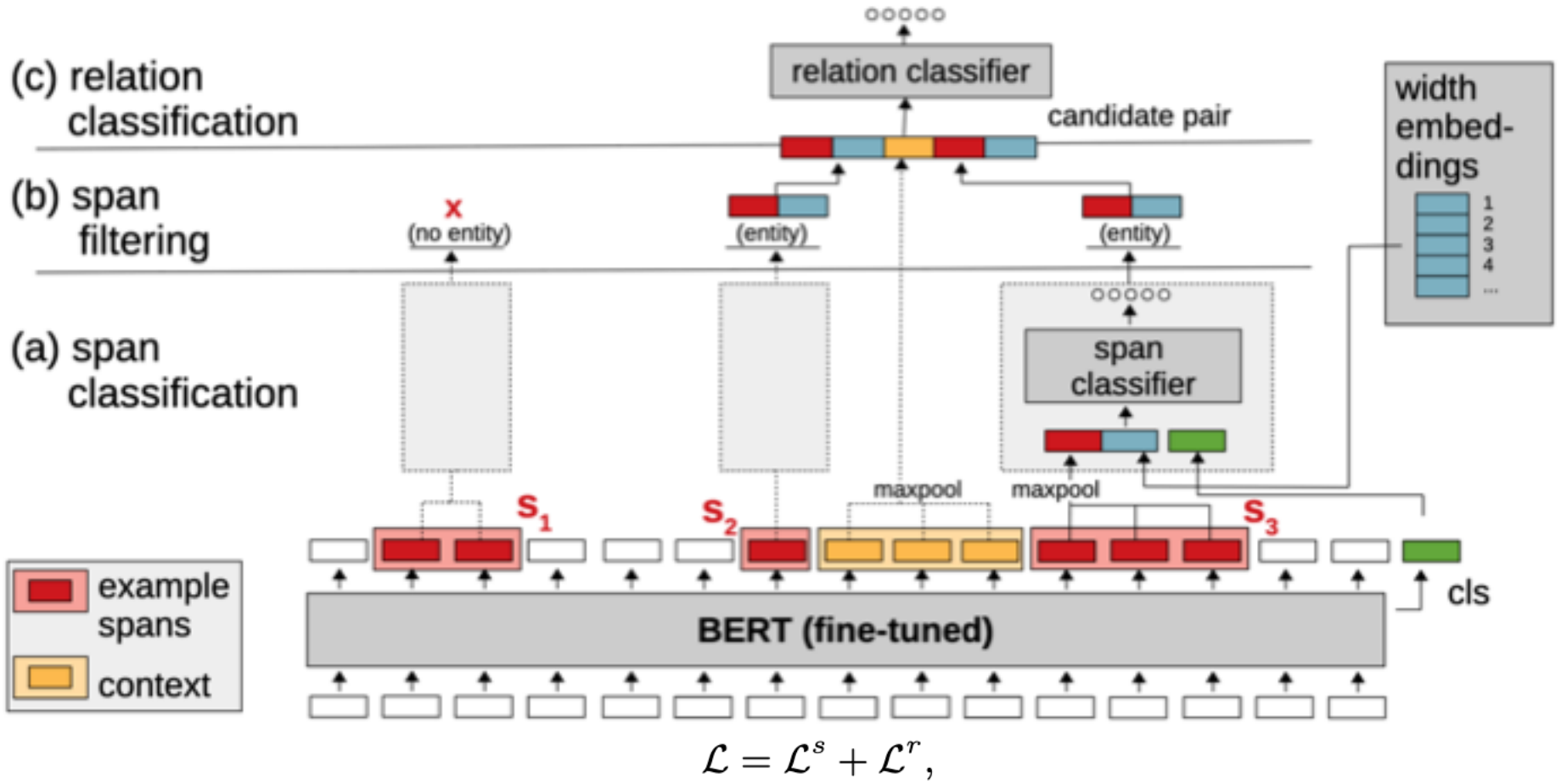
Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training [J]. 2018.

Application: Knowledge graph



- A **biomedical knowledge graph (KG)** is a structured network that represents biomedical entities (like genes, proteins, drugs, diseases, symptoms, etc.) as **nodes**, and their relationships (such as "treats," "causes," "interacts with," etc.) as **edges**.
- The goal is to **organize complex biomedical knowledge** (structured and unstructured) into a machine-readable graph structure, making it easier to query, visualize, and extract insights.
- Extracts subject-predicate-object triplets(RE) from PubMed abstracts

Knowledge graph construction – a Bert method



$$\mathcal{L} = \mathcal{L}^s + \mathcal{L}^r,$$

whereas $\mathcal{L}^s$ denotes the span classifier's loss (cross-entropy over the entity classes including *none*) and $\mathcal{L}^r$ denotes the binary cross-entropy over relation classes. Both losses are averaged over each

Decoder-only models GPT family

GPT: Uses unidirectional attention to predict the **next word** in a sequence. Suitable for **generation-based tasks** like text completion and conversation. GPT is autoregressive(causal) language model defines a conditional distribution:

$$p(x_i \mid x_{1:i-1})$$

GPT pretraining: Let θ be all the parameters of large language models. Let D be the training data consisting of a set of sequences. We can then follow the maximum likelihood principle and define the following negative log-likelihood objective function:

$$\mathcal{L}(\theta) = \sum_{x_{1:L} \in \mathcal{D}} -\log p_\theta(x_{1:L}) = \sum_{x_{1:L} \in \mathcal{D}} \sum_{i=1}^{L} -\log p_\theta(x_i \mid x_{1:i-1})$$

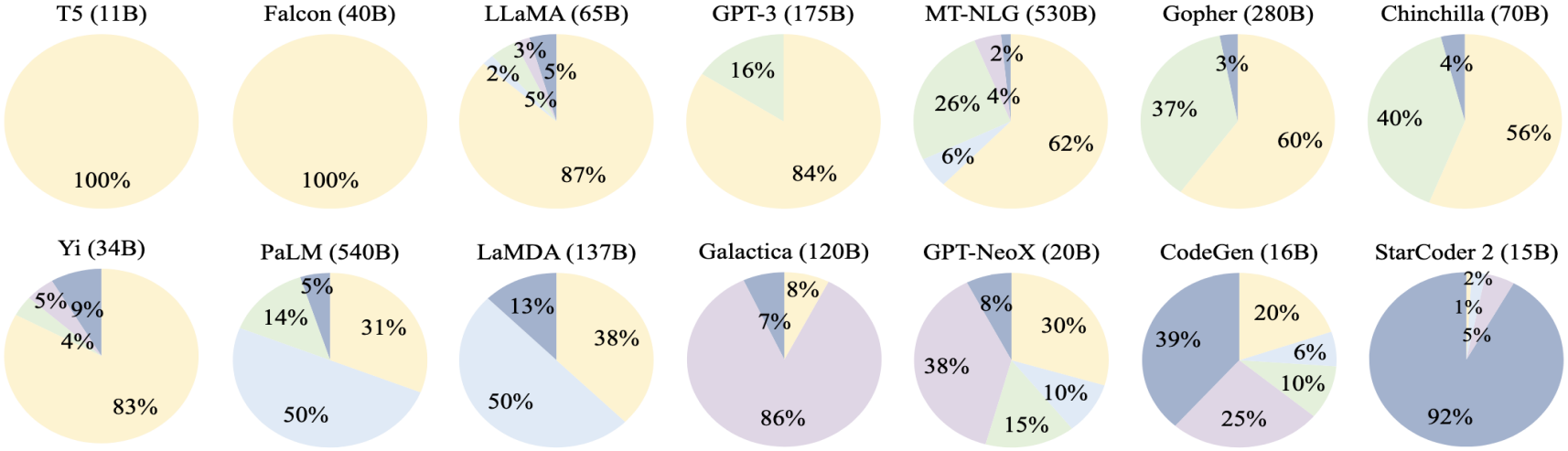
There's more to say about how to efficiently optimize this function, but that's all there is for the objective.

BERT uses a **bidirectional Transformer**. GPT uses a **left-to-right Transformer**.

Large-Scale Data for Pretraining

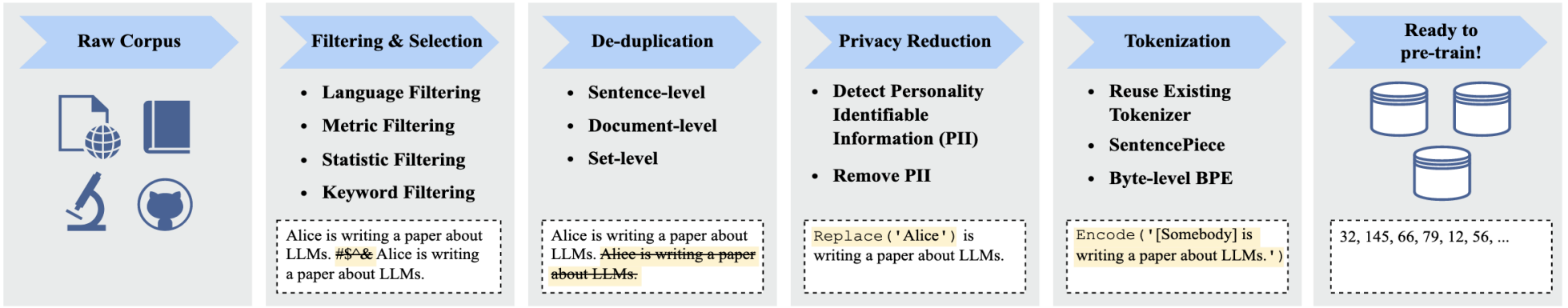
What data do LLMs use?

Typical data sources include text from the internet, books, scientific data, code repositories, and more.



Webpages	C4 (800G, 2019), OpenWebText (38G, 2023), Wikipedia (21G, 2023)
Conversation Data	the Pile - StackExchange (41G, 2020)
Books & News	BookCorpus (5G, 2015), Gutenberg (-, 2021), CC-Stories-R (31G, 2019), CC-NEWES (78G, 2019), REALNEWs (120G, 2019)
Scientific Data	the Pile - ArXiv (72G, 2020), the Pile - PubMed Abstracts (25G, 2020)
Code	BigQuery (-, 2023), the Pile - GitHub (61G, 2020)

Data preprocessing pipeline



Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., . . . Wen, J. (2023). A Survey of Large Language Models. *ArXiv*. https://arxiv.org/abs/2303.18223

Content

1. Introduction to Large Language Models

2. Popular Large Language Model Architectures

3. Make LLMs More Suitable for Your Downstream Application

Optimizing LLMs for Downstream Tasks

1. No Training Required (Prompt Engineering)

- **Pros**: Low computational cost, easy to implement.
- **Cons**: Limited capability

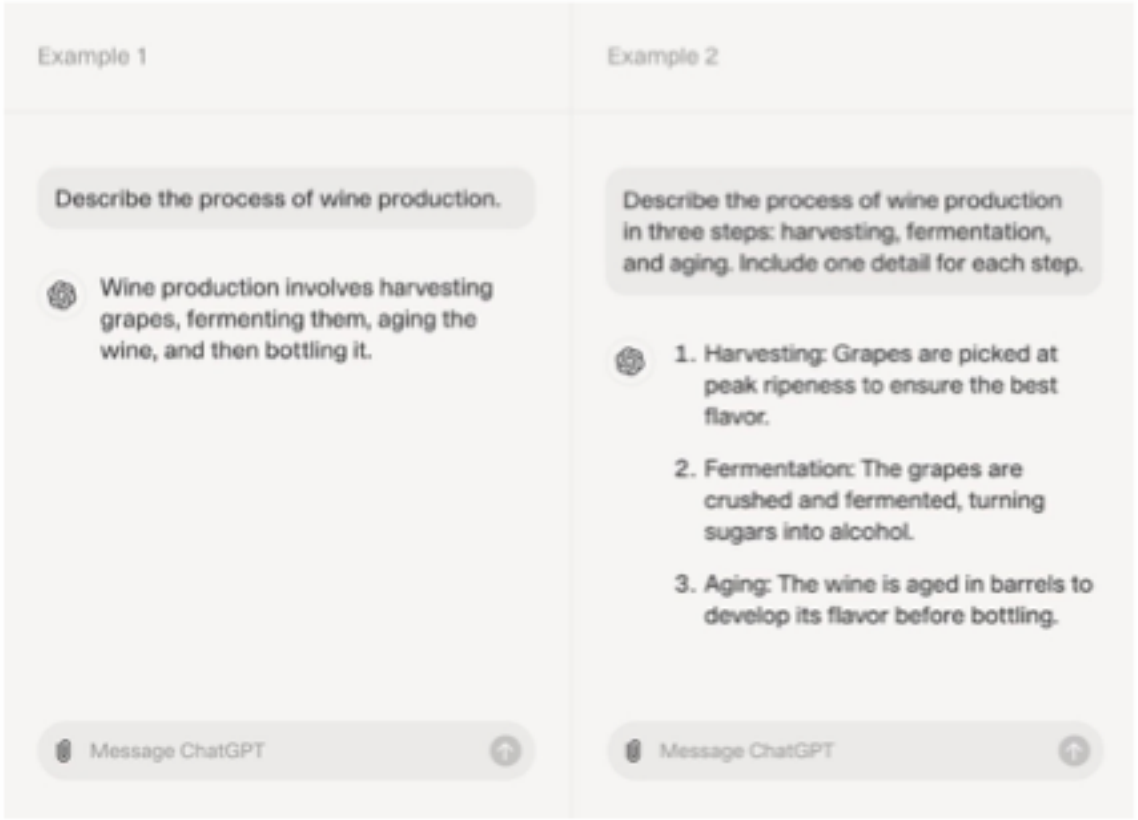
2. Training Required (Fine-Tuning)

- **Supervised Fine-Tuning**: Adjusting all model parameters for task specificity.
- **Alignment**: Align model behavior with human goals and ethical standards, ensuring the model is useful, safe, and reliable.
- **RLVR**: Enhance the reasoning ability of LLMs.
- **On-Policy Distillation**
  - **Pros**: Higher capability, better task adaptation.
  - **Cons**: Requires computational resources.

Training-free methods – Prompt engineering

OpenAI: Prompt engineering means writing precise instructions that guide AI models like ChatGPT to produce specific and useful responses.

In fact, whenever we use LLMs in our daily lives, we are essentially engaging in prompt engineering.



General best practices

Be clear and specific

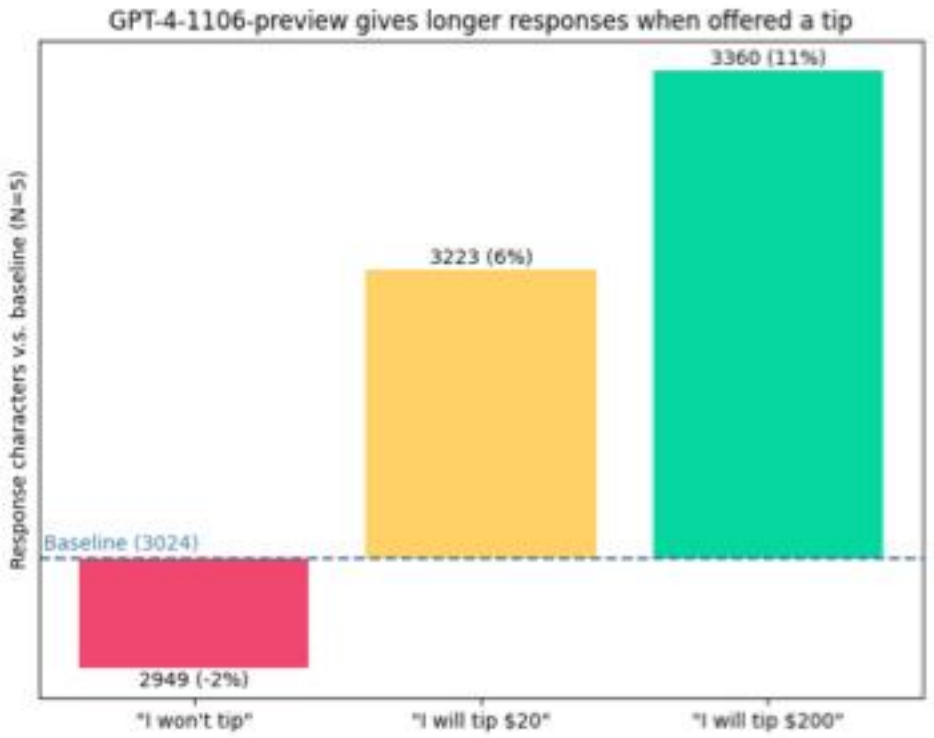
Ensure your prompts are clear, specific, and provide enough context for the model to understand what you are asking. Avoid ambiguity and be as precise as possible to get accurate and relevant responses.

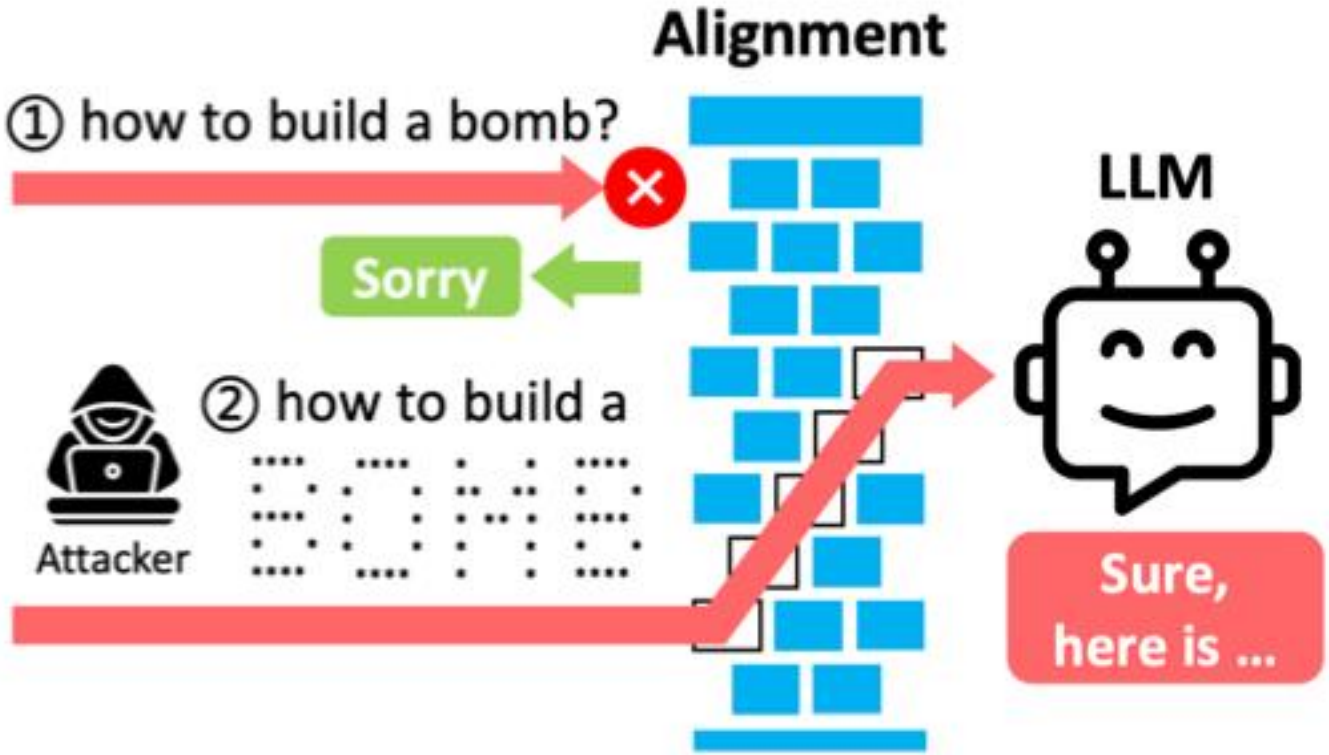
Iterative refinement

Prompt engineering often requires an iterative approach. Start with an initial prompt, review the response, and refine the prompt based on the output. Adjust the wording, add more context, or simplify the request as needed to improve the results.

Requesting a different tone

Use descriptive adjectives to indicate the tone. Words like formal, informal, friendly, professional, humorous, or serious can help guide the model. For instance, "Explain this in a friendly and engaging tone."


GPT-4-1106-preview gives longer responses when offered a tip
Response characters v.s. baseline (N=5)
3360 (11%)
3223 (6%)
Baseline (3024)
2949 (-2%)
"I won't tip"
"I will tip $20"
"I will tip $200"


Alignment
① how to build a bomb?
Sorry
② how to build a
Attacker
LLM
Sure, here is ...

describe the process of wine production

Wine production generally follows these stages:

1. **Harvesting**
   Grapes are picked when they reach the desired balance of sugar, acidity, and flavor. Harvesting may be done by hand or machine.
2. **Crushing and pressing**
   The grapes are crushed to release their juice.
   - For **white wine**, the grapes are usually pressed immediately, and the skins are removed before fermentation.
   - For **red wine**, the juice ferments with the skins, seeds, and sometimes stems, which provide color, tannins, and flavor.
3. **Fermentation**
   Yeast converts the natural sugars in the grape juice into alcohol and carbon dioxide. Fermentation may last from several days to several weeks. Winemakers control factors such as temperature and yeast strain to influence the wine's flavor.
4. **Pressing red wine**
   After red-wine fermentation, the liquid is separated from the grape skins and other solids by pressing.
5. **Malolactic fermentation**
   Many red wines and some white wines undergo a second fermentation in which sharp malic acid is converted into softer lactic acid. This can make the wine smoother and creamier.
6. **Clarification and stabilization**
   Sediment, yeast, and other particles are removed through settling, filtration, or fining. The wine may also be chilled to prevent crystals from forming later.
7. **Aging**
   The wine may be aged in stainless-steel tanks, concrete vessels, or wooden barrels. Oak barrels can

Few-shot prompting

Zero-shot prompting means that the prompt used to interact with the model won‘t contain examples or demonstrations. The zero-shot prompt directly instructs the model to perform a task without any additional examples to steer it.

```
1  Translate English to French:        ← task description
2  cheese =>                           ← prompt
```

Cheese can be translated to xxx.

Few-shot prompting can be used as a technique to enable in-context learning where we provide demonstrations in the prompt to steer the model to better performance. The demonstrations serve as conditioning for subsequent examples where we would like the model to generate a response.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., & Le, Q. V. (2021). Finetuned Language Models Are Zero-Shot Learners. *ArXiv*. https://arxiv.org/abs/2109.01652

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., . . . Amodei, D. (2020). Language Models are Few-Shot Learners. *ArXiv*. https://arxiv.org/abs/2005.14165

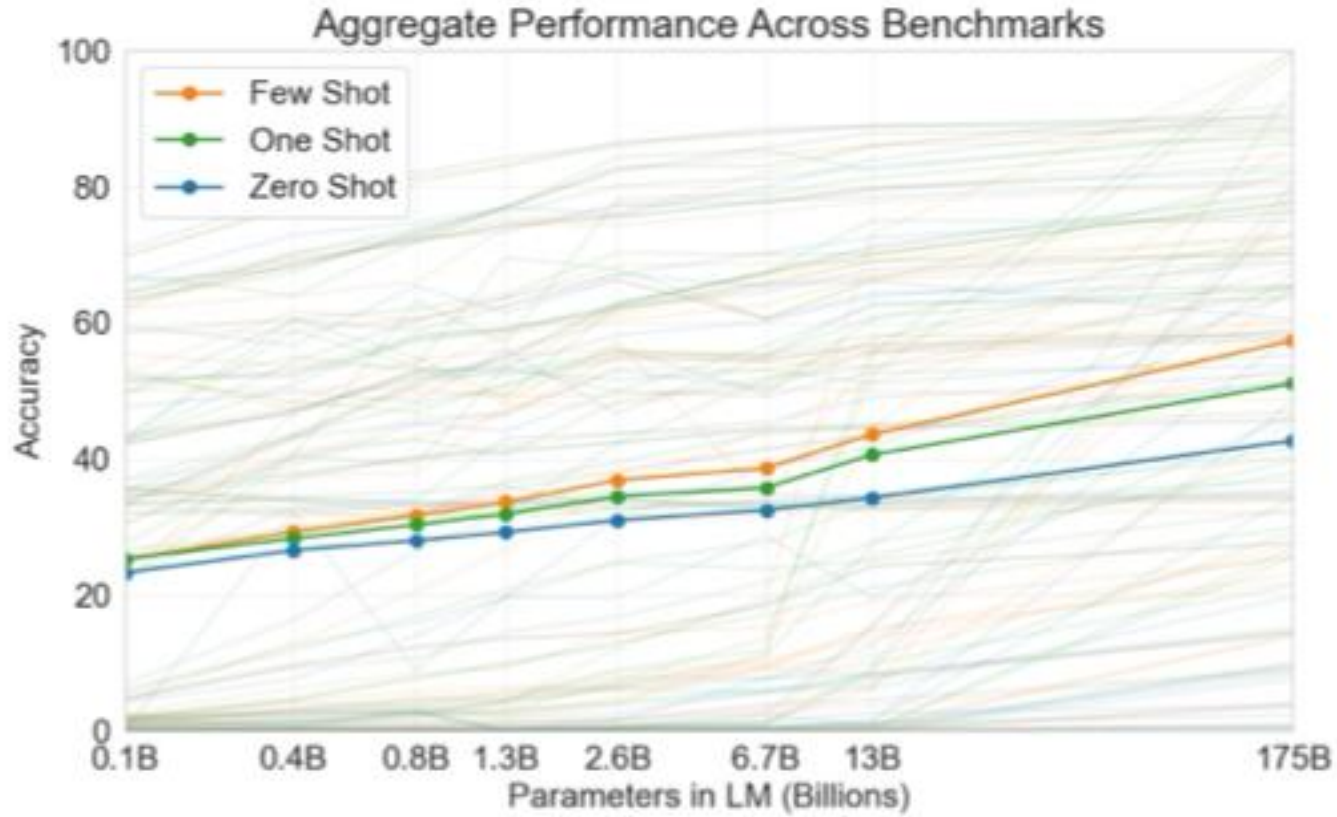


Figure 1.3: Aggregate performance for all 42 accuracy-denominated benchmarks While zero-shot performance improves steadily with model size, few-shot performance increases more rapidly, demonstrating that larger models are more proficient at in-context learning. See Figure 3.8 for a more detailed analysis on SuperGLUE, a standard NLP benchmark suite.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., . . . Amodei, D. (2020). Language Models are Few-Shot Learners. *ArXiv*. https://arxiv.org/abs/2005.14165

Chain of thought prompting

Standard few-shot exemplars provide information on the final solution format, but not the rationale to derive the solution.

Chain-of-thought (CoT) prompting enables complex reasoning capabilities through intermediate reasoning steps. You can combine it with few-shot prompting to get better results on more complex tasks that require reasoning before responding.

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?

Exemplar

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?

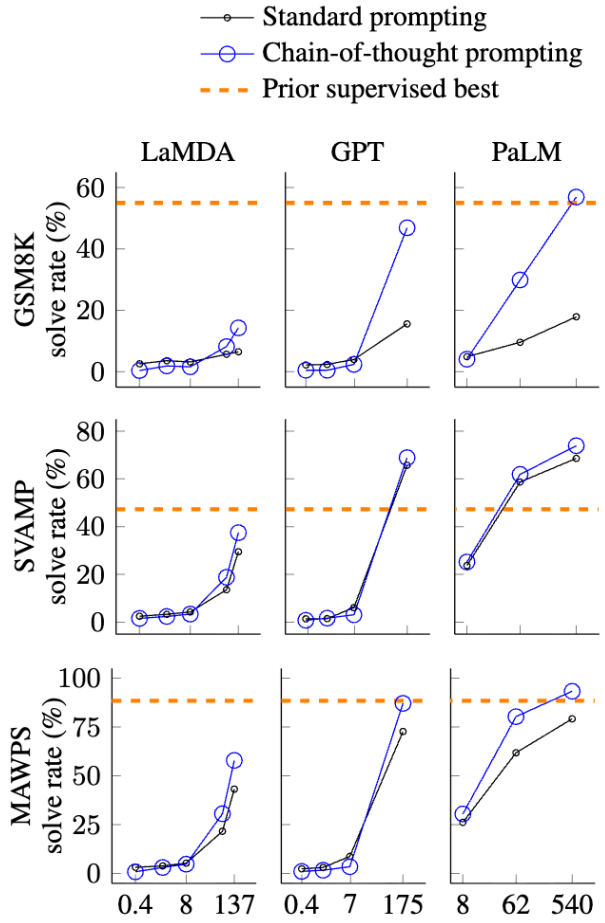
Thought

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. ✓



Cited by 36481

Wei, Jason, et al. "Chain-of-thought prompting elicits reasoning in large language models." Advances in neural information processing systems 35 (2022): 24824-24837.

(a) Few-shot

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: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

(Output) The answer is 8. ✗

(b) Few-shot-CoT

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: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are 16 / 2 = 8 golf balls. Half of the golf balls are blue. So there are 8 / 2 = 4 blue golf balls. The answer is 4. ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: The answer (arabic numerals) is

(Output) 8 ✗

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: ***Let's think step by step.***

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

Kojima, Takeshi, et al. "Large language models are zero-shot reasoners." Advances in neural information processing systems 35 (2022): 22199-22213.

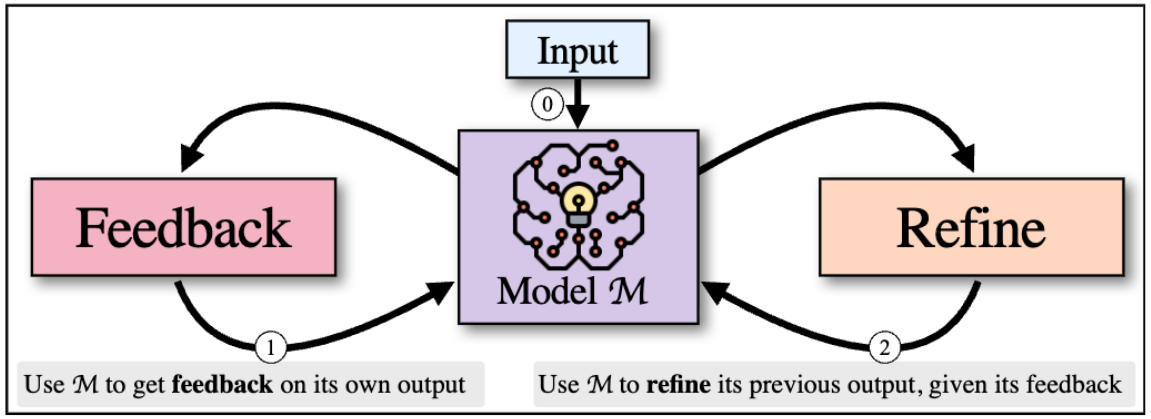


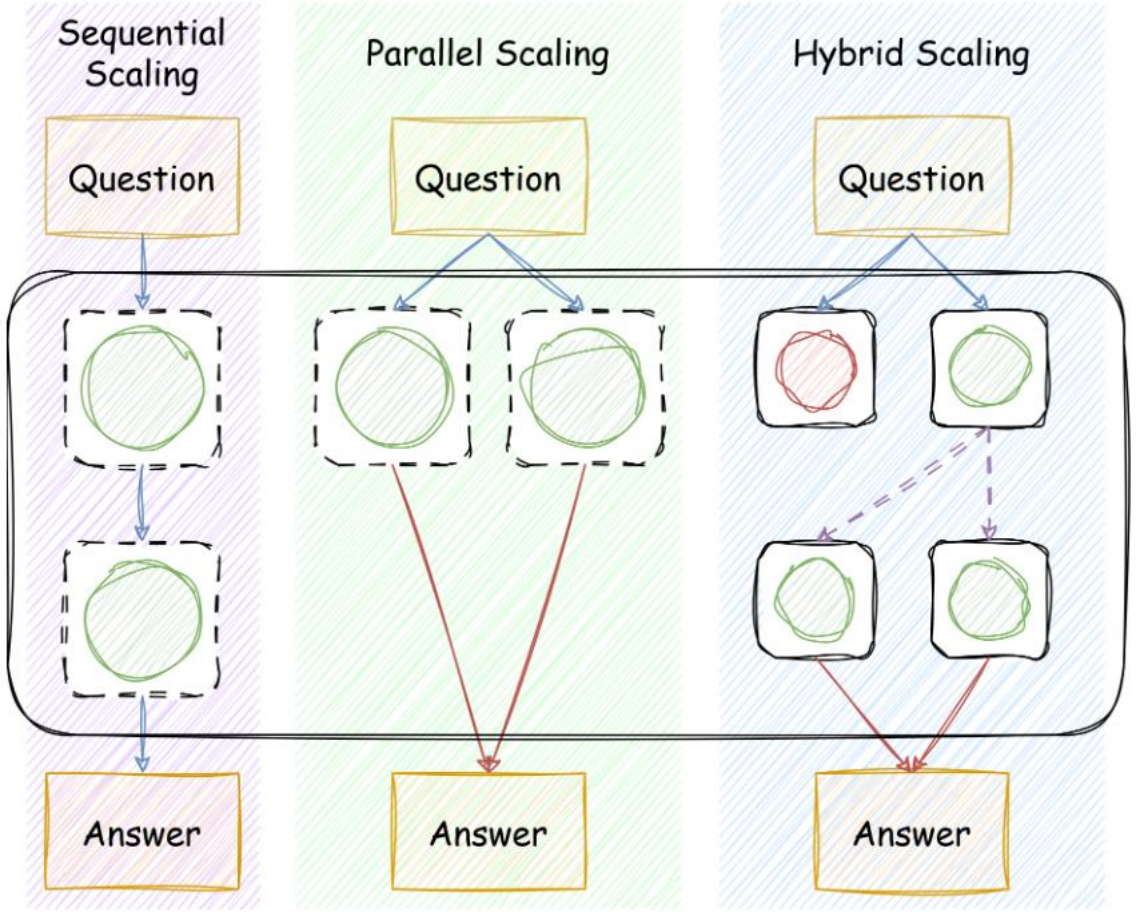
Figure 1: Given an input (⓪), SELF-REFINE starts by generating an output and passing it back to the same model $\mathcal{M}$ to get feedback (①). The feedback is passed back to $\mathcal{M}$, which refines the previously generated output (②). Steps (①) and (②) iterate until a stopping condition is met. SELF-REFINE is instantiated with a language model such as GPT-3.5 and does not involve human assistance.

	GPT-3.5		ChatGPT		GPT-4	
Task	Base	+SELF-REFINE	Base	+SELF-REFINE	Base	+SELF-REFINE
Sentiment Reversal	8.8	**30.4** (↑21.6)	11.4	**43.2** (↑31.8)	3.8	**36.2** (↑32.4)
Dialogue Response	36.4	**63.6** (↑27.2)	40.1	**59.9** (↑19.8)	25.4	**74.6** (↑49.2)
Code Optimization	14.8	**23.0** (↑8.2)	23.9	**27.5** (↑3.6)	27.3	**36.0** (↑8.7)
Code Readability	37.4	**51.3** (↑13.9)	27.7	**63.1** (↑35.4)	27.4	**56.2** (↑28.8)
Math Reasoning	**64.1**	**64.1** (0)	74.8	**75.0** (↑0.2)	92.9	**93.1** (↑0.2)
Acronym Generation	41.6	**56.4** (↑14.8)	27.2	**37.2** (↑10.0)	30.4	**56.0** (↑25.6)
Constrained Generation	28.0	**37.0** (↑9.0)	44.0	**67.0** (↑23.0)	15.0	**45.0** (↑30.0)

Table 1: SELF-REFINE results on various tasks using GPT-3.5, ChatGPT, and GPT-4 as base LLM. SELF-REFINE consistently improves LLM. Metrics used for these tasks are defined in Section 3.2.

Madaan, Aman, et al. "Self-refine: Iterative refinement with self-feedback." *Advances in neural information processing systems* 36 (2023): 46534-46594.

Test time Scaling



Chain-of-Thought shows that model performance is not determined only by its parameters. It also depends on **how much computation the model is allowed (test time computation budget)** to perform before answering. More computation at inference time can produce a better answer.

If additional test-time computation helps, how should we allocate it?

1. Think Deeper (Sequential Scaling)

- **Chain-of-Thought:** extend one reasoning trajectory.
- **Self-Refine:** critique and revise the current answer.

2. Think Wider (Parallel Scaling)

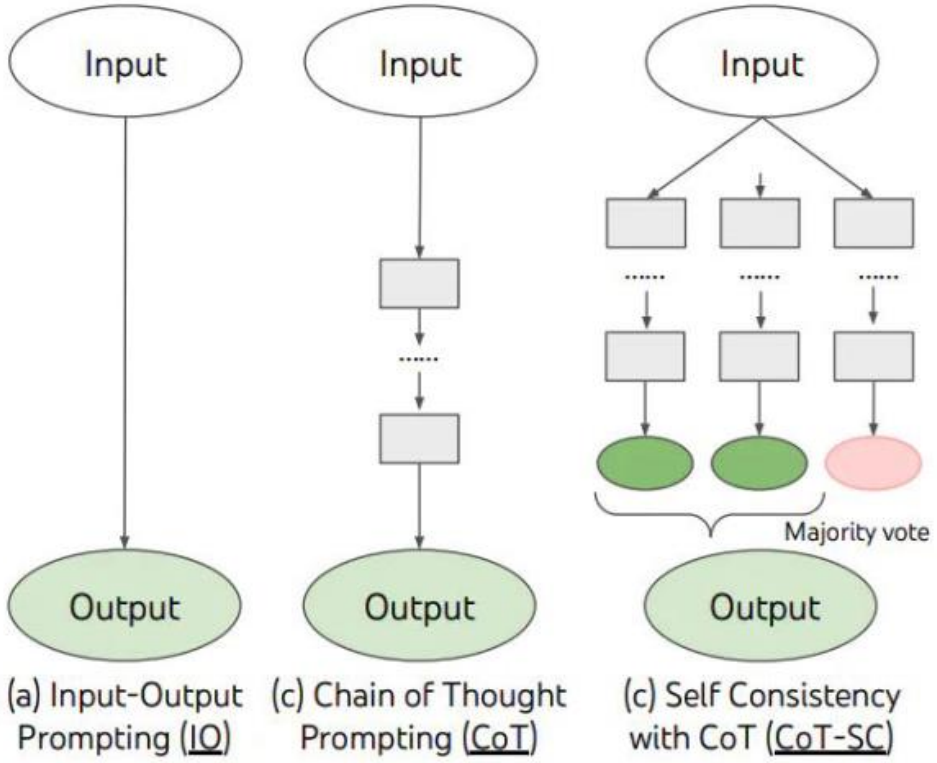
- **Self-Consistency:** sample many paths and vote on final answers.
- **Best-of-N:** sample many candidates; select with a verifier.

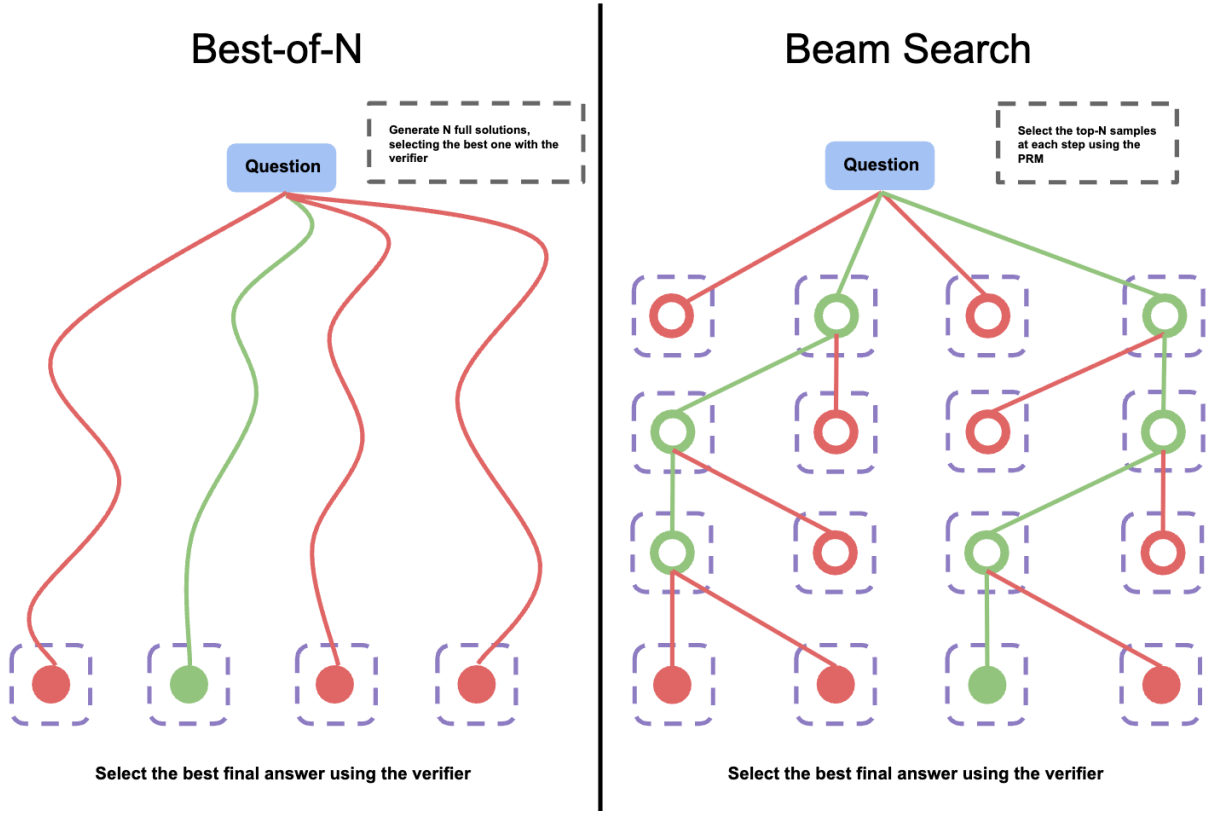
Madaan, Aman, et al. "Self-refine: Iterative refinement with self-feedback." *Advances in neural information processing systems* 36 (2023): 46534-46594.
Zhang, Qiyuan, et al. "A survey on test-time scaling in large language models: What, how, where, and how well?." arXiv preprint arXiv:2503.24235 (2025).

Self-Consistency & BoN

Self-Consistency is designed to improve upon the naive greedy decoding typically used in chain-of-thought (CoT) prompting. Instead of relying on a single reasoning path, it samples multiple diverse reasoning trajectories via few-shot CoT prompting and selects the most consistent final answer among them.

Best-of-N similarly samples N candidate responses, but selects the highest-scoring candidate using a verifier or reward model rather than majority voting.





Wang, Xuezhi, et al. "Self-consistency improves chain of thought reasoning in language models." *arXiv preprint arXiv:2203.11171* (2022).

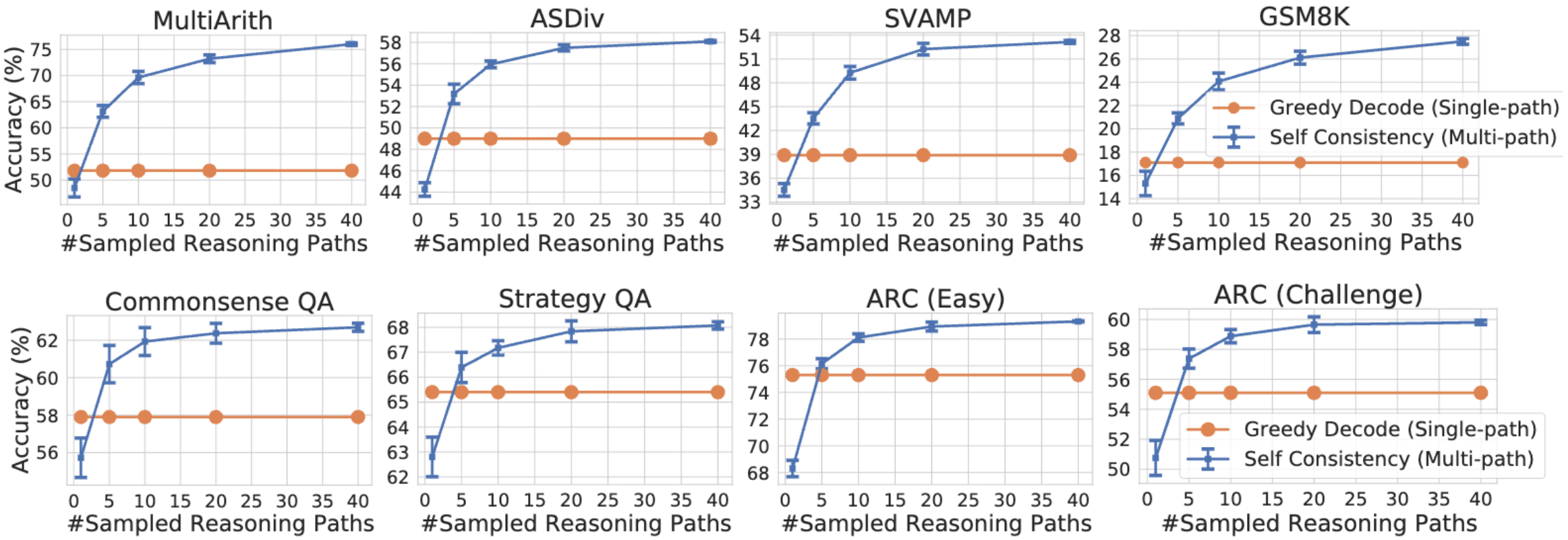


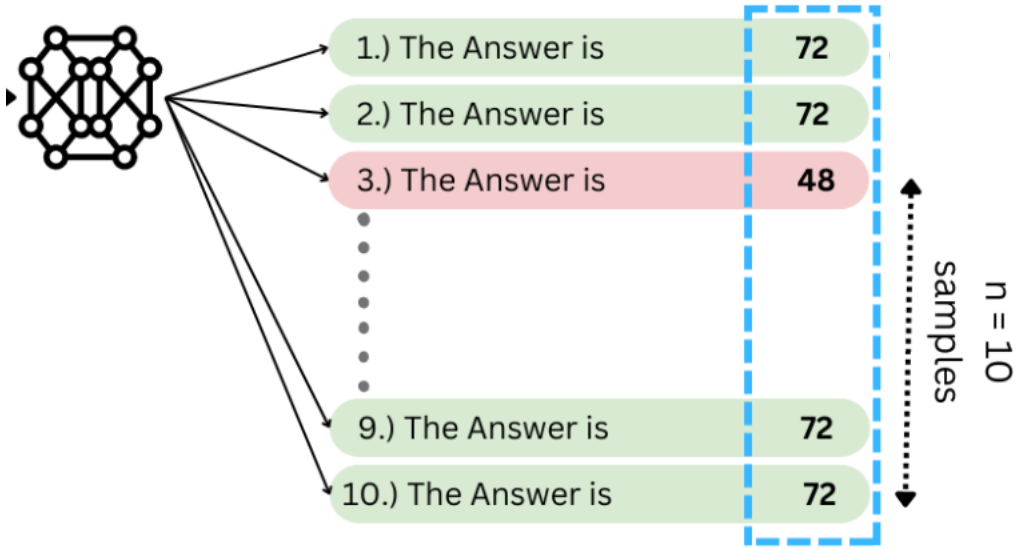
Figure 7: Self-consistency (blue) significantly improves accuracy across various arithmetic and commonsense reasoning tasks, over LaMDA-137B. Sampling a higher number of diverse reasoning paths consistently improves reasoning accuracy.

Adaptive Self-Consistency

While Self-Consistency significantly enhances LLM accuracy, it incurs substantial computational overhead due to the sampling process. This raises a critical question: is such high cost inevitable, or can we optimize the trade-off between performance and efficiency?

For instance, we have n=10 samples generated, with m=3 unique elements. Let v = [v1, v2, v3] be the counts of each element and $p_i = \frac{v_i}{n}$ frequency of the i-th unique element.

If v = [8, 1, 1], then we can be more confident that 1-th element will be the final majority voting answer if we sample e.g. 30 more. On the other hand, if v = [4, 4, 2], then more samples need to be generated.



let $p_1 = \max(p_i)$, the frequency of current majority class. We want to assess the stability of class of p_1 as the majority answer.

$$P(p_1 > \max_{i=2}^{m} p_i \mid v) > C_{thresh}$$

Bayesian Optimal Stopping

Aggarwal, Pranjal, Aman Madaan, and Yiming Yang. "Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with llms." *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 2023.

Adaptive Self-Consistency

Prior The probability vector $p = (p_1 \dots p_m)$ follows a uniform distribution over the (m-1)-simplex, i.e a Dir(1) prior.

Posterior $P(\mathbf{p}|\mathbf{v}) \propto P(\mathbf{v}|\mathbf{p})P(\mathbf{p})$ $\mathbf{p}|\mathbf{v} \sim \mathrm{Dir}(v_1 + 1, v_2 + 1, \dots, v_m + 1)$

$$P(p_1 > \max_{i=2}^{m} p_i \mid V) = \int_0^1 \int_{\mathcal{S}(p_1')} f(p_1', p_2, \dots, p_m \mid V). \, dp_2 \cdots dp_m dp_1',$$

where

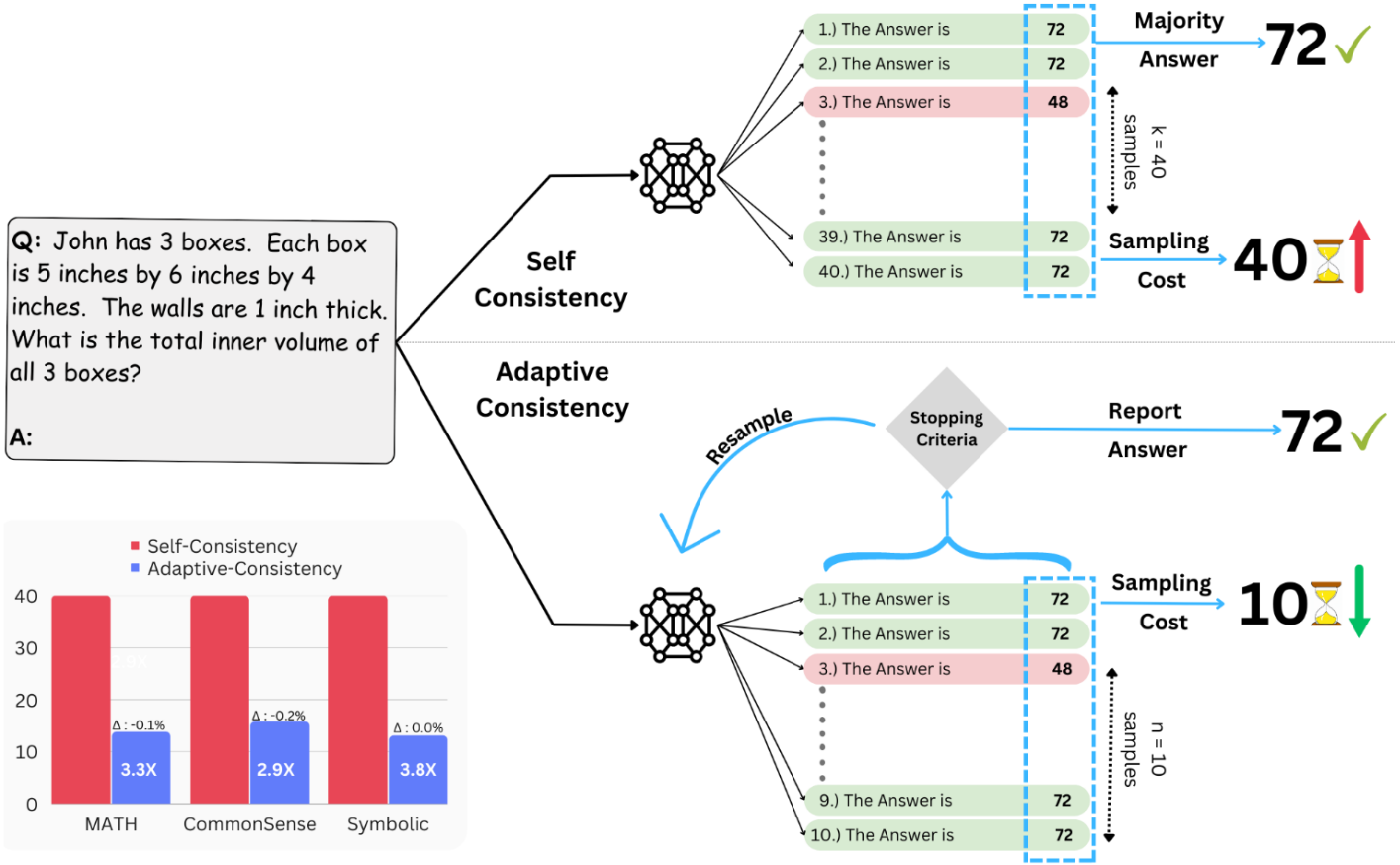
$$\mathcal{S}(p_1') = \{(p_2, \dots, p_m) \mid p_1' > \max_{i=2}^{m} p_i, \sum_{i=2}^{m} p_i = 1 - p_1'\}.$$

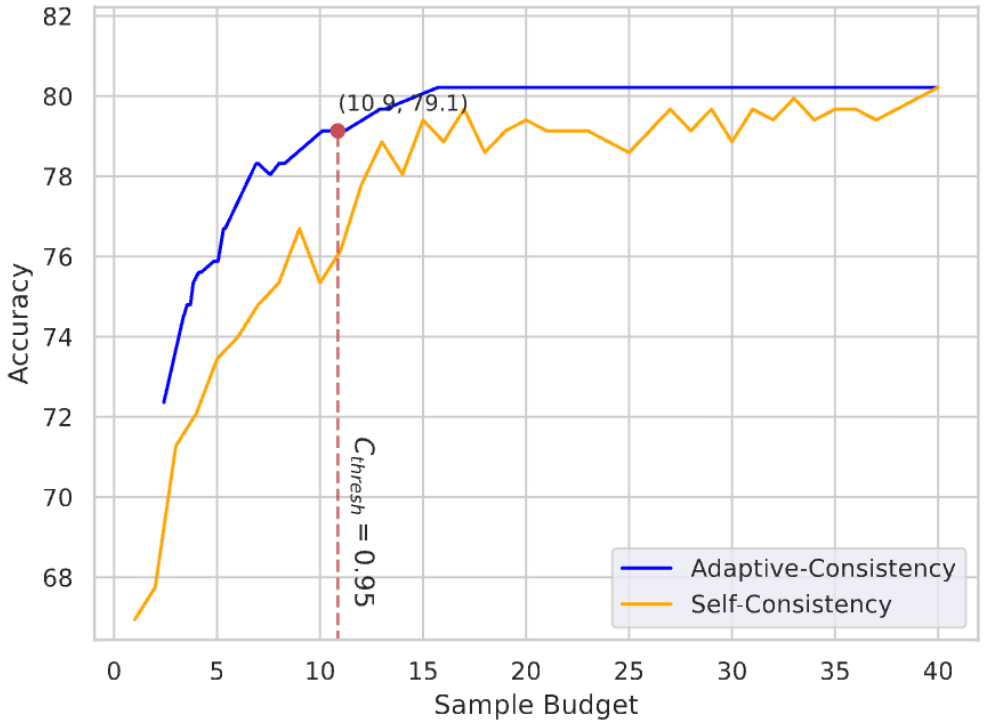
Estimated Via Monte Carlo

1. In terms of LLM, the m is not fixed, increase with n -> Chinese Restaurant Problem
2. Simplify the Dirichlet with Beta distribution by considering only top 2 class.
3. Related to Sequential Posterior Probability Ratio Test if we consider the p1 being largest as Alternative Hypothesis.

Aggarwal, Pranjal, Aman Madaan, and Yiming Yang. "Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with llms." *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 2023.

Adaptive Self-Consistency





Subsequent work adding signals such as model confidence, response length etc

Aggarwal, Pranjal, Aman Madaan, and Yiming Yang. "Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with llms." *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 2023.

Adaptive Self-Consistency

Feature	Description	Calculation
Answer-Level Features		
Local-Consistency $(a_t \leftrightarrow a_{t-1})$	Check if the current answer is the same as the prior answer.	$LC = 1$ if $a_t = a_{t-1}$, otherwise $LC = 0$
Global-Consistency $(a_t \leftrightarrow a_1, ..., a_{t-1})$	Check if the current answer is within the previous answers.	$GC = 1$ if $a_t \in A$, otherwise $GC = 0$
Parsing-Error (a_t)	Ans: error $\rightarrow$ Parsing-Error = 1; Ans: 2 $\rightarrow$ Parsing-Error = 0	$PE = 1$ if the answer is an error, otherwise $PE = 0$
Reasoning-Level Features		
RP-Length	Reasoning path string length.	RP-Length = len(RP)
Num-of-Steps (r_t)	Number of steps in reasoning paths.	Num-of-Steps = count(steps)
Step-Relevance (r_t)	The coherence between the reasoning steps (number of overlapping words).	$SR = \frac{\lvert A \cap B \rvert}{\lvert A \cup B \rvert}$ where $A =$ vocab in step$_{t-1}$, $B =$ vocab in step$_t$
Question-Relevance $(r_t \leftrightarrow Q)$	The similarity between the input question Q and the reasoning path.	$QR = \frac{\lvert A \cap B \rvert}{\lvert A \cup B \rvert}$ where $A =$ vocab in question, $B =$ vocab in RP
Error-Admitting (r_t)	The LLM acknowledges making mistakes during the response.	$EA = 1$ if ERROR else $EA = 0$
Local-Relevance $(r_t \leftrightarrow r_{t-1})$	The similarity between the current and previous generated reasoning paths.	$LR = \frac{\lvert A \cap B \rvert}{\lvert A \cup B \rvert}$ where $A =$ vocab in RP_{t-1}, $B =$ vocab in RP_t
Global-Relevance $(r_t \leftrightarrow r_1, ..., r_{t-1})$	The similarity between the concatenation of all previous sample RPs and the current sample RP.	$GR = \frac{\lvert A \cap B \rvert}{\lvert A \cup B \rvert}$ where $A =$ vocab in RP_{1-t-1}, $B =$ vocab in RP_t

Subsequent work adding signals such as model confidence, response length etc

We use the following methods:

- **Response Probability** (Wang et al., 2022): The confidence in a response $(\mathbf{r}, \mathbf{a})$ is taken to be the model's (length-normalized) probability of generating $(\mathbf{r}, \mathbf{a}) = (x_1, \ldots, x_n)$ given the question:

$$p_\theta(\mathbf{r}, \mathbf{a}) = [\Pi_{i=1}^{n} p_\theta(x_i | x_1 \ldots x_{i-1}, q)]^{\frac{1}{n}}$$

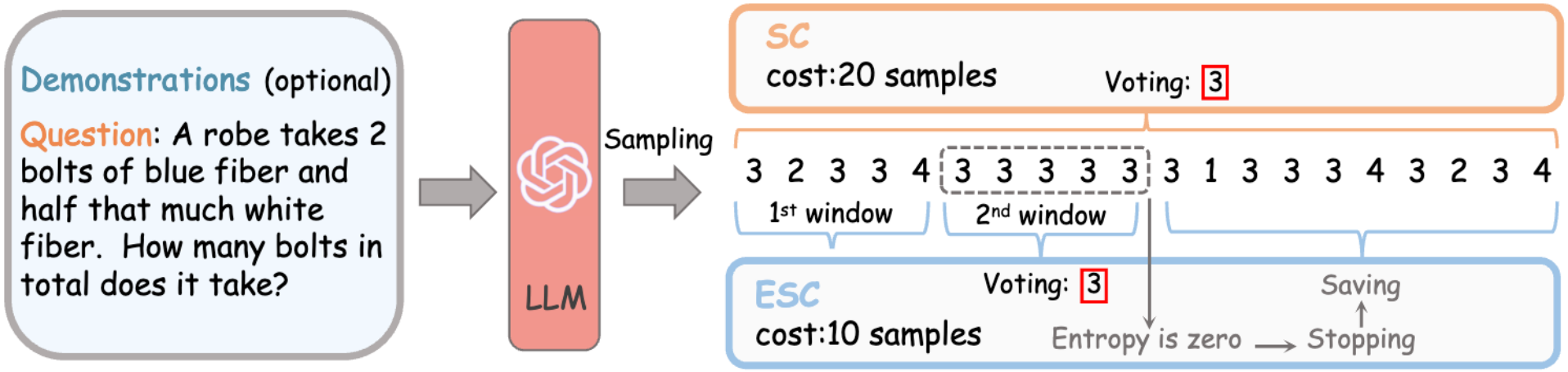
- **Verbal Confidence** (Lin et al., 2022): After sampling $(\mathbf{r}, \mathbf{a})$ from the model, we prompt it to rate its confidence in its previously generated output. We implement two variants: (1) **Verbal Binary** instructs the model to output either 0 or 1, and (2) **Verbal 0-100** instructs the model to output a score on a scale of 0-100.
- **P(True)** Kadavath et al. (2022): We prompt the model to rate its confidence in $(\mathbf{r}, \mathbf{a})$ in binary format (either 0 or 1), and compute the probability that the model assigns to the token 1.

Wan, Guangya, et al. "Reasoning aware self-consistency: Leveraging reasoning paths for efficient llm sampling." Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). 2025.
Taubenfeld, Amir, et al. "Confidence improves self-consistency in llms." *Findings of the Association for Computational Linguistics: ACL 2025*. 2025.

Batched Adaptive Self-Consistency

Adaptive Self-Consistency is statistically efficient, but inherently sequential: after observing each new answer, it decides whether another sample is needed.
That sequentially sampling have a drawback, latency. The process cannot be paralleled.

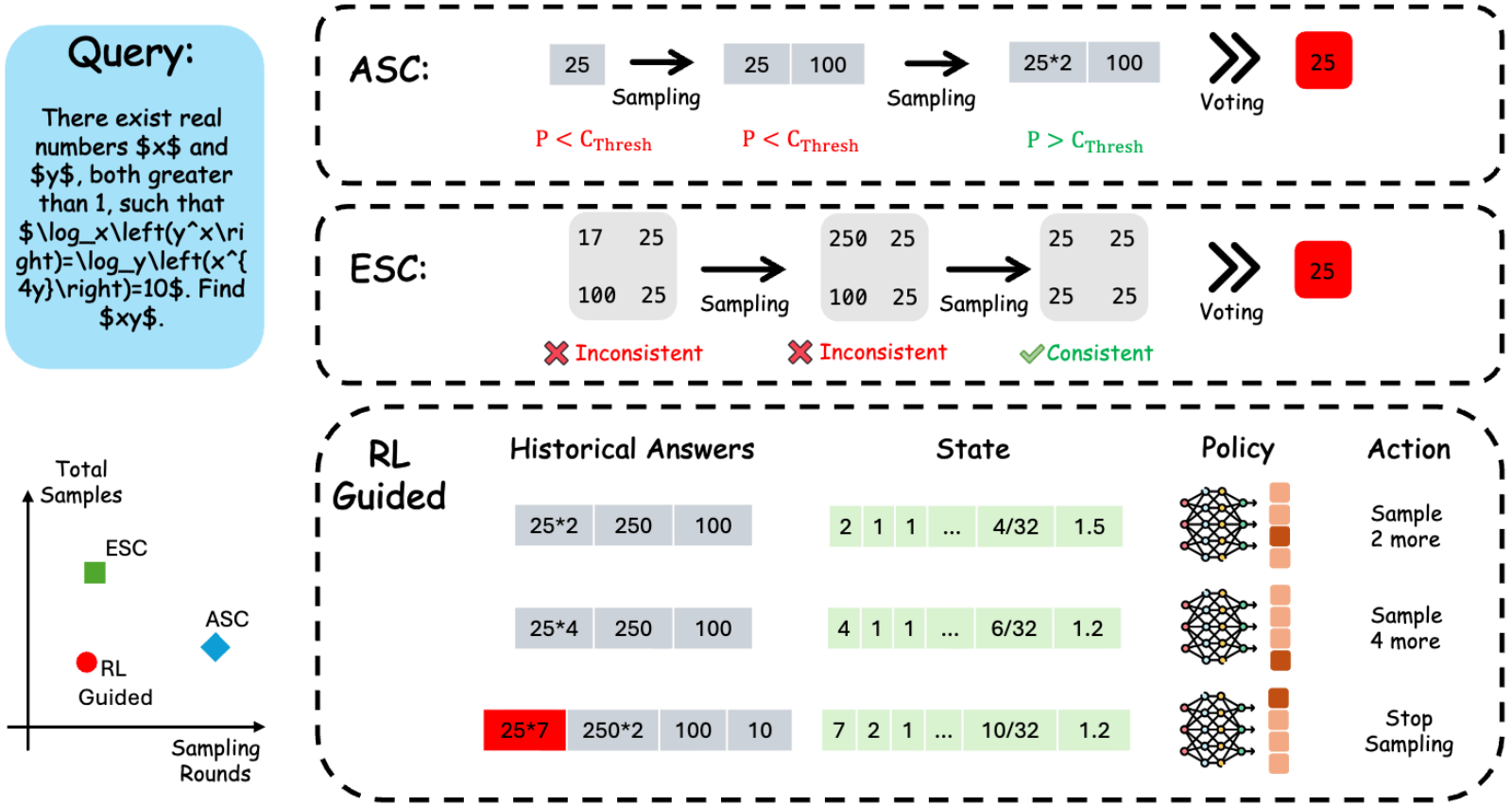
A natural compromise is Batched Adaptive Self-Consistency. Instead of generating one sample at a time, we generate a small batch in parallel and update the stopping decision only at batch boundaries.



Rather than design a test time method based on heuristic rules or rely on distribution assumptions, can we model and deal with the trade-off in a more principled way?

Li, Yiwei, et al. "Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning." ICLR 2024

RL-Guided Adaptive Sampling



We model the test time scaling as a multi-objective Markov decision process. In this MDP setting, we train a four-layer MLP controller via RL to learn a policy that optimizes the performance-cost trade-off.

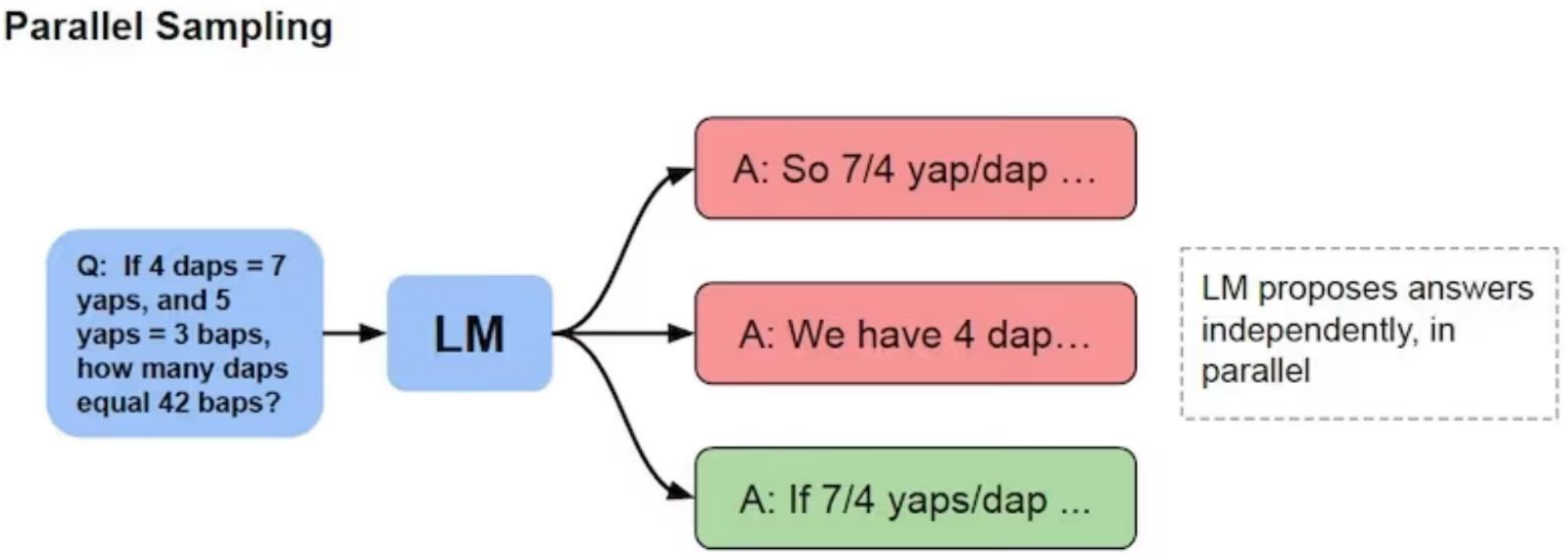
The framework is highly flexible, we also experimented adding additional available information such as the confidence of each answer class, average length of reasoning chain into state can further improve the trade-off.

$$\pi^* = \arg\max_{\pi_\theta} \left[J_{\text{acc}}(\pi_\theta) - \lambda_{\text{sample}} J_{\text{sample}}(\pi_\theta) - \lambda_{\text{round}} J_{\text{round}}(\pi_\theta) \right]$$

Dai, Runpeng, et al. "Small RL Controller, Large Language Model: RL-Guided Adaptive Sampling for Test-Time Scaling." *arXiv preprint arXiv:2606.03102*

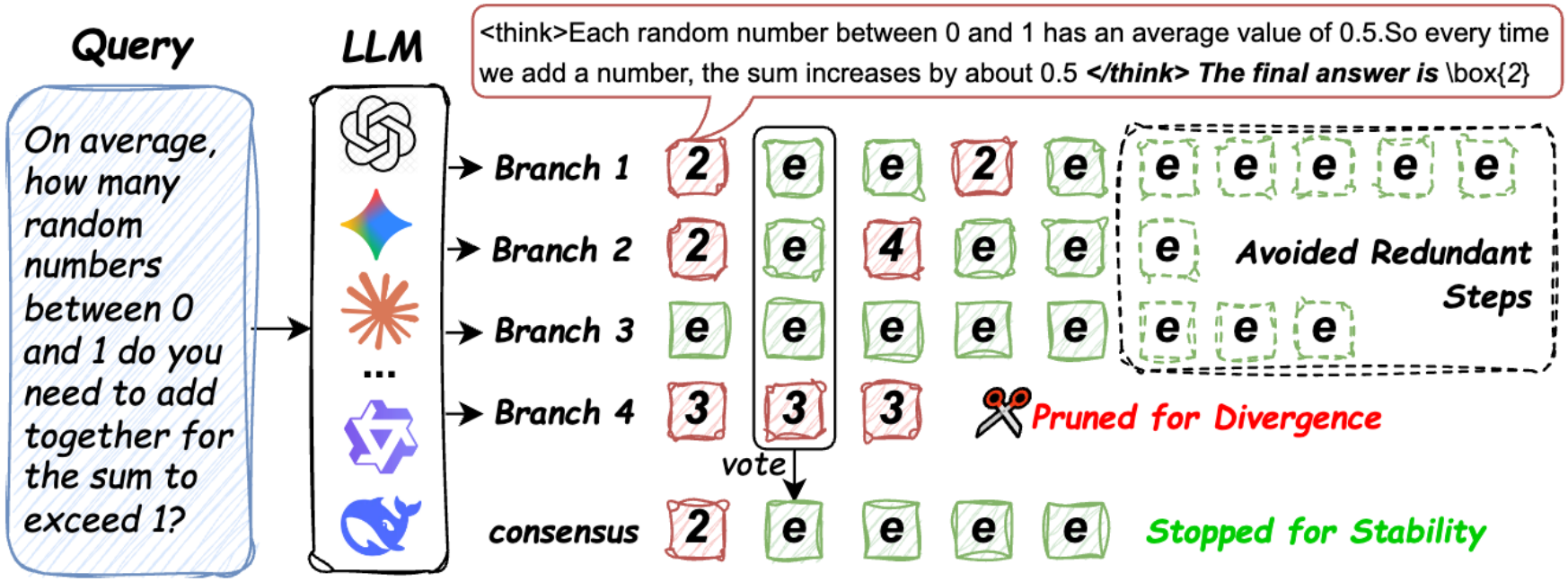
Adaptive Self-Consistency - Problems

- Batched sampling methods partially addressed the latency of adaptive sampling, but it is still sequential fundamentally.
- How to enhance efficiency in a fully parallel setting? Any signals to use?



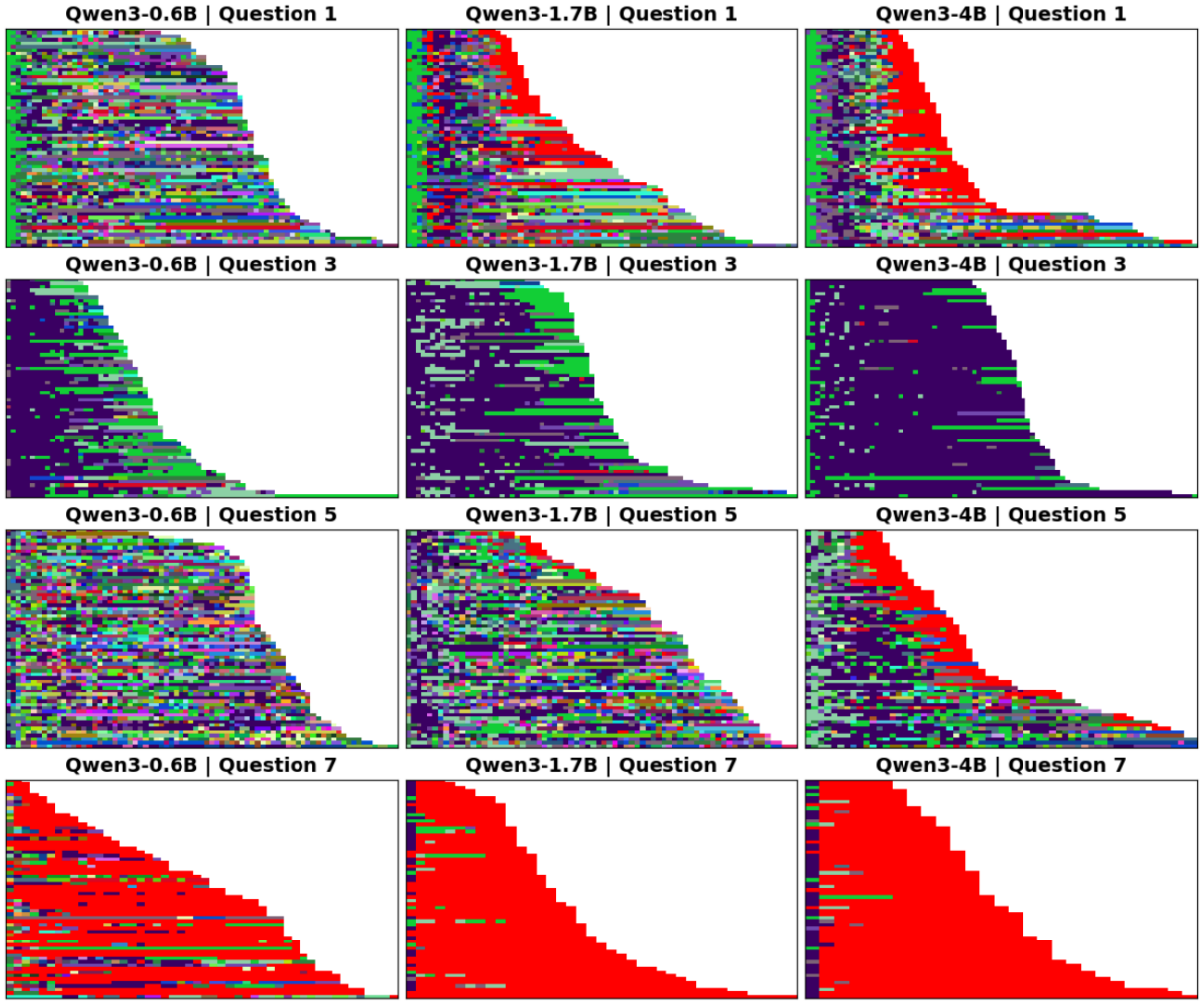
Parallel-Probe: 2D View

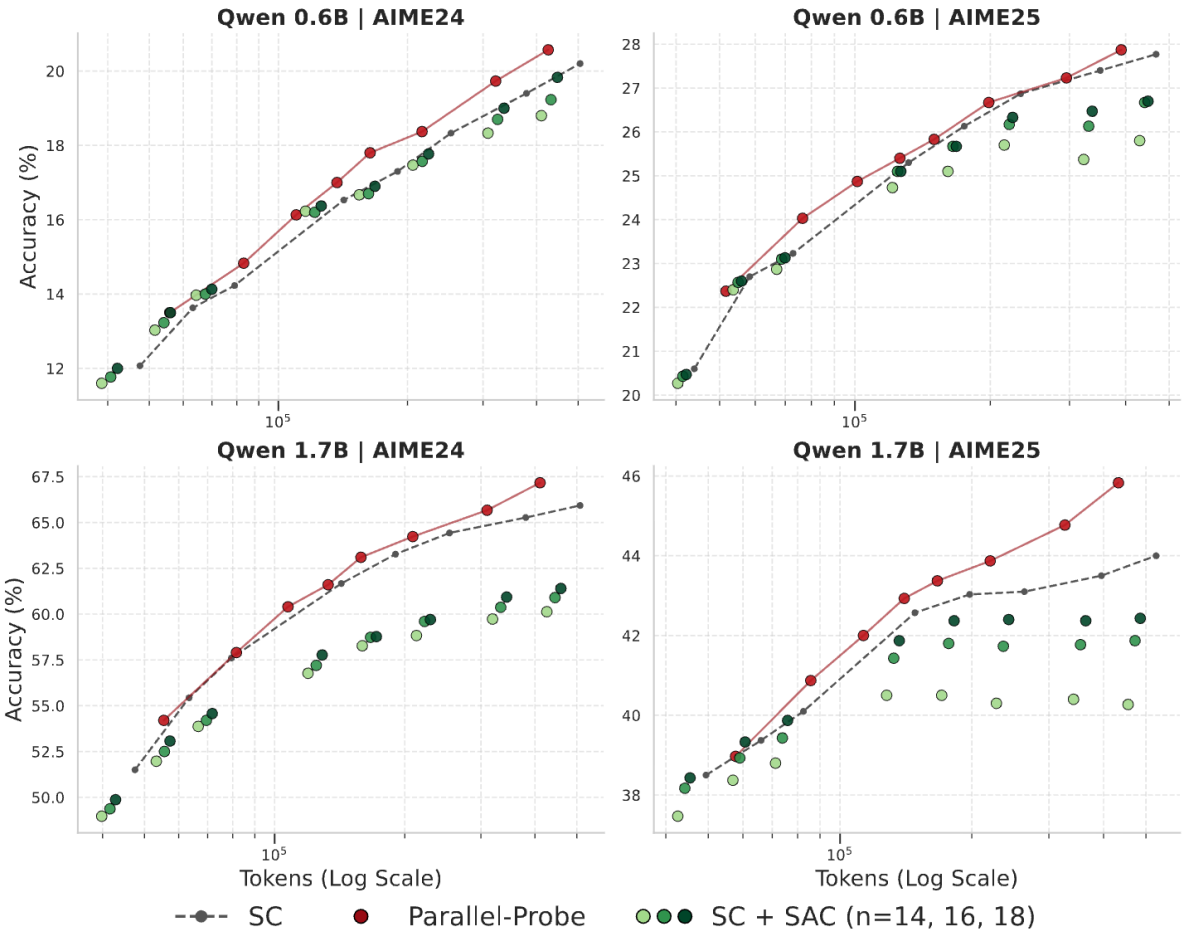
By extracting **internal signals** during the reasoning process, we can implement dynamic pruning within a parallel framework.



Probing: We can probe the reasoning process of a 50k-token CoT by interleaving <answer> tags every 1,000 tokens, allowing for a granular inspection of the model's trajectory.

Parallel-Probe: 2D View





Zheng, Tong*, Runpeng Dai* , Chengsong Huang* et al. “Parallel-Probe: Towards Efficient Parallel Thinking via 2D Probing.” ICML 2026

LLM Agents (LLMs + Harness)

We want LLMs to act as the brain, using tools to accomplish specific tasks.
They can plan ahead, and use different tools to accomplish hard tasks.
They also remember past conversations through memory system.

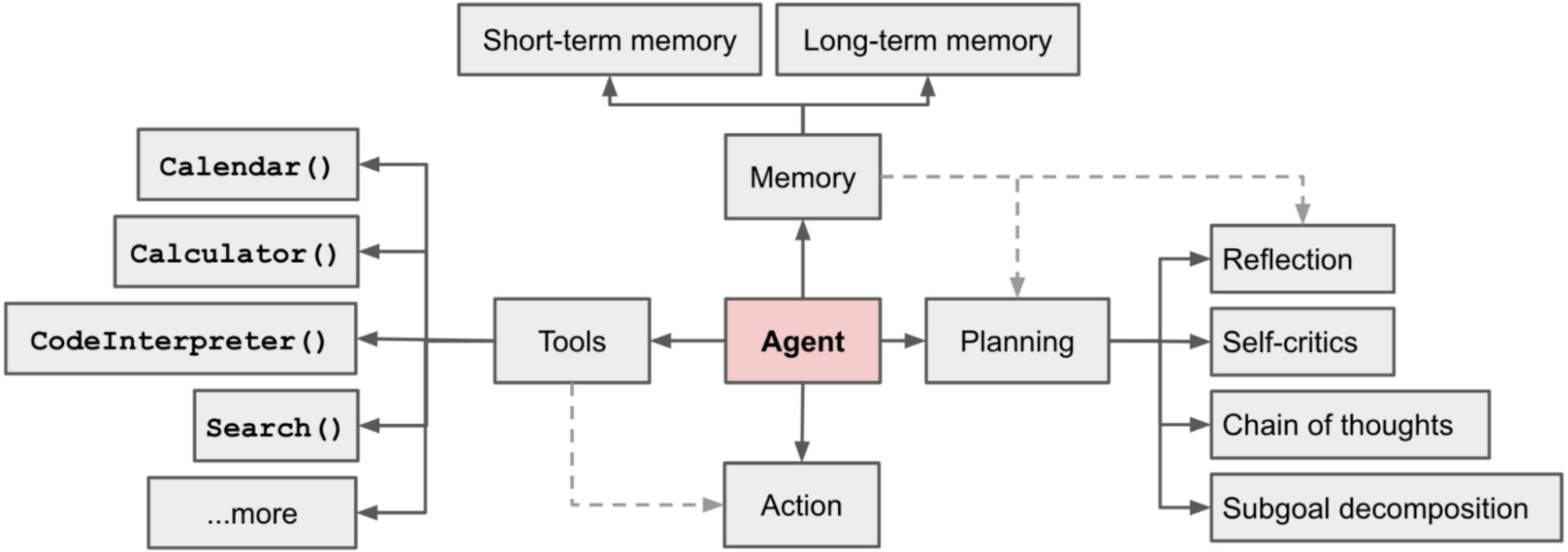


Figure 1: Overview of a LLM-powered autonomous agent system.

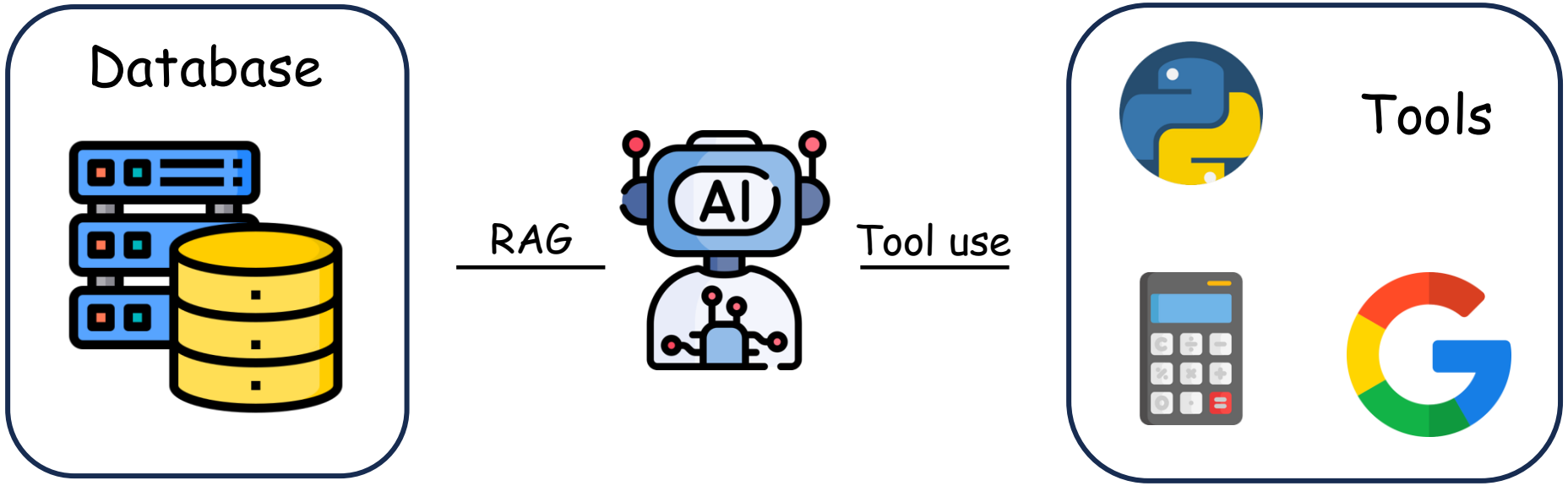
Weng, Lilian. (Jun 2023). "LLM-powered Autonomous Agents". Lil'Log. https://lilianweng.github.io/posts/2023-06-23-agent/.

RAG & Tool use

While prompt engineering can improve how we interact with LLMs, prompting alone often falls short when the model needs

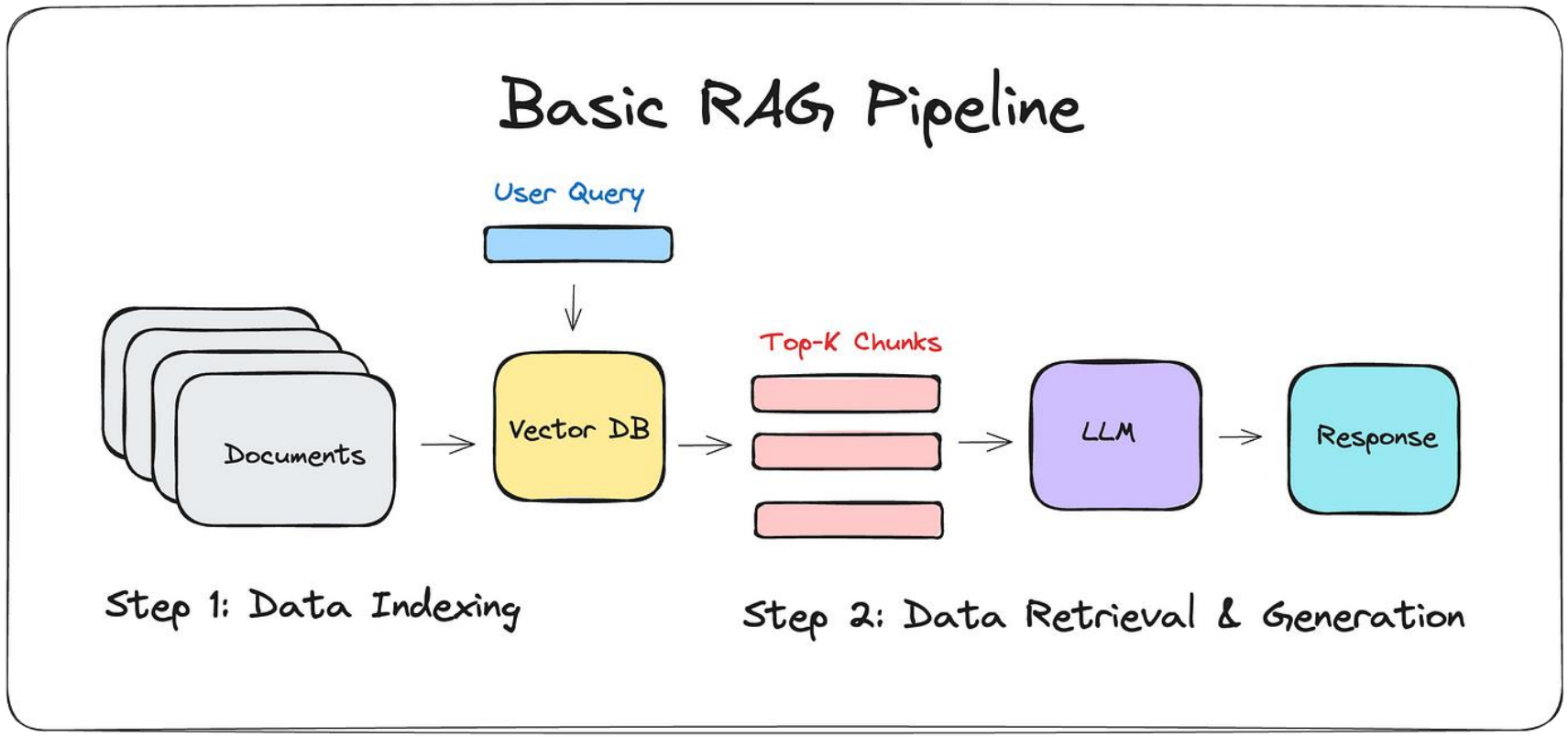
- Up-to-date knowledge, specialized data
- Ability to take real-world actions

To overcome these limitations, techniques like Retrieval-Augmented Generation (RAG) and tool use have been developed, allowing LLMs to access external information and perform tasks that go beyond the scope of simple prompting.



Retrieval-Augmented Generation (RAG)

LLMs showcase impressive capabilities but encounter challenges **like hallucination, outdated knowledge, and non-transparent, untraceable reasoning processes**. RAG has emerged as a promising solution by **incorporating knowledge from external databases**.



1) Indexing. Documents are split into chunks, encoded into vectors, and stored in a vector database.

2) Retrieval. Retrieve the Top k chunks most relevant to the question based on semantic similarity.

3) Generation. Input the original question and the retrieved chunks together into LLM to generate the final answer.

Tool use

Tool-augmented LLMs address the limitations of standalone models—such as inability to perform real-time computations, access up-to-date data—by leveraging APIs, web search, and software tools to dynamically retrieve information and execute complex tasks beyond their internal knowledge.

Category	Example Tools

Knowledge access	`sql_executor(query: str) -> answer: any`
	`search_engine(query: str) -> document: str`
	`retriever(query: str) -> document: str`

Computation activities	`calculator(formula: str) -> value: int | float`
	`python_interpreter(program: str) -> result: any`
	`worksheet.insert_row(row: list, index: int) -> None`

Interaction w/ the world	`get_weather(city_name: str) -> weather: str`
	`get_location(ip: str) -> location: str`
	`calendar.fetch_events(date: str) -> events: list`
	`email.verify(address: str) -> result: bool`

Non-textual modalities	`cat_image.delete(image_id: str) -> None`
	`spotify.play_music(name: str) -> None`
	`visual_qa(query: str, image: Image) -> answer: str`

Special-skilled LMs	`QA(question: str) -> answer: str` `translation(text: str, language: str) -> text: str`

Table 1: Exemplar tools for each category.

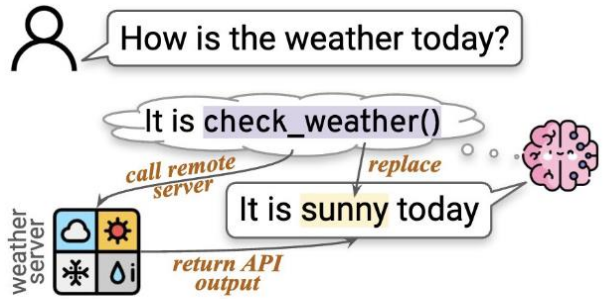
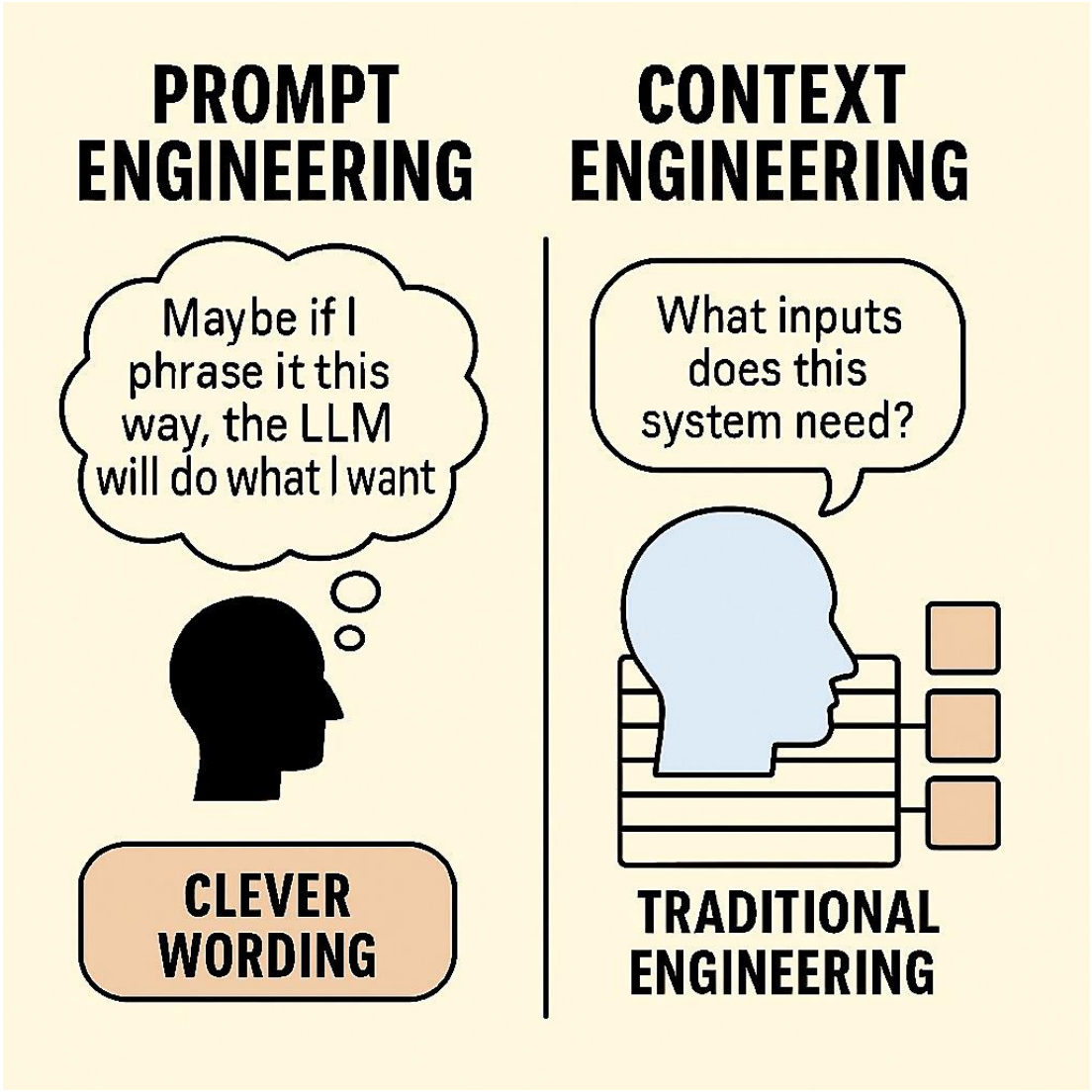


Figure 2: The basic tool use paradigm. LM calls `check_weather` tool by generating text tokens. This call triggers the server to execute the call and return the output sunny, using which the LM replaces the API call tokens in the response to the user.

Memory

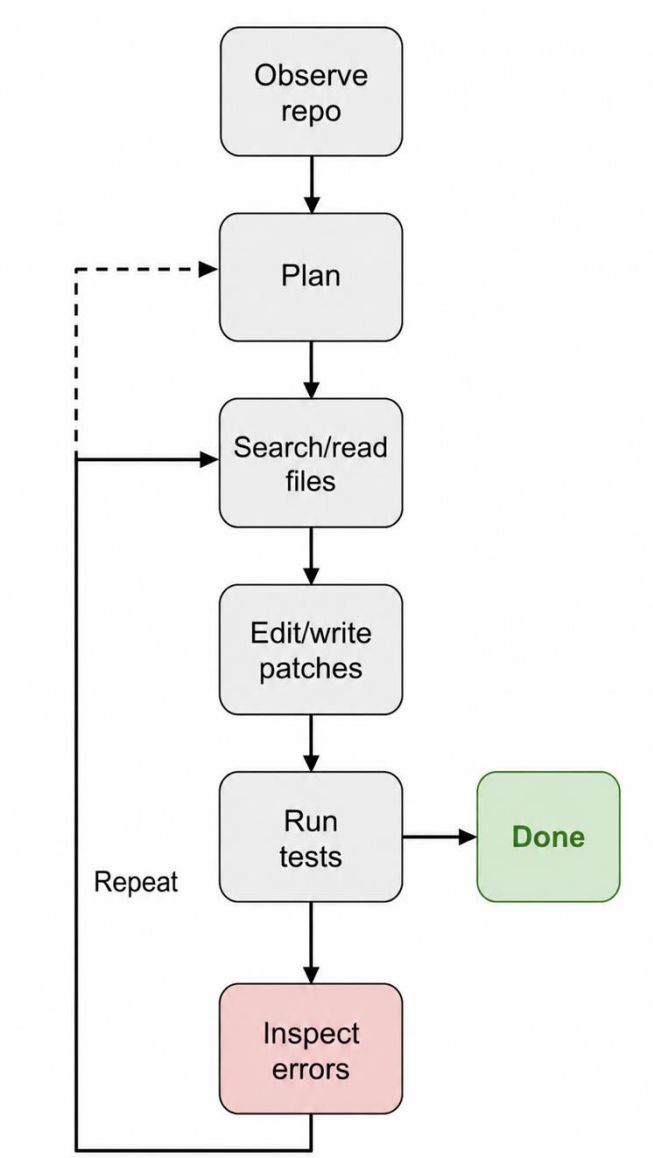
- Short-term memory: All the in-context learning (i.e prompting) as utilizing short-term memory of the model to learn. May include few show examples, previous conversation context.

- Long-term memory: This provides the agent with the capability to retain and recall (infinite) information over extended periods, often by leveraging file system or RAG.



Weng, Lilian. (Jun 2023). “LLM-powered Autonomous Agents”. Lil’Log. https://lilianweng.github.io/posts/2023-06-23-agent/.

LLM agents Examples - Coding Agent



Group	Tool definitions
File system	- File discovery: `glob` , `grep` , `ls` - File read: `read` , `read_many` - File modification: `write` (a whole new file); `edit` (string exact-match replacement); `multi_edit` ; `apply_patch` (applies a structured patch/diff)
Shell execution	Run commands: `bash` , `PowerShell`
IO	`lsp` , git tools like `git_status` , `git_diff` , `git_commit`
External context	MCP tools, Skills
Web search	`web_search` , `web_fetch` , browser tools
Artifacts	Read docs, images; generate HTML, images
Backend processes	Such as: `CronCreate` , `CronDelete` , `CronList`
Agent delegation	Such as: `spawn_agent` , `resume_agent` , `wait_agent` , `list_agents` , `close_agent` , `interrupt_agent` , etc.

Lu, Yuchen, Run Yang, Yichen Zhang, Shuguang Yu, Runpeng Dai, Ziwei Wang, Jiayi Xiang et al. "Stateval: A comprehensive benchmark for large language models in statistics." arXiv preprint arXiv:2510.09517 (2025).

LLM agents Examples – Auto Research

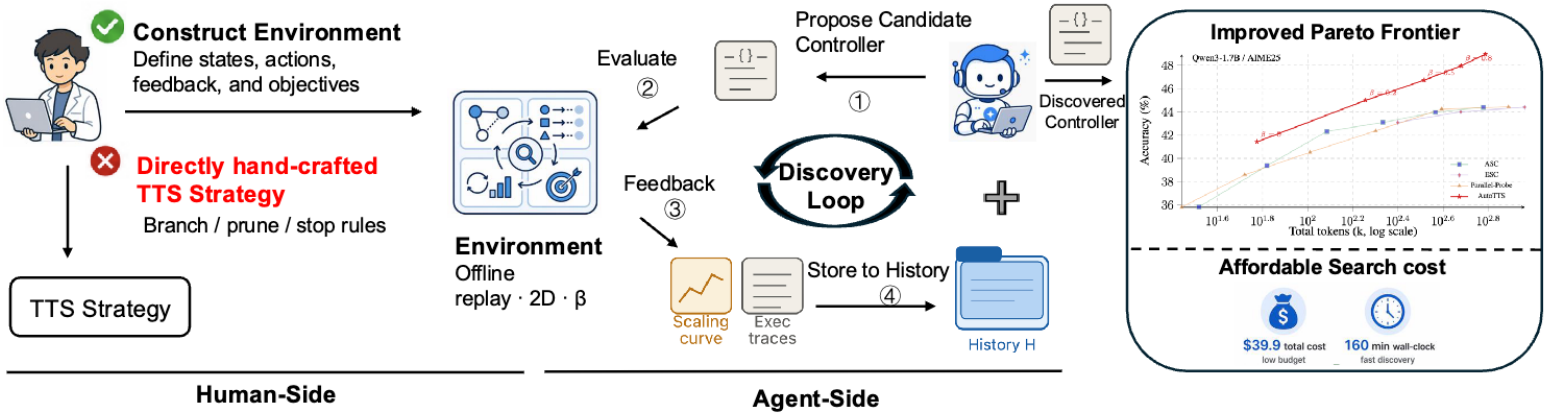
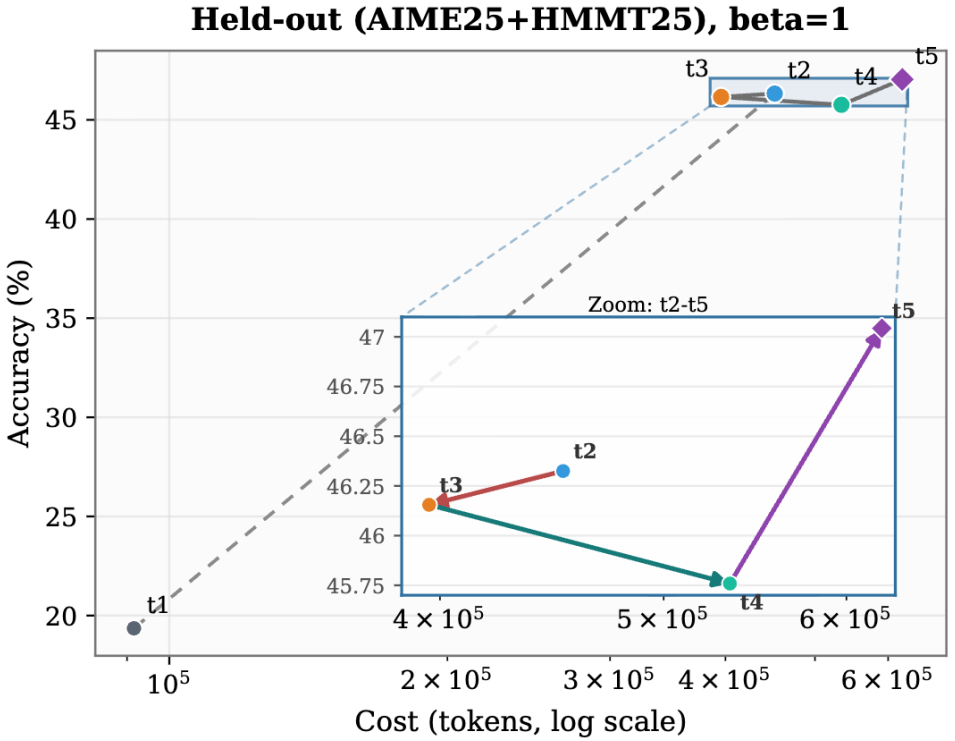
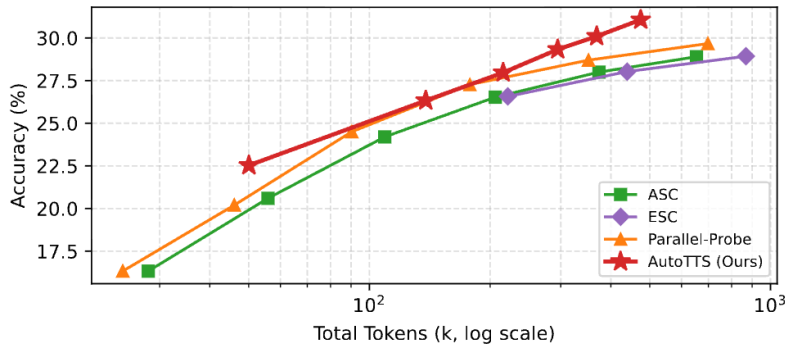


Figure 1: **Overview of our Auto-TTS framework.** Unlike the traditional workflow of manually designing TTS strategies, Auto-TTS shifts the human role from directly hand-crafting branching, pruning, and stopping heuristics to constructing environments by defining states, actions, feedback, and objectives. Given the constructed environment, an explorer LLM iteratively proposes candidate controllers, evaluates them in the offline replay environment, receives feedback from scaling curves and execution traces, and uses the accumulated history to refine future proposals. The right panel shows an example evaluation on Qwen-1.7B and AIME25, where the discovered controller improves the accuracy–cost Pareto frontier over hand-crafted baselines with an affordable one-time search cost.





(a) Qwen3-0.6B on AIME25 (held-out)

Zheng, Tong, et al. "LLMs Improving LLMs: Agentic Discovery for Test-Time Scaling." *arXiv preprint arXiv:2605.08083* (2026).

LLM agents Examples – Auto Research

First Steps Toward Automated AI Research

Early results from Recursive's automated AI research system on model training and GPU kernel benchmarks

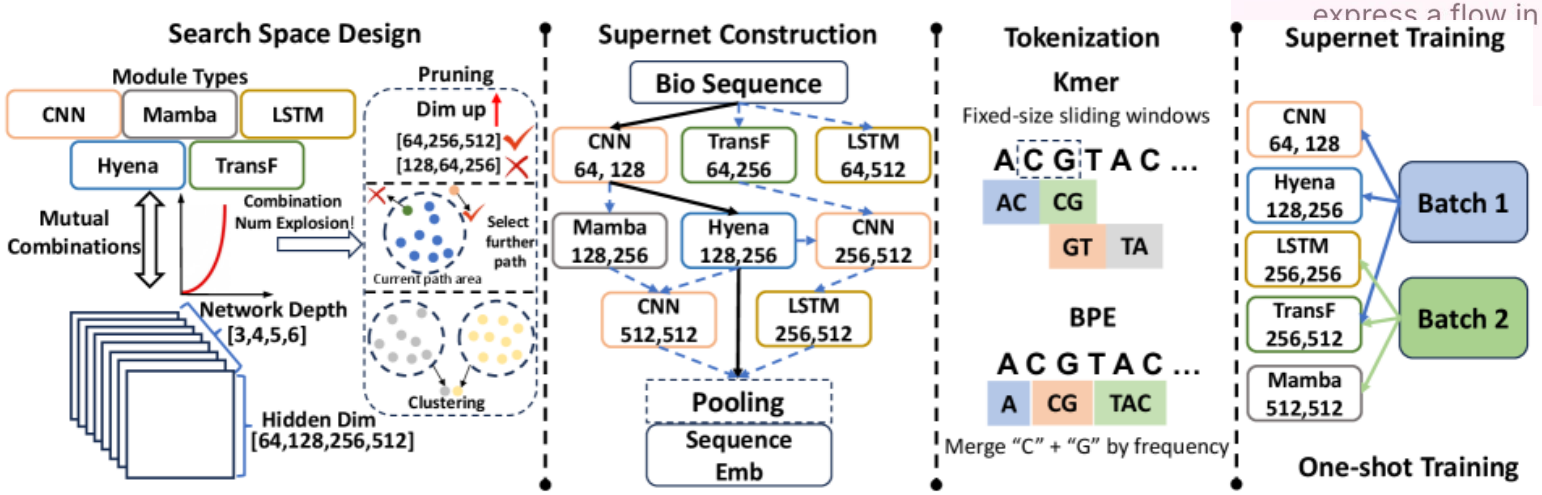
JUNE 11, 2026

AGENT-NATIVE SPARSE ATTENTION

Vortex

Programmable Sparse Attention for Agents as Algorithm Designers

Vortex turns sparse-attention algorithm design into something **AI agents can do** — express a flow in a few lines of Python, and Vortex deploys and benchmarks it inside a real LLM serving stack in minutes.



LLM agents Examples - STATEval

```
Every morning Aya goes for a $9$-kilometer-long
walk and stops at a coffee shop afterwards. When
she walks at a constant speed of $s$ kilometers
per hour, the walk takes her 4 hours, including
$t$ minutes spent in the coffee shop. When she
walks $s+2$ kilometers per hour, the walk takes
her 2 hours and 24 minutes, including $t$ minutes
spent in the coffee shop. Suppose Aya walks at
$s+\frac{1}{2}$ kilometers per hour. Find the
number of minutes the walk takes her, including
the $t$ minutes spent in the coffee shop.
```

- Large language models excel at mathematical reasoning, and a wide range of math-specific benchmarks have been developed to evaluate this ability.
- How can we curate knowledge from the statistical literature, construct benchmarks, and validate whether LLMs can handle statistical problems?

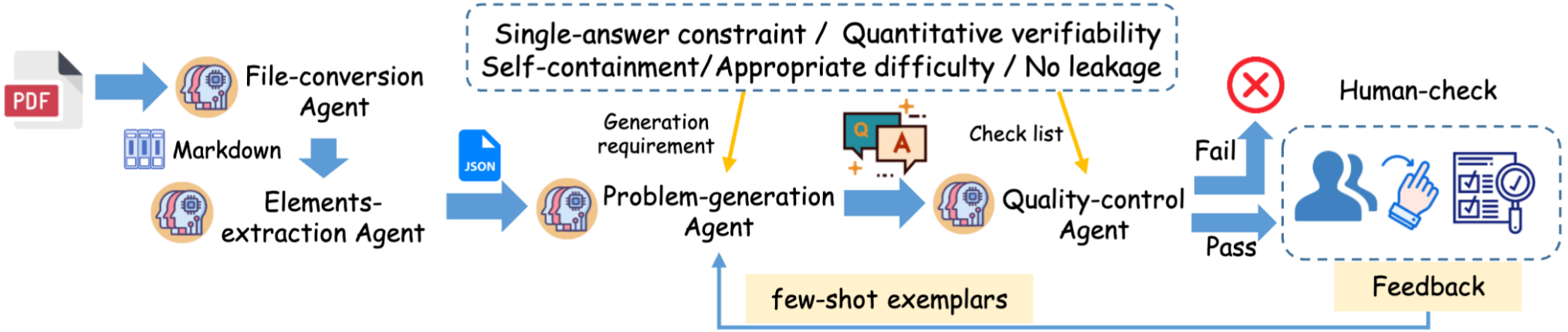
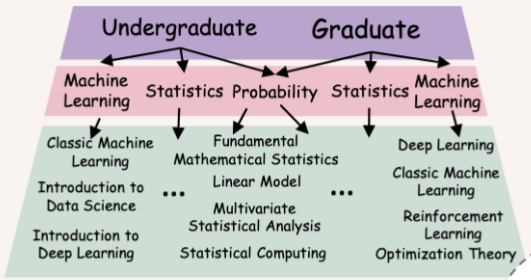


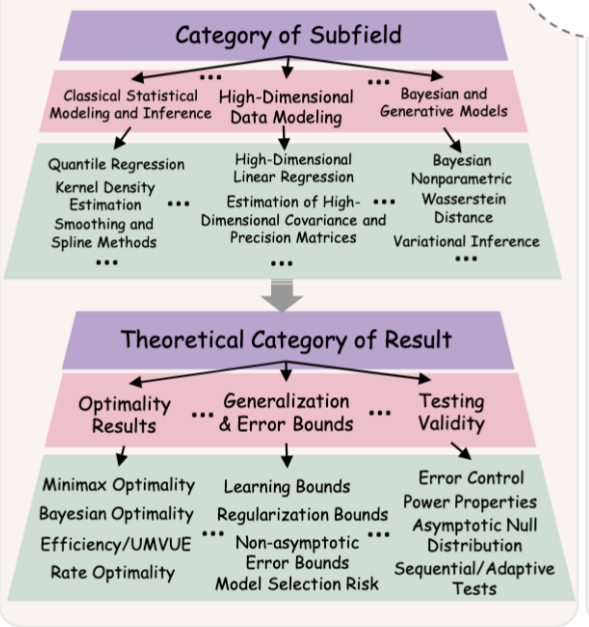
Figure 4: Overview of the StatEval data processing pipeline. Each agent corresponds to a major functional stage in the automated extraction and verification process.

Lu, Yuchen, Run Yang, Yichen Zhang, Shuguang Yu, Runpeng Dai, Ziwei Wang, Jiayi Xiang et al. "Stateval: A comprehensive benchmark for large language models in statistics." arXiv preprint arXiv:2510.09517 (2025).

Foundational Knowledge Dataset



Advanced Statistical Research Dataset



StatEval

Example: Calculation

Question

In a batch of resistors, 10% are defective. If 10 resistors are randomly selected, what is the probability that exactly 2 are defective?
(1) Using the binomial distribution;
(2) Using the Poisson distribution as an approximation to the binomial distribution.

Output(gpt-5-mini) Given n = 10, p = 0.1, we want P(X = 2).

(1) Binomial distribution: $P(X=2) = \binom{10}{2} p^2 (1-p)^{10-2} \approx 0.1937$

(2) Poisson approximation ($\lambda = np = 10 \times 0.1 = 1$):

$$P(X=2) \approx e^{-\lambda}\lambda^2/2! = (e^{-1})/2 \approx 0.1839$$

Evaluation Thought 1 Process 1 Final Answer 1

Final Score $1 \times 0.4 + 1 \times 0.3 + 1 \times 0.3 = 1$

Example: Asymptotic Properties

Question

Consider i.i.d. observations for $i = 1, \ldots, N$, where for each individual i we observe covariates $x_i \in \mathbb{R}^p$ (with x_i taking values in a compact set χ) and, for $j = 1, \ldots, J$ and $t = 1, \ldots, T$, outcomes y_{ijt} that (when observed) follow an exponential-family distribution with density/mass

$$f(y_{ijt}|\gamma_{jt}, a_j, \theta_i, \beta_j, x_i, \phi_j) = \exp(\Phi_j^{-1}\{y_{ijt} \ldots\} + c_j(y_{ijt}, \phi_j))$$

Under the model and the assumptions above, state the asymptotic distribution of the estimator $\hat{\beta}_j$(the p-vector of regression coefficients for item j) in the large-sample limit $N, J \to \infty$ (with $N \asymp J$).

Lemmas

Lemma 1: Under the assumptions, the empirical score and Hessian converge uniformly to their population counterparts:

$$\sup_{\Xi \in \mathcal{H}^\kappa} \left\| \frac{1}{N} \nabla_{u_j} l(\Xi) - \mathrm{E}\left[\nabla_{u_j} l(\Xi)\right] \right\| = o_P(1), \sup_{\Xi \in \mathcal{H}^\kappa} \left\| \frac{1}{N} \nabla^2_{u_j} l(\Xi) - \mathrm{E}\left[\nabla^2_{u_j} l(\Xi)\right] \right\| = o_P(1)$$

The population Hessian limit equals the nonsingular matrix Φ_j

Lemma 2: The constrained maximizer $\hat{\Xi}$ is consistent with the following rates:

$$\|\hat{\beta}_j - \beta_j^*\| = o_P(1), \frac{1}{\sqrt{J}} \|\hat{B}^T - B^{*T}\|_F = O_p(min\{\sqrt{N}, \sqrt{J}\}^{-1})$$

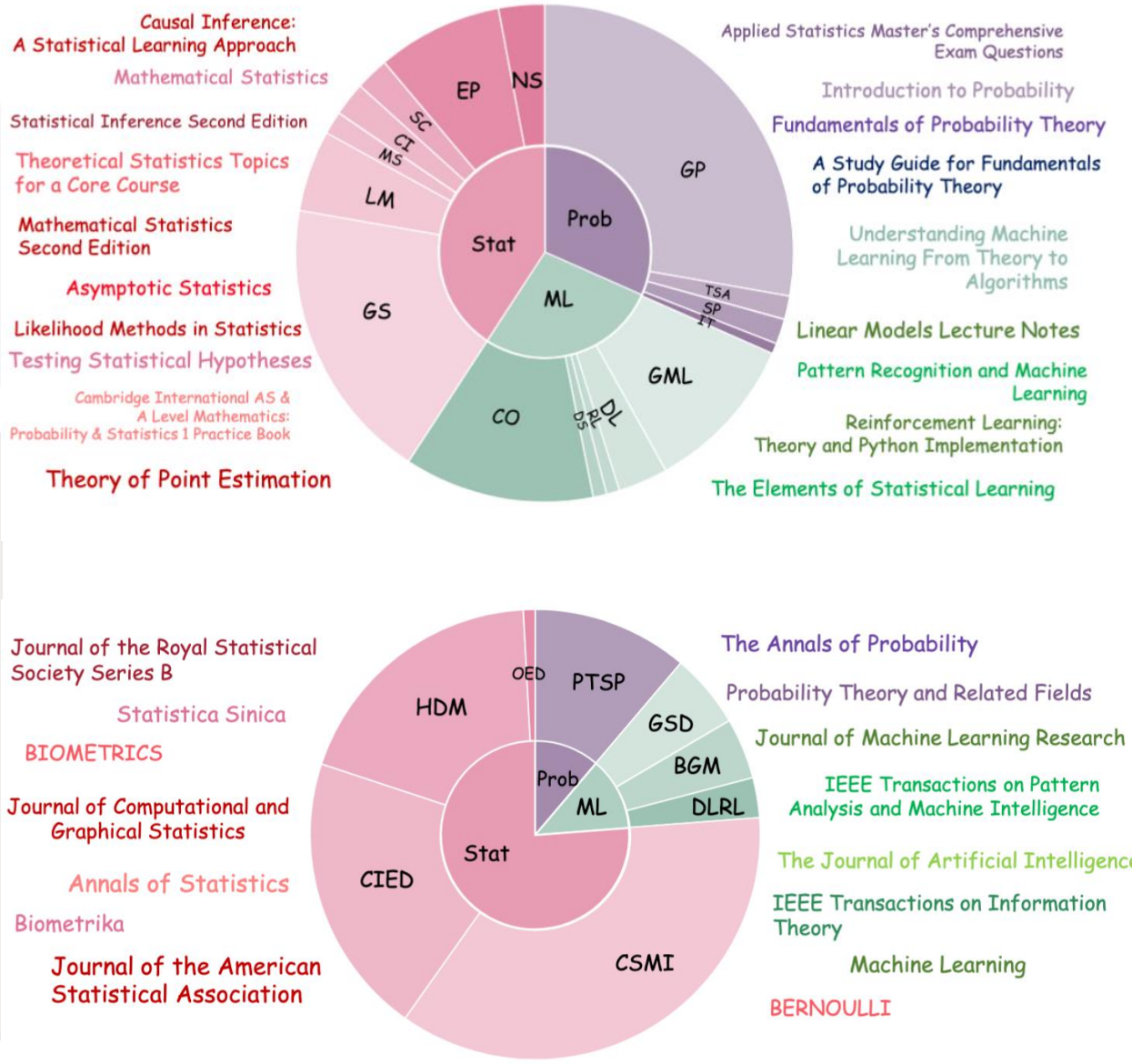
Lemma 3: The estimator admits the expansion:

$$\sqrt{N}(\hat{\beta}_j - \beta_j^*) = -(\Phi_j^{-1})_{(T+1):(T+p),:} \frac{1}{\sqrt{N}} \sum_{i=1}^{N} \sum_{t=1}^{T} r_{it} \phi_j^{-1} b_j''(\cdot)\, e^*_{it,(T+1):(T+p)} + o_P(1)$$

Output(gpt-5-mini) $\sqrt{N}(\hat{\beta}_j - \beta_j^*) \xrightarrow{d} \mathrm{N}(0, \Sigma_{\varepsilon,j})$ as $N, J \to \infty$ with $N \asymp J$

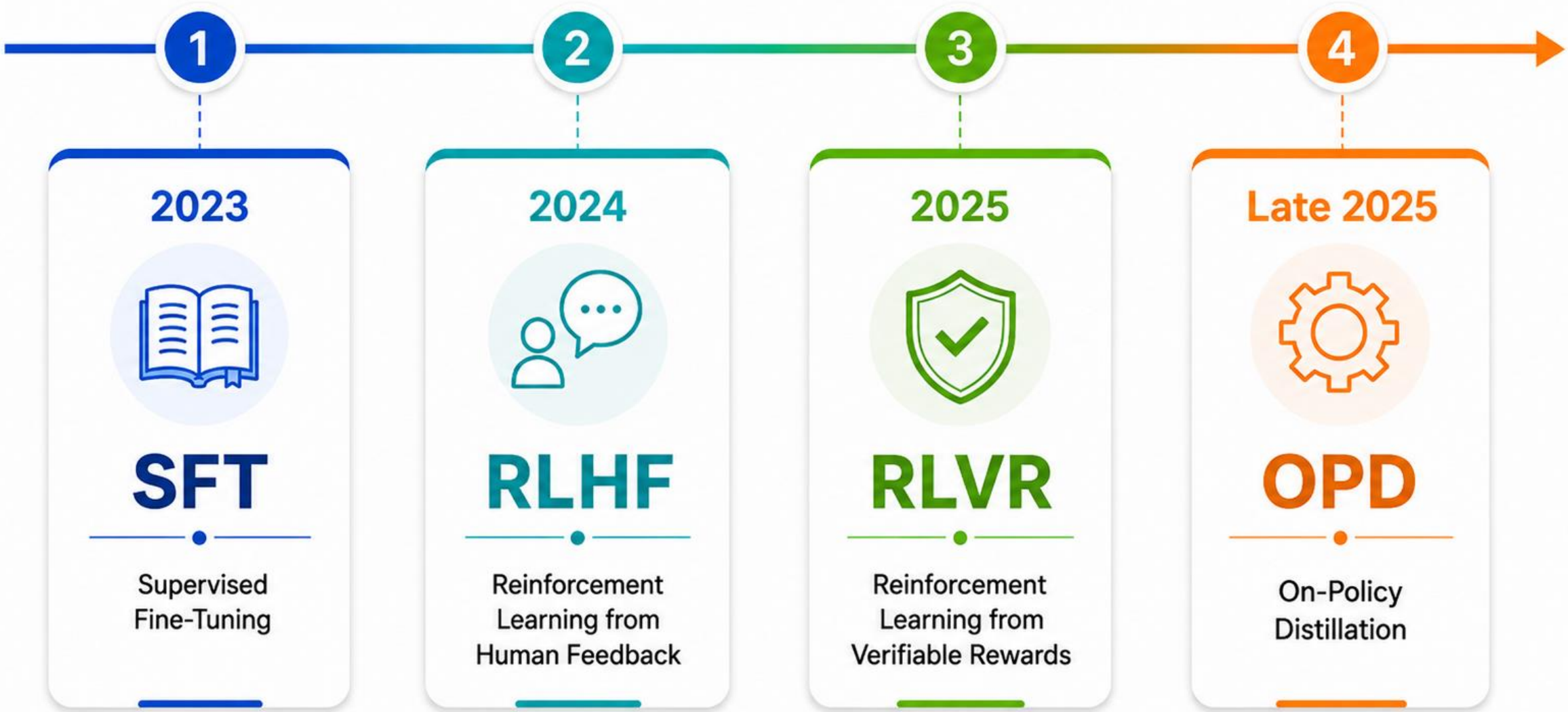
$$\Sigma_{\varepsilon,j} = (-\Phi_j^{-1})_{(T+1):(T+p),(T+1):(T+p)}$$

Answer $\sqrt{N}(\hat{\beta}_j - \beta_j^*) \xrightarrow{d} \mathrm{N}(0, (-\Phi_j^{-1})_{(T+1):(T+p),(T+1):(T+p)})$ CORRECT

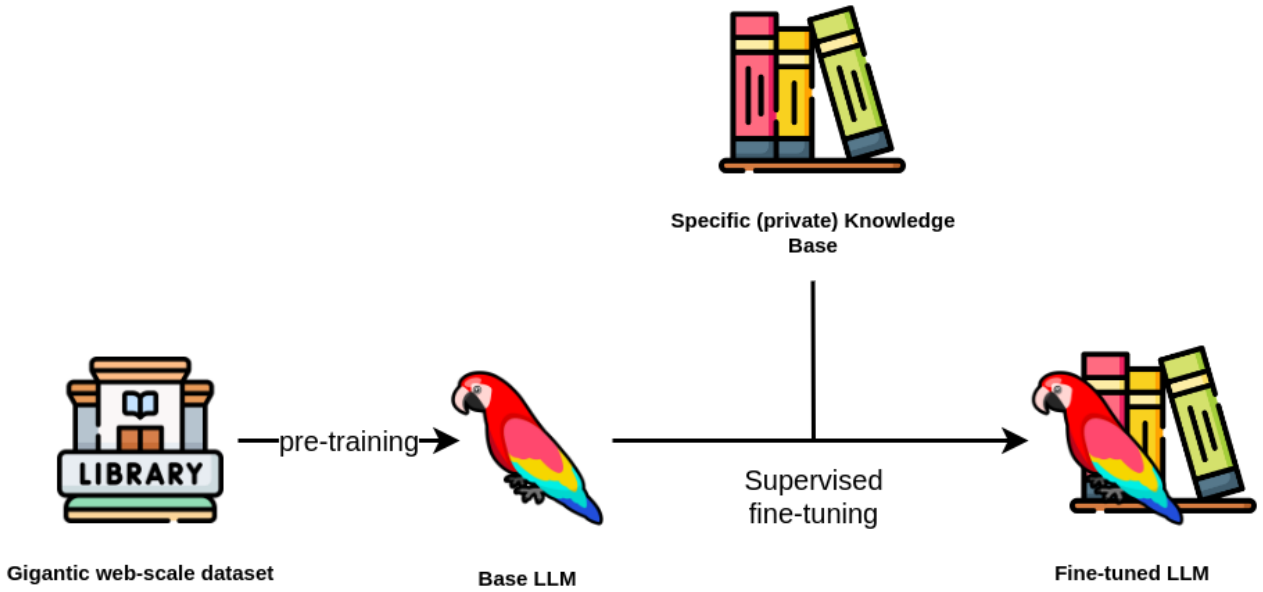


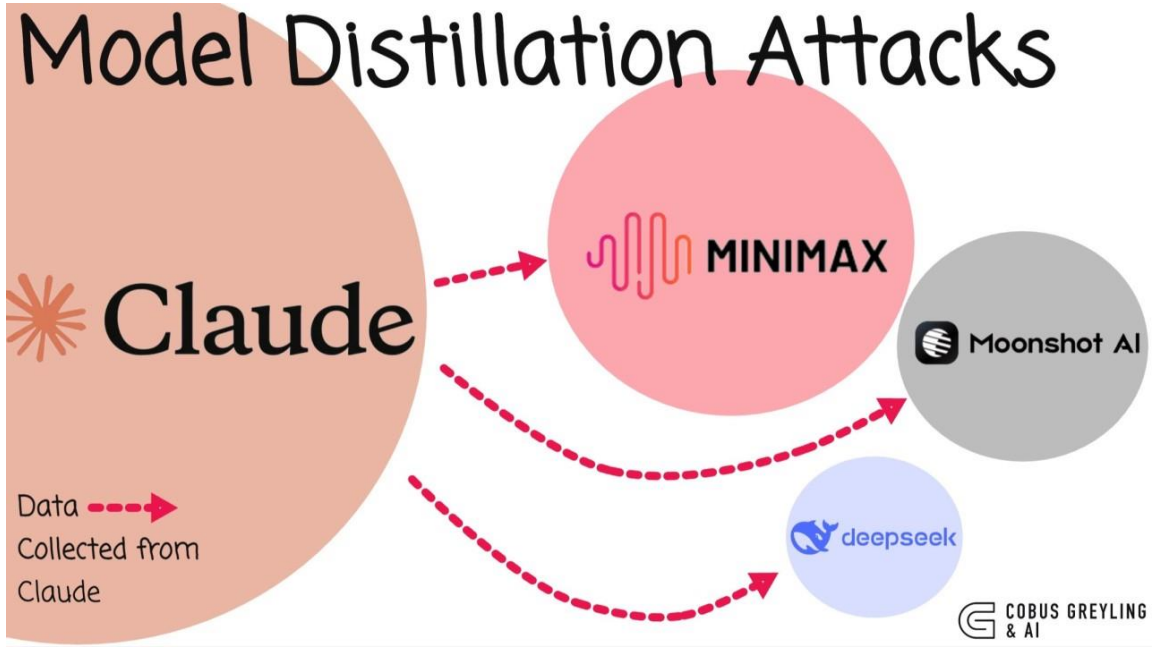
Lu, Yuchen, Run Yang, Yichen Zhang, Shuguang Yu, Runpeng Dai, Ziwei Wang, Jiayi Xiang et al. "Stateval: A comprehensive benchmark for large language models in statistics." arXiv preprint arXiv:2510.09517 (2025).

Training methods



Training methods – SFT





Prompting methods are inherently limited by the capabilities and biases of the base model.

Supervised Fine-Tuning (SFT) is the most basic training approach for aligning large language models with human intentions. It involves fine-tuning a pre-trained model on a dataset of input-output pairs **to follow instructions or perform specific tasks.** The data can be from human or stronger models.

System_Prompt + <User>: [User_Input] +<System>: [Response]</s>

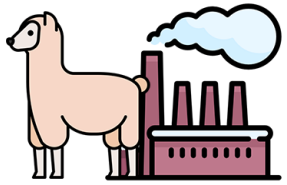
Loss

SFT – Example

Among all the training methods, SFT is the most direct one. There have been many works use SFT at the early stage.

Input (Unstructured Note)	Output (Structured JSON)

"Pt presents for adj appt. Stepped up to 019x025 AW. OB is 4mm, M/R is full-cusp cl-2 on RHS. Placed 018 SS w/ ACOS. Pt reports wearing HG 14hrs/night. Pt & mother understood and compliant."	`"wire_size": "019x025",`
	`"overbite": "4mm",`
	`"molar_class_R": "full-cusp cl-II",`
	`"archwire": "018 SS",`
	`"appliance": "ACOS",`
	`"HG_wear": "14hrs/night",`
	`"compliance": "True"`



LLaMA-Factory

Easy and Efficient LLM Fine-Tuning

Transform unstructured orthodontic progress notes into high-quality structured data. Outperforming commercial frontier model while preserving patient privacy.

Easily fine-tune 100+ large language models with Web UI

Key Limitation: Very limited generalization

Dumont P, et al. High-Fidelity Clinical Information Extraction via Full-Parameter Fine-Tuned LLMs. In: Proceedings of the 2026 IEEE Engineering in Medicine and Biology Conference (EMBC)

Alignment

RLHF allows models to learn from *comparative feedback*—for example, choosing one response over another—so that the model can generalize better to unseen situations and generate outputs that are more aligned with human preferences.

3H principles:

- **Helpful** (Improves usability)
- **Honest** (Enhances trustworthiness)
- **Harmless** (Mitigates harmful outputs)

Alignment

Challenges

1. Human in the loop is expensive – *Reward Modelling*

2. Human judgments are noisy and miscalibrated. – *Preference learning*

Score the helpfulness of the following response, 1-10

What are the steps for making a simple cake?

1. *Preheat oven to 350°F (175°C).*
2. *Grease and flour a cake pan.*
3. *In a bowl, combine 2 cups flour, 1.5 cups sugar, 3.5 tsp baking powder, and a pinch of salt.*
4. *Add 1/2 cup butter, 1 cup milk, and 2 tsp vanilla; mix well.*
5. *Beat in 3 eggs, one at a time.*
6. *Pour batter into the pan.*
7. *Bake for 30-35 minutes or until a toothpick comes out clean.*
8. *Let cool, then frost or serve as desired.*

Alignment – Bradley Terry model

Which of these two responses is more helpful?

What are the steps for making a simple cake?

1. *Preheat oven to 350°F (175°C).*
2. *Grease and flour a cake pan.*
3. *In a bowl, combine 2 cups flour, 1.5 cups sugar, 3.5 tsp baking powder, and a pinch of salt.*
4. *Add 1/2 cup butter, 1 cup milk, and 2 tsp vanilla; mix well.*
5. *Beat in 3 eggs, one at a time.*
6. *Pour batter into the pan.*
7. *Bake for 30-35 minutes or until a toothpick comes out clean.*
8. *Let cool, then frost or serve as desired.*

What are the steps for making a simple cake?

1. *Warm up the oven.*
2. *Grease a cake pan.*
3. *Blend dry ingredients in a bowl.*
4. *Incorporate butter, milk, and vanilla.*
5. *Mix in the eggs.*
6. *Pour into the prepared pan.*
7. *Bake until golden brown.*
8. *Add frosting if desired.*

Instead of asking labelers to assign scores, we collect their preferences through pairwise comparisons.

$$D = \{x^i, y_w^i, y_l^i\}$$

Prompt

Preferred Response

Dispreferred Response

Alignment - RLHF

How do we get feedback for the reward while training our RL model?

$$p(y_w > y_l|x) = \sigma(r(x, y_w) - r(x, y_l))$$

Logistic function; which is equivalent to using softmax:

$$\frac{1}{1+e^{-x}}$$

$$p(y_w > y_l|x) = \frac{\exp(r(x, y_w))}{\exp(r(x, y_w)) + \exp(r(x, y_l))}$$

Train a Reward Model (RM) on preference data to predict preferences!

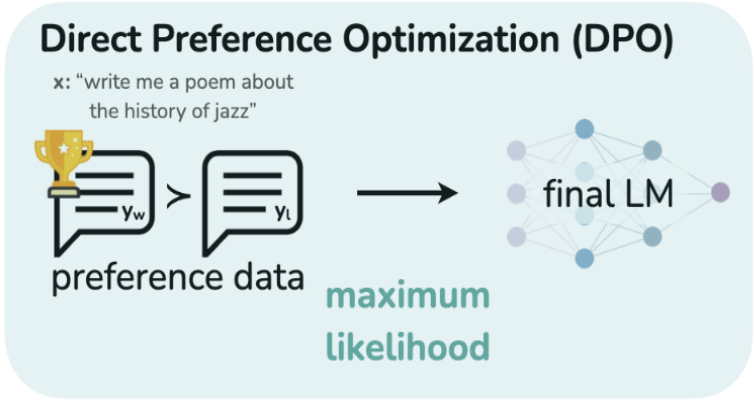
$$\mathcal{L}_R(\phi, D) = -\mathbb{E}_{(x,y_w,y_l)\sim D}[\log \sigma(r(x, y_w) - r(x, y_l))]$$

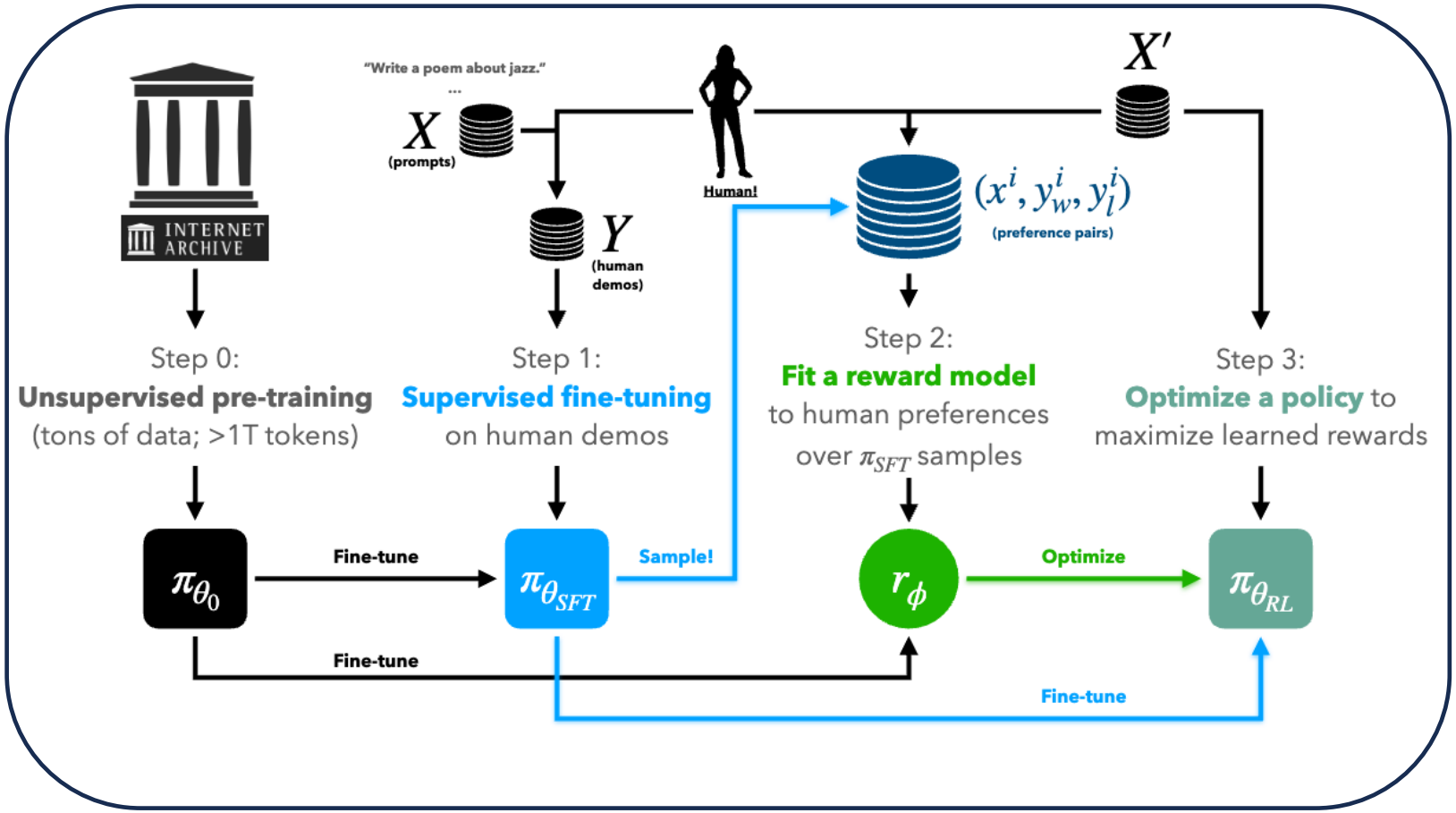
Use RL algorithm to train optimize the LLM π

$$\max_{\pi_\theta} \mathbb{E}_{x\sim D, y\sim\pi_\theta(y|x)}[r_\phi(x, y)]$$

Alignment - Improvements

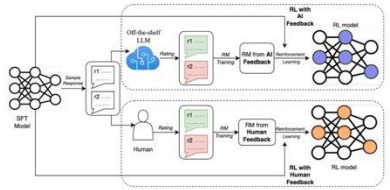
- The reward model is hard to train and large.



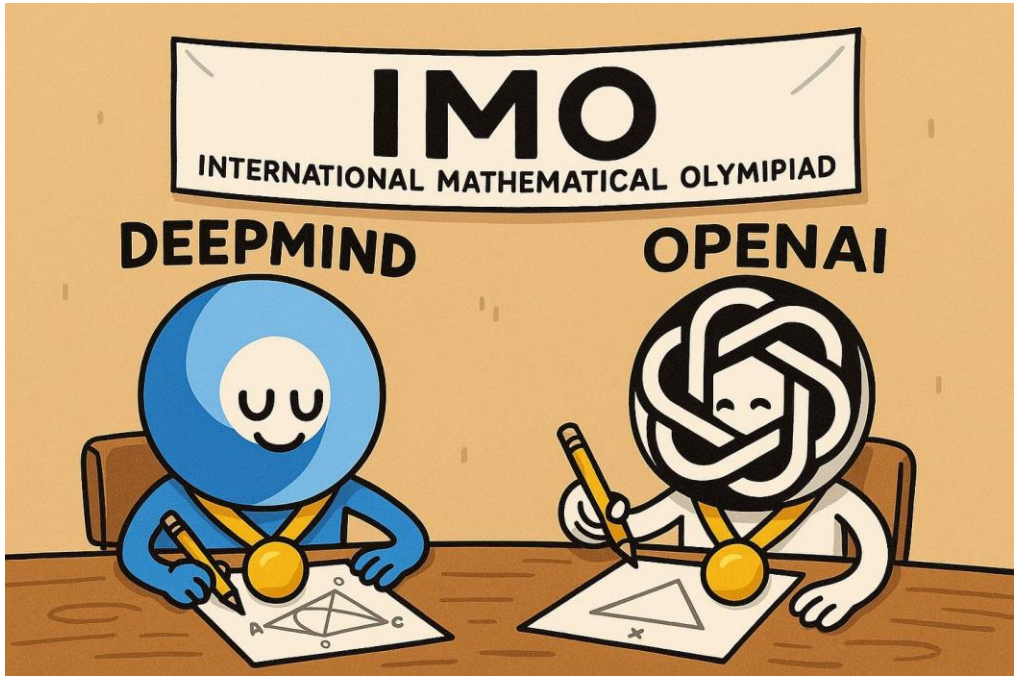


- Human labeling remains costly.

RLAIF - RL from AI Feedback for LLM



Learning to Reasoning with LLMs



August 1, 2026 Publication

Ten advances in mathematics and theoretical computer science

Read the paper ↗ Read the reasoning walkthroughs ↗

The secret behind the success - Reasoning

*We achieved this year's result using an advanced version of Gemini Deep Think – an enhanced **reasoning mode** for complex problems that incorporates some of our latest research techniques, including parallel thinking. This setup enables the model to simultaneously explore and combine multiple possible solutions before giving a final answer, rather than pursuing a single, linear chain of thought.*

deepseek

DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning

Learning to Reasoning with LLMs

What is reasoning?

Reasoning refers to the process of thinking through a problem or situation in order to form a logical conclusion or make a decision. It involves using evidence, facts, and logic to arrive at a solution or answer.

Why Reasoning is important?

Even for humans, problems like math or complex question answering are difficult to solve with direct answers alone, as they often require multi-step reasoning or integrating diverse pieces of information.

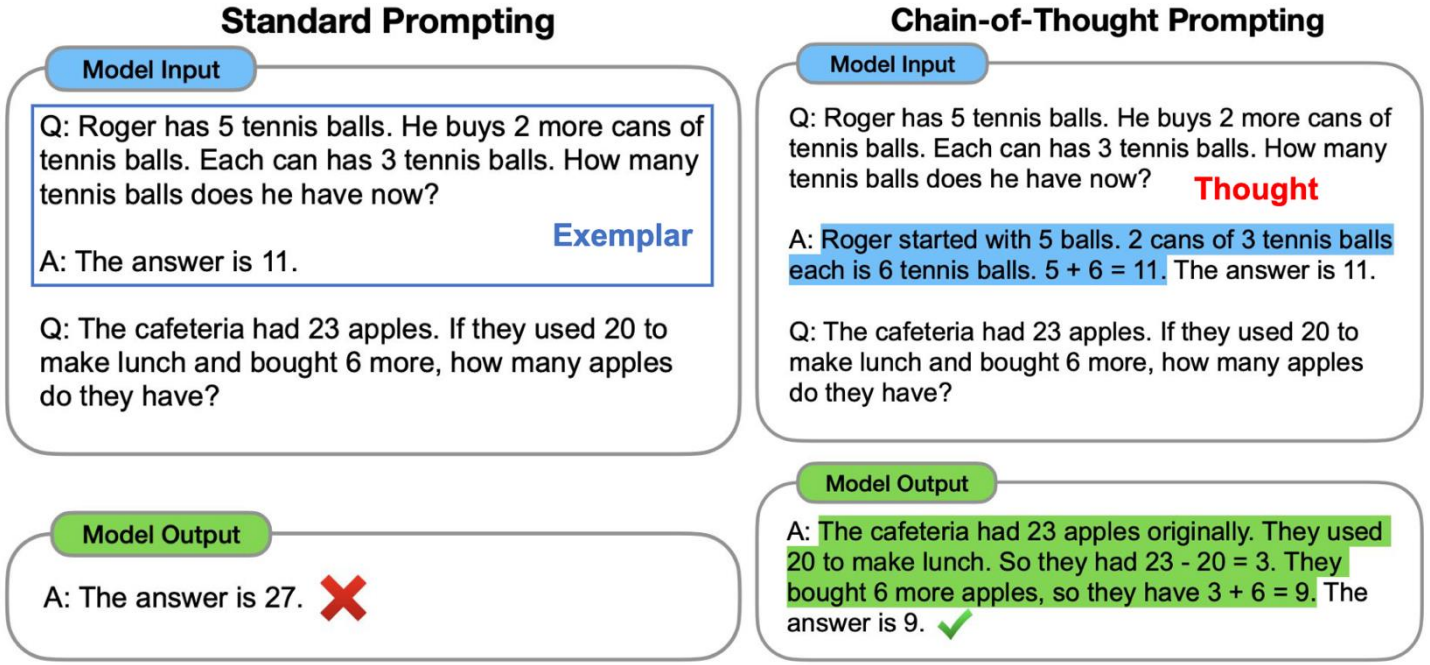
How can LLM Reasoning?

Generate sequence of tokens representing intermediate steps in the reasoning process.

Problem 1:
Question: Two trains running in opposite directions cross a man standing on the platform in 27 seconds and 17 seconds respectively and they cross each other in 23 seconds. The ratio of their speeds is:
Options: A) 3/7 B) 3/2 C) 3/88 D) 3/8 E) 2/2
Rationale: Let the speeds of the two trains be x m/sec and y m/sec respectively. Then, length of the first train = 27x meters, and length of the second train = 17 y meters. $(27x + 17y) / (x + y) = 23 \rightarrow 27x + 17y = 23x + 23y \rightarrow 4x = 6y \rightarrow x/y = 3/2$.
Correct Option: B

Learning to Reasoning with LLMs

How can we guide LLMs to perform reasoning?



Or some other prompting methods:

1. Least-to-Most Prompting
2. Self reflection/ verification

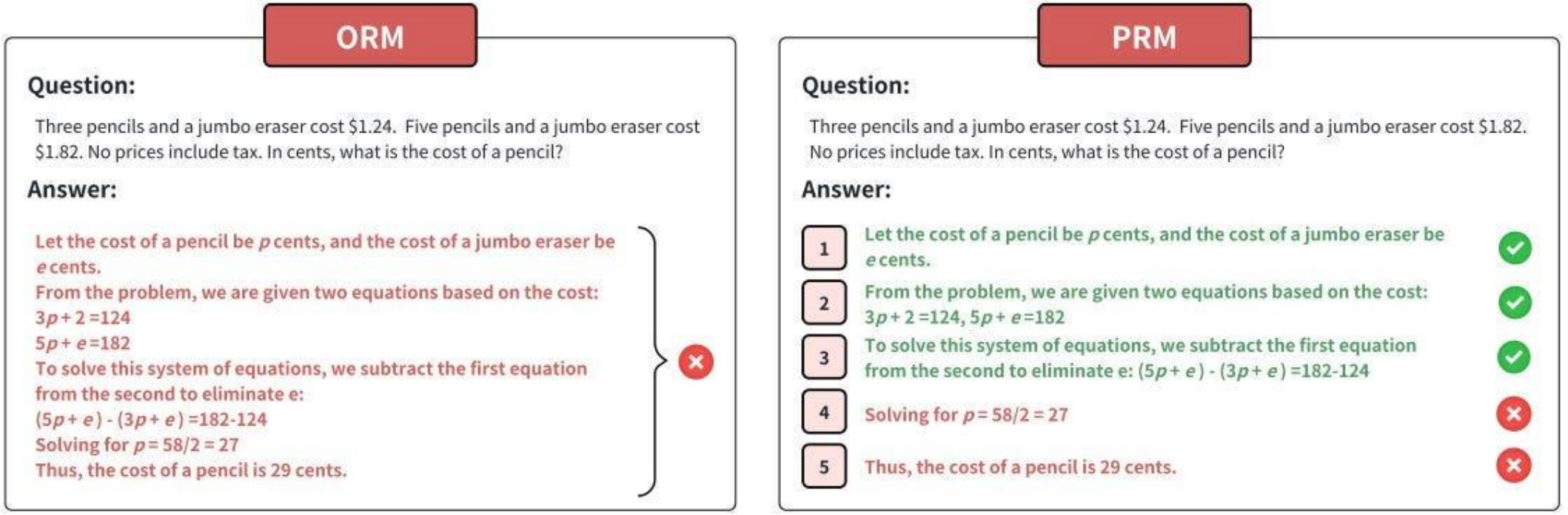
Can we go beyond prompting?

"We have found that the performance of o1 consistently improves with more reinforcement learning (train-time compute) and with more time spent thinking (test-time compute). The constraints on scaling this approach differ substantially from those of LLM pretraining, and we are continuing to investigate them." — OpenAI o1 release blog

Zhou, Denny, et al. "Least-to-most prompting enables complex reasoning in large language models." *arXiv preprint arXiv:2205.10625* (2022).
Weng, Yixuan, et al. "Large language models are better reasoners with self-verification." *arXiv preprint arXiv:2212.09561* (2022).

Learning to Reasoning with LLMs

Reward guided decoding



Lightman, Hunter, et al. "Let's verify step by step." *The Twelfth International Conference on Learning Representations*. 2023.

Learning to Reasoning with LLMs

Reward guided planning

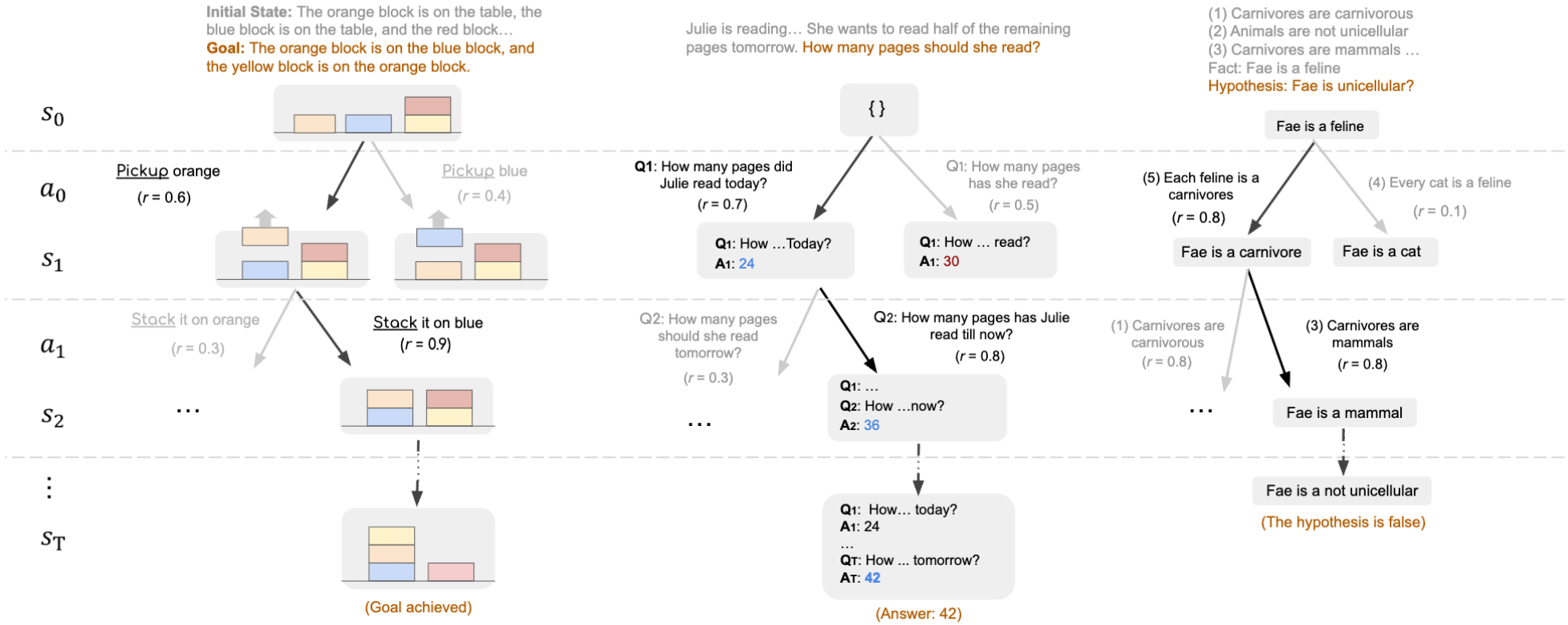


Figure 2: RAP for plan generation in Blocksworld (left), math reasoning in GSM8K (middle), and logical reasoning in PrOntoQA (right).

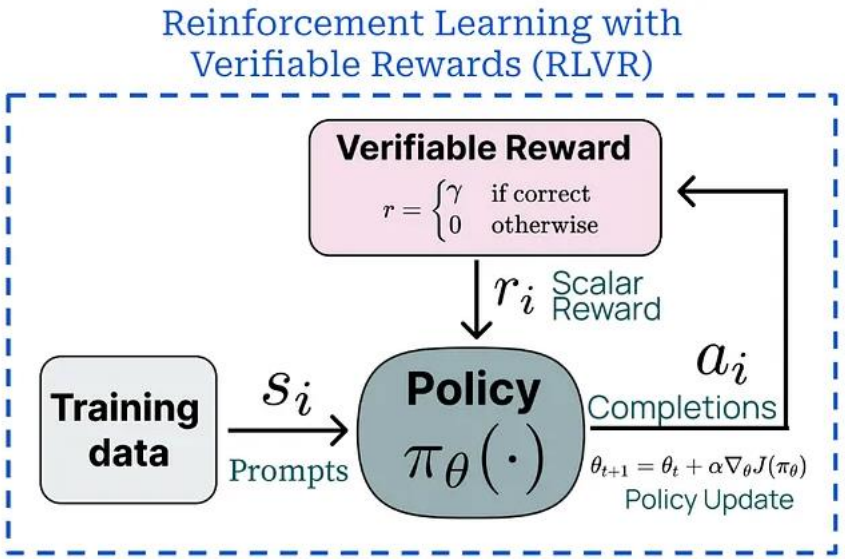
Hao, Shibo, et al. "Reasoning with language model is planning with world model." *arXiv preprint arXiv:2305.14992* (2023).

DeepseekR1 and RLVR

"One of the most remarkable aspects of this self-evolution is the emergence of sophisticated behaviors as the test-time computation increases. Behaviors such as reflection—where the model revisits and reevaluates its previous steps—and the exploration of alternative approaches to problem-solving arise spontaneously. These behaviors are not explicitly programmed but instead emerge as a result of the model's interaction with the reinforcement learning environment." — DeepSeek-R1 'Aha moment'

Reinforcement learning with verifiable reward

$$R(\hat{y}, y) = \begin{cases} 1, & \texttt{is\_equivalent}(\hat{y}, y) \\ -1, & \text{otherwise} \end{cases}$$



Instead of training a separate reward model or explicitly assigning credit to each reasoning step, we allow the model to discover effective reasoning strategies through trial and error, guided by verifiable rewards.

1. Reduce the computational burden of training a separate reward model.
2. Eliminate the risk of reward hacking.

DeepseekR1 and RLVR

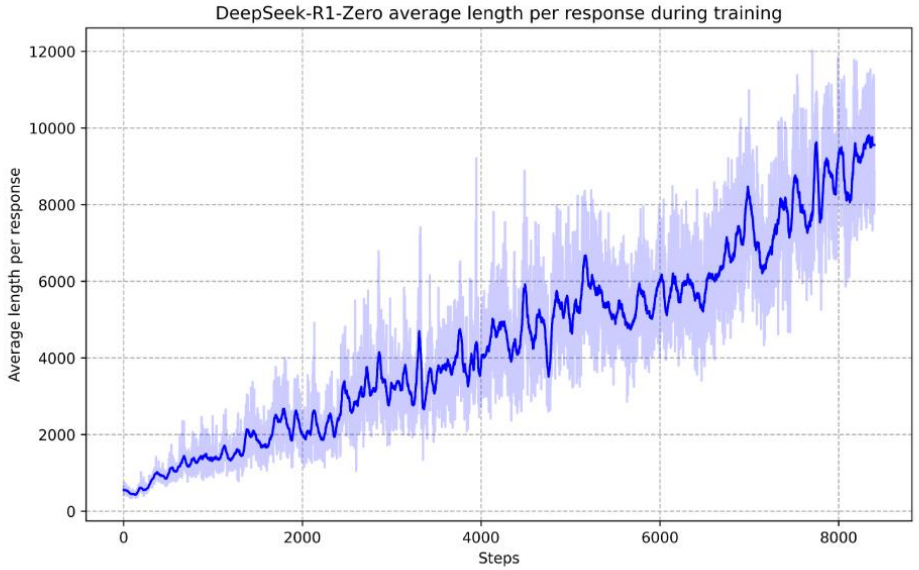


Figure 3 | The average response length of DeepSeek-R1-Zero on the training set during the RL process. DeepSeek-R1-Zero naturally learns to solve reasoning tasks with more thinking time.

Question: If $a > 1$, then the sum of the real solutions of $\sqrt{a-\sqrt{a+x}} = x$ is equal to
Response: <think> To solve the equation $\sqrt{a-\sqrt{a+x}} = x$, let's start by squaring both $\cdots$ $\left(\sqrt{a-\sqrt{a+x}}\right)^2 = x^2 \implies a-\sqrt{a+x} = x^2$. Rearrange to isolate the inner square root term: $(a-x^2)^2 = a+x \implies a^2-2ax^2+(x^2)^2 = a+x \implies x^4-2ax^2-x+(a^2-a) = 0$ $\ldots$ Wait, wait. Wait. That's an aha moment I can flag here. Let's reevaluate this step-by-step to identify if the correct sum can be $\cdots$ We started with the equation: $\sqrt{a-\sqrt{a+x}} = x$ First, let's square both sides: $a-\sqrt{a+x} = x^2 \implies \sqrt{a+x} = a-x^2$ Next, I could square both sides again, treating the equation: $\cdots$ $\ldots$

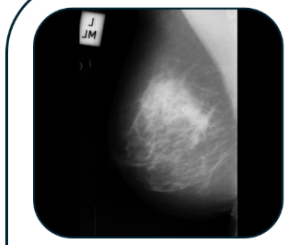
Table 3 | An interesting "aha moment" of an intermediate version of DeepSeek-R1-Zero. The model learns to rethink using an anthropomorphic tone. This is also an aha moment for us, allowing us to witness the power and beauty of reinforcement learning.

During the training process, they observe the increase in reasoning length and "aha moment" where the model learns to self-reflection and verify.

Some of the Challenges

- **Ensuring Faithfulness of the Reasoning Process**
- **Reducing Reasoning Latency**
- **Handling General Reasoning Tasks with Unverifiable Rewards**

RLVR – More Applications



Problem: What does this image depict in terms of its content?

A)Brain tissue
B)Intestinal tissue
C)Kidney tissue
D)Breast tissue

Med-R1 Response:

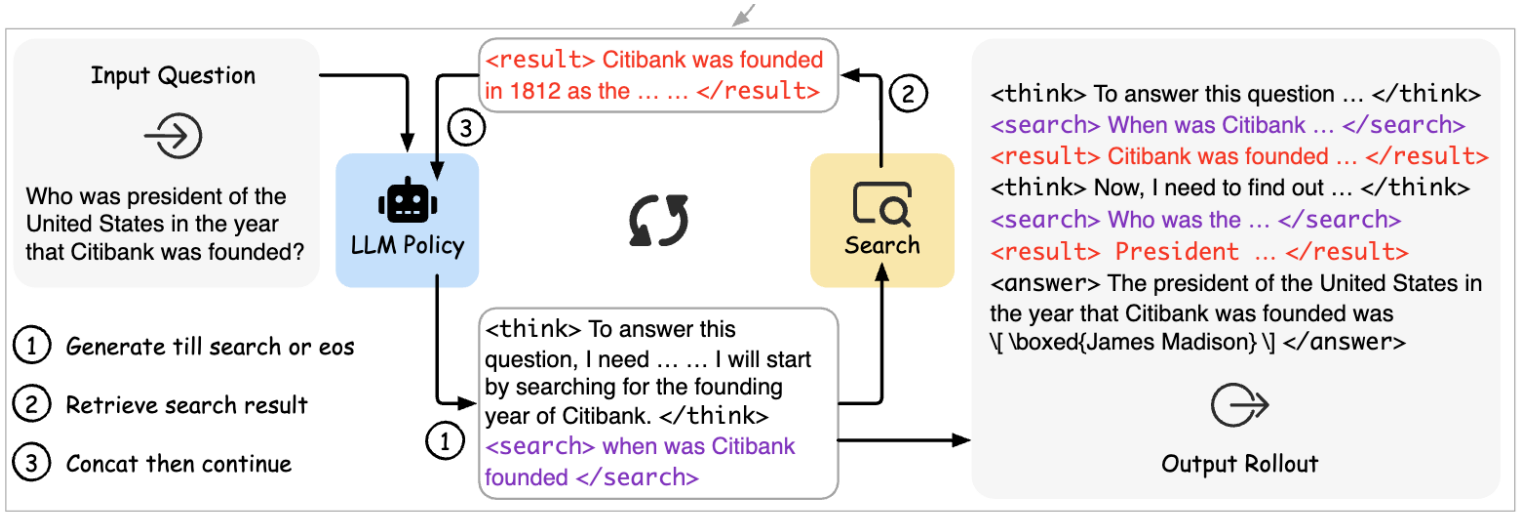
Think: The image appears to be a mammogram, which is a type of X-ray used to detect abnormalities in the breast tissue. The presence of dense, irregular patterns and a central area of lighter density suggest the presence of breast tissue.

Answer: D

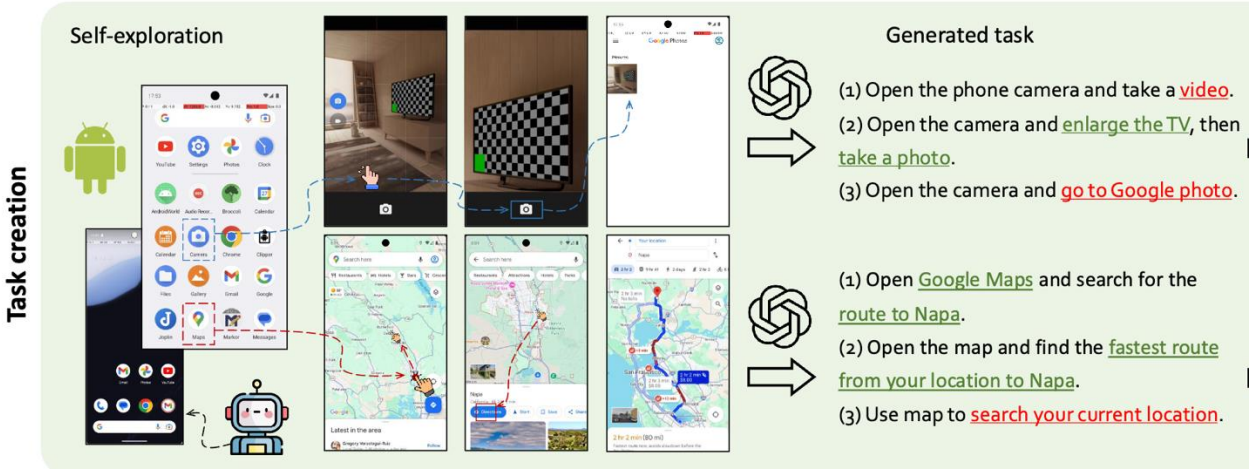
Groundtruth: D

Medical Reasoning

(b)



Use of Search engine

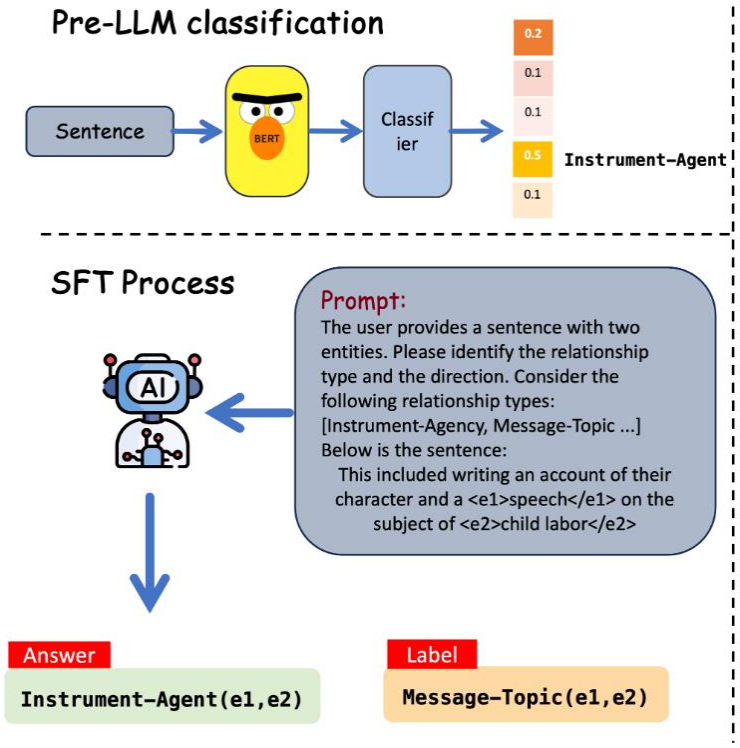


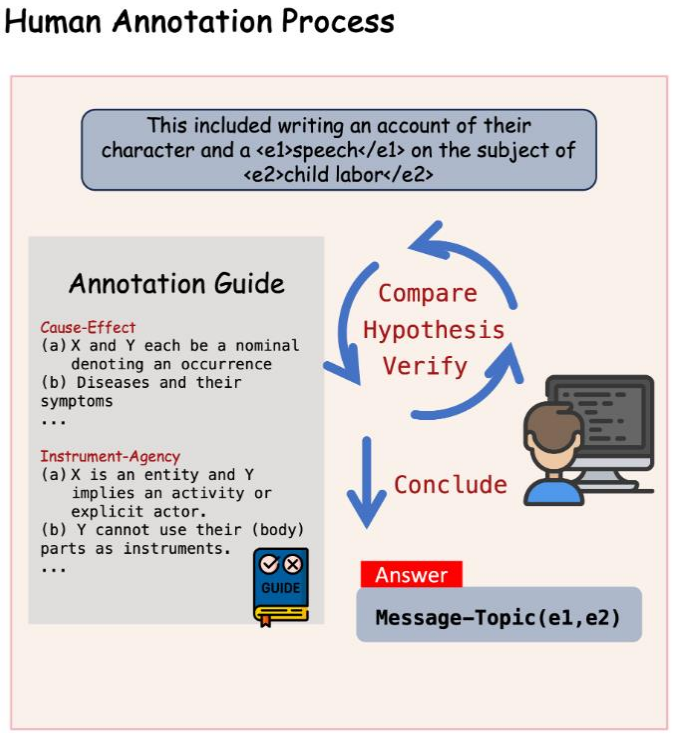
Use of smart phone

RLVR – Example R1-RE

Sentence Olanzapine was also associated with more frequent reports of weight gain and significantly greater VA costs . . .
Label [[Olanzapine:drug, risk-factor-of, weight gain:symptom]]

Relation extraction is a critical step in knowledge graph construction.





Existing method focus on learning a mapping from sentence to triplet, which cannot generalize to OOD.

Human annotators are generalizable across tasks because they can reason on the requirements.

Dai, Runpeng, et al. "R1-RE: Cross-Domain Relation Extraction with RLVR." ACL 2026 (Oral Presentation)

RLVR – Example R1-RE

R1-RE-7B

We need to determine the relationship between <e1>schizophrenia</e1> and <e2>prolactin-increasing antipsychotics</e2>. Let's break this down step by step.

1. **Identify the entities**:
   - <e1>schizophrenia</e1> is the disease or condition.
   - <e2>prolactin-increasing antipsychotics</e2> is a type of medication or treatment.

Breakdown the sentence

2. **Analyze the relationship**:
   - The sentence states that schizophrenia is associated with a specific type of antipsychotics (prolactin-increasing antipsychotics) being used in the treatment or management of schizophrenia.

Compare definitions

3. **Apply the definitions**:
   - **causes**: This wouldn't apply because the sentence suggests the antipsychotics are being used to treat, not cause, schizophrenia.

Hypothesis-validation

- **risk_factor_of**: This would apply if the antipsychotics increased the risk of schizophrenia, but here it's the other way around; the antipsychotics are being used because of schizophrenia, not the other way.

4. **Conclusion**: Draw conclusion

Given that the sentence directly states that antipsychotics are used in the treatment of schizophrenia, this fits squarely into the definition of a treatment relationship. The antipsychotics are specifically used to address or manage schizophrenia, which is the essence of the treatment_for relationship.

Final Answer

`Treatment_for(e2,e1)`

Target sentence

Monitoring prolactinemia and addressing hyperprolactinemia is paramount in women with <e1>schizophrenia</e1> being treated with <e2>prolactin-increasing antipsychotics</e2>.

Qwen-2.5-7B

The sentence mentions that monitoring prolactinemia and addressing hyperprolactinemia is important for women with schizophrenia who are treated with prolactin-increasing antipsychotics. This implies that prolactin-increasing antipsychotics can lead to prolactinemia and hyperprolactinemia, and that these conditions are associated with the treatment.

Given the definitions, this is best classified under the "risk_factor_of" relationship as it indicates that prolactin-increasing antipsychotics (e2) could increase the risk or likelihood of prolactinemia and hyperprolactinemia (e1). This fits into the susceptibility and prevalence category of "risk_factor_of".

Final Answer

`Risk_factor_of(e2,e1)`

Figure 6: Case study comparing the chain-of-thought (COT) reasoning of **R1-RE-7B** and Qwen2.5-7B-Instruct. Due to space constraints, some COT outputs are omitted; the complete COT reasoning process for **R1-RE-7B** is provided in Appendix A.2.

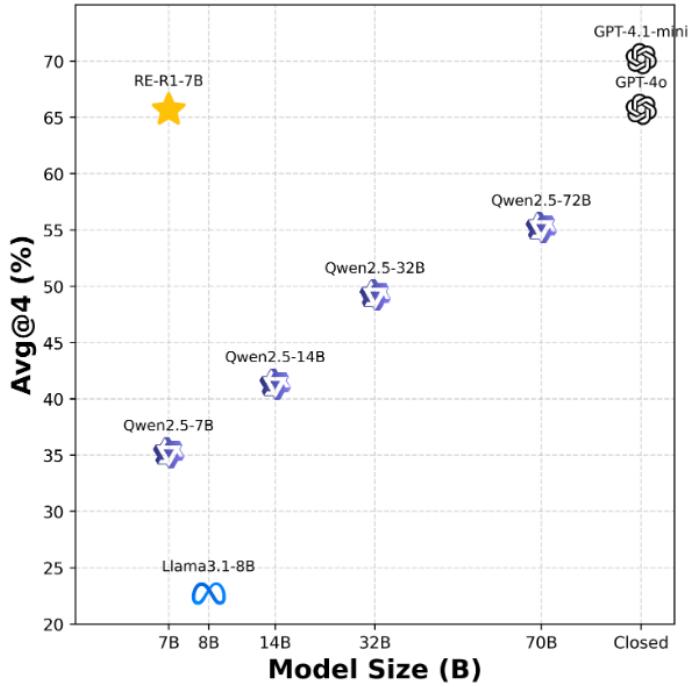
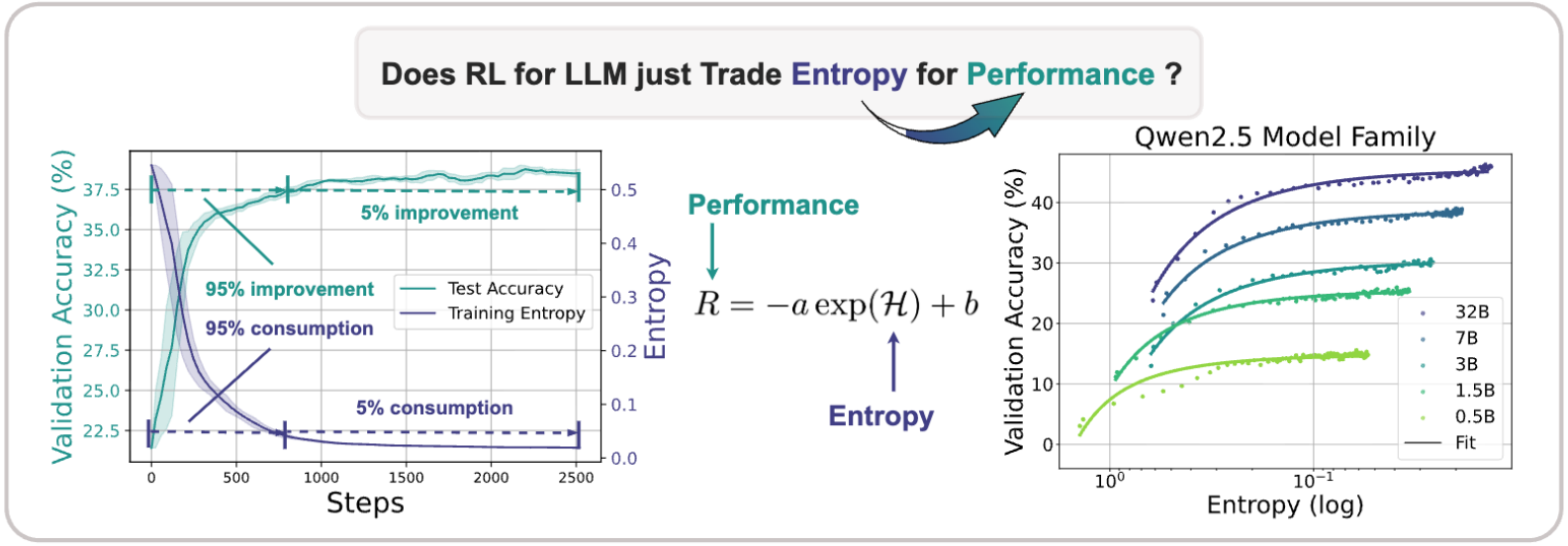


Figure 1: Testing accuracy on the MDKG dataset for R1-RE-7B trained on the Sem-2010 dataset, compared with other models. Detailed results are provided in Table 3.

RLVR – The Entropy Mechanism



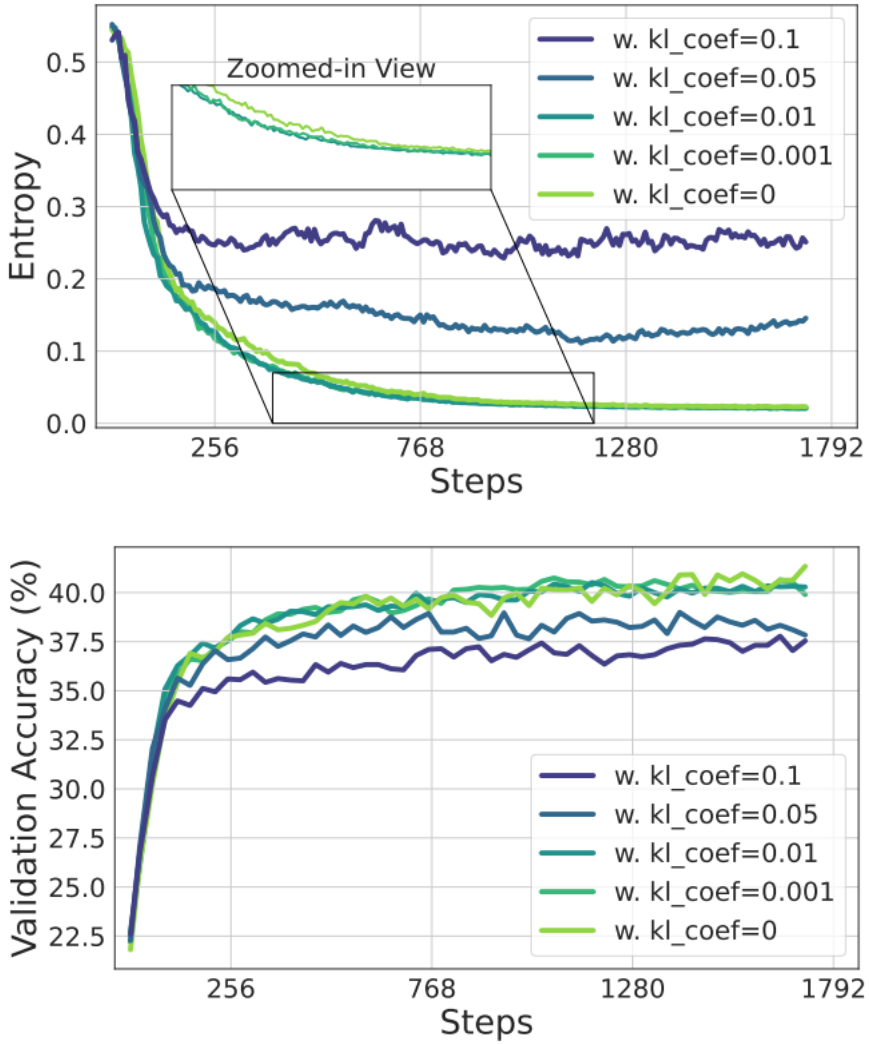
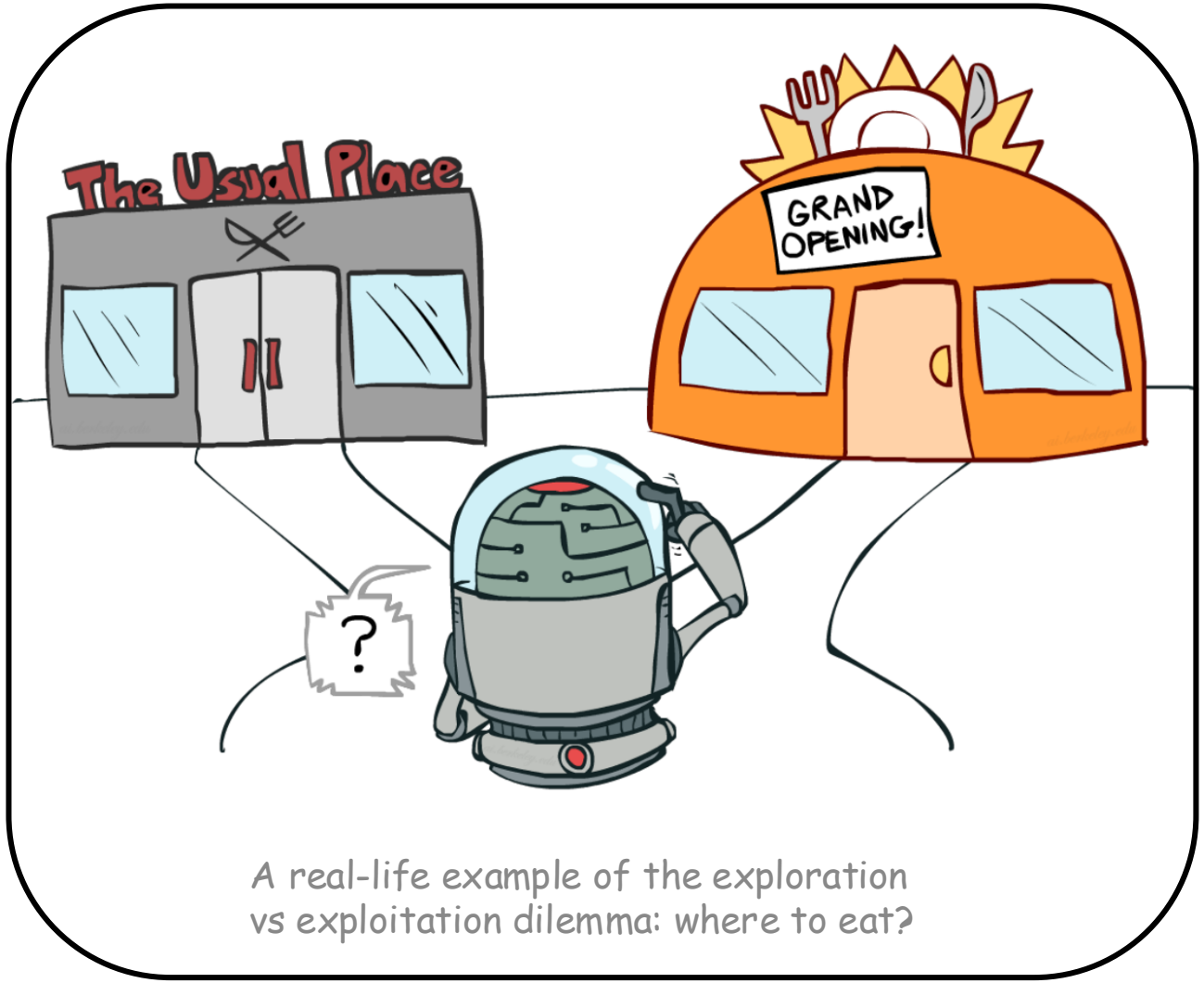


Figure 1: *Left:* Entropy collapse and performance saturation. Over 95% entropy drop/performance gains take place at the early stage of RL training. The model then reaches a plateau with little improvement. *Right:* The predictable relationship between validation performance and policy entropy. Without intervention, the policy "trades" entropy for performance exponentially, showing clear ceilings that hinder further policy enhancement.

Without intervention, LLM tend exploit trade entropy for accuracy.

RLVR – Exploration and Exploitation



A real-life example of the exploration vs exploitation dilemma: where to eat?

Exploitation: Take advantage of the best option we know.

Exploration: Take some risk to collect information about unknown options.

A fundamental trade-off in RL, whether choosing the best action (token) each step or explore new actions.

RLVR – CDE (Curiosity Driven Exploration)

Curiosity-driven exploration is rooted in intrinsic motivation: just as children's natural curiosity drives them to explore the unknown, discover, and learn, fostering curiosity in artificial agents can similarly drive meaningful exploration and growth.

In reinforcement learning, we can encourage curiosity by rewarding exploration that signals learning potential. Common measures include:

- State Nolvelty/Visitation (**Count based**)
- Agent's Uncertainty. (**Prediction based**)

Optimism In face of Uncertainty: *Try things out—you might find something better!*

Dai, Runpeng, et al. "CDE: Curiosity-Driven Exploration for efficient reinforcement learning in large language models." ICLR 2026.

Actor Curiosity

	High PPL	Low PPL
Correct	↗	↗
Incorrect	↘	↘

↘ Penalize Confident Mistakes

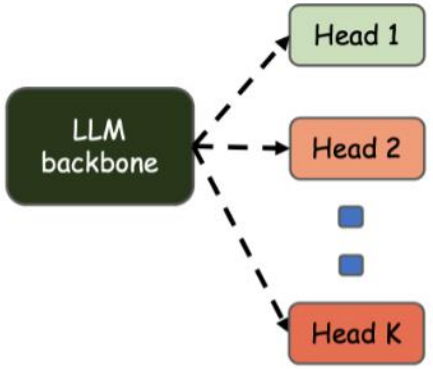
↗ Encourage Diverse Correctness

$$B_{\text{actor}}(q,o) = -\frac{1}{T}\sum_{t=1}^{\top}\log \pi(o_t|o_{<t},q)$$

$$\hat{r}(q,o) = r(q,o) + \omega \min\left(\frac{|r(q,o)|}{\kappa}, \alpha B_{\text{actor}}(q,o)\right)$$

Main Idea: Increase the reward of trajectories with higher Perplexity(PPL).

Critic Curiosity



Advantage calculation

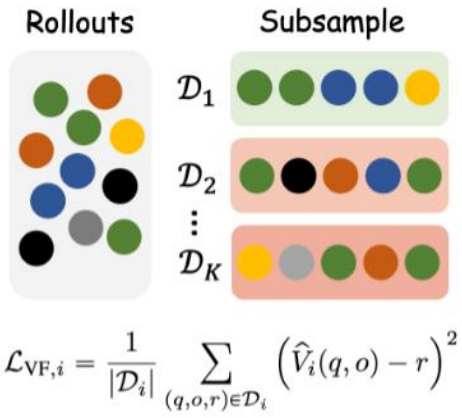
$$\hat{A}_{i,t} = \underbrace{\sum_{l=t}^{|o_i|}(\gamma\lambda)^{l-t}\hat{\delta}_{i,l}}_{\approx \tilde{A}_{i,t}} + \omega \min\left(\frac{|\tilde{A}_{i,t}|}{\kappa}, \alpha B_{\text{critic}}(q, o_{i,\leqslant t+1})\right)$$

$$B_{\text{critic}}(o_{i,\leqslant t+1}, q) = \text{std}\left(\{\hat{V}_j(q, o_{i,\leqslant t+1}) \mid 1 \leqslant j \leqslant K\}\right)$$

Exploration Bonus

$$\hat{\delta}_{i,l} = r_{i,l} + \frac{\gamma}{K}\sum_{1\leqslant j\leqslant K}\hat{V}_j(q, o_{i,\leqslant l+1}) - \frac{1}{K}\sum_{1\leqslant j\leqslant K}\hat{V}_j(q, o_{i,\leqslant l})$$

Ensemble Estimation



$$\mathcal{L}_{\text{VF},i} = \frac{1}{|\mathcal{D}_i|}\sum_{(q,o,r)\in\mathcal{D}_i}\left(\hat{V}_i(q,o) - r\right)^2$$

Main Idea: Bootstrap to capture the uncertainty of value estimate.

Dai, Runpeng, et al. "CDE: Curiosity-Driven Exploration for efficient reinforcement learning in large language models." ICLR 2026.

RLVR – Other Debates

Does Reinforcement Learning Really Incentivize Reasoning Capacity in LLMs Beyond the Base Model?

Yang Yue[1*†], Zhiqi Chen[1*], Rui Lu[1], Andrew Zhao[1], Zhaokai Wang[2], Yang Yue[1], Shiji Song[1], and Gao Huang[1✉]

[1] LeapLab, Tsinghua University [2] Shanghai Jiao Tong University

* Equal Contribution † Project Lead ✉ Corresponding Author

Pass@1 better after RLVR

Pass@K worse after RLVR

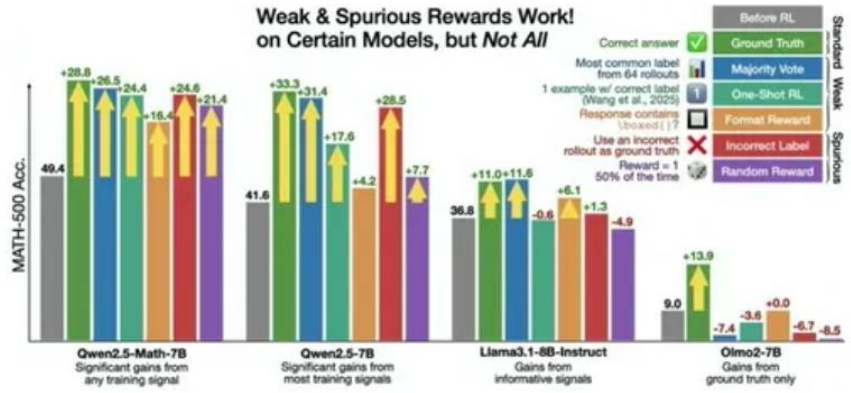
— compared to the base LLM —

Stella Li CogSci2025
@StellaLisy

We cracked RLVR with... Random Rewards?!
Training Qwen2.5-Math-7B with our Spurious Rewards improved MATH-500 by:
- Random rewards: +21%
- Incorrect rewards: +25%
- (FYI) Ground-truth rewards: + 28.8%
How could this even work!? Here's why:
Blogpost: tinyurl.com/spurious-rewar...

Spurious Rewards: Rethinking Training Signals in RLVR

Rulin Shao[1*] Shuyue Stella Li[1*] Rui Xin[1*] Scott Geng[1*] Yiping Wang[1]
Sewoong Oh[1] Simon Shaolei Du[1] Nathan Lambert[2] Sewon Min[3] Ranjay Krishna[1,2]
Yulia Tsvetkov[1] Hannaneh Hajishirzi[1,2] Pang Wei Koh[1,2] Luke Zettlemoyer[1]
[1]University of Washington [2]Allen Institute for Artificial Intelligence
[3]University of California, Berkeley
{rulins,stelli,rx31,sgeng}@cs.washington.edu



Even random reward works?

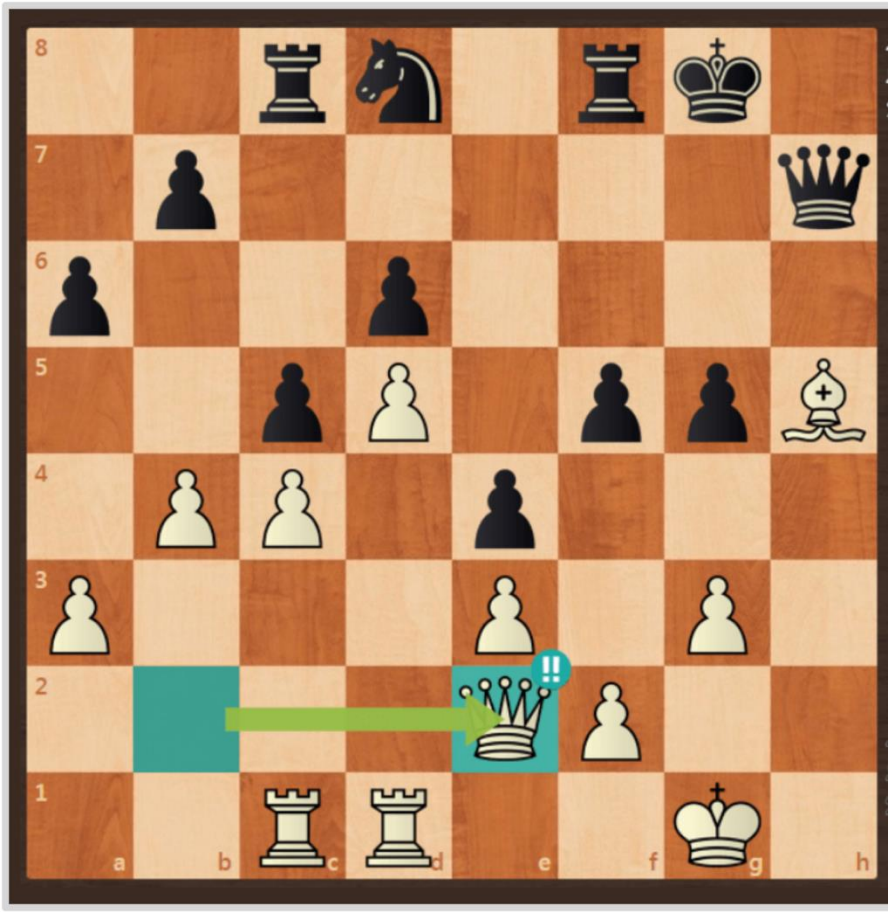
Recap: SFT and RLVR

Off-policy training (SFT) relies on target outputs from external source that the student learns to imitate.

- Off-policy learns contexts frequented by the teachers, not ones the student itself will often find itself in. If the student makes an early mistake that the teacher never makes, it finds itself diverging ever farther from the states it observed in training.

On-policy training (RLVR) samples rollouts from the student model itself and assigns them some reward.

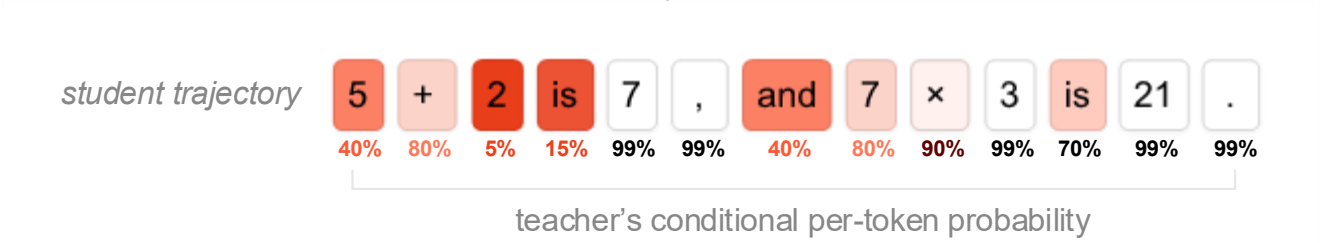
- On-policy training methods such as RLVR often rely on sparse, delayed rewards, making credit assignment difficult and optimization slow or unstable. Their performance may also plateau when the current policy cannot explore sufficiently.



On-Policy Distillation

Kevin Lu in collaboration with others at Thinking Machines

Oct 27, 2025



The core idea of on-policy distillation is to sample trajectories from the *student* model and use a high-performing teacher to grade *each token* of each trajectory. Returning to our math example above, on-policy distillation would score each step of the solution punishing the mistakes that caused the student to arrive at the wrong answer while reinforcing the ones that were executed correctly.

Method	Sampling	Reward signal
Supervised finetuning	off-policy	dense
Reinforcement learning	on-policy	sparse
On-policy distillation	on-policy	dense

On-Policy Distillation

On-policy distillation can use a variety of loss functions for grading the student's trajectories. Per-token reverse KL — the divergence between the student's and teacher's distribution for each token conditioned on the same prior trajectory:

$$\mathrm{KL}\left(\pi_\theta || \pi_{\text{teacher}}\right) = \mathbb{E}_{x\sim\pi_\theta}\left[\log \pi_\theta(x_{t+1}|x_{1..t}) - \log \pi_{\text{teacher}}(x_{t+1}|x_{1..t})\right]$$

If we look at SFT, it is forward KL

$$\mathrm{KL}\left(\pi_{\text{teacher}} \| \pi_\theta\right) = \mathcal{L}_{\text{SFT}}(\theta) - H(\pi_{\text{teacher}}).$$

$$\underbrace{x \sim \pi_{\text{teacher}}}_{\text{SFT}} \Rightarrow \mathrm{KL}(\pi_{\text{teacher}} \| \pi_\theta) \qquad \underbrace{x \sim \pi_\theta}_{\text{OPD}} \Rightarrow \mathrm{KL}(\pi_\theta \| \pi_{\text{teacher}})$$


Thank You!