CS395T: Communicative & Grounded AI Agents

Learning & Optimization for Agents

Fangcong Yin

September 09, 2025

What to Learn

- Task-specific knowledge
- General knowledge
  - Language/instruction
  - How to interact with environments/feedback

How to Learn

- Pre-LLM approaches
  - Traditional ML
  - Artzi and Zettlemoyer 2013
- Pretrained + SFT
  - Gorilla
- RL
  - DeepSeekMath
  - Agentic reasoning
  - ToolRL

Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions

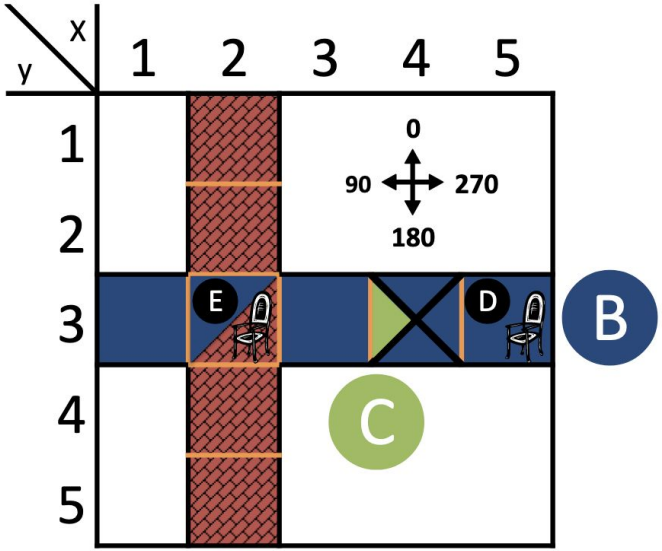
Yoav Artzi, Luke Zettlemoyer

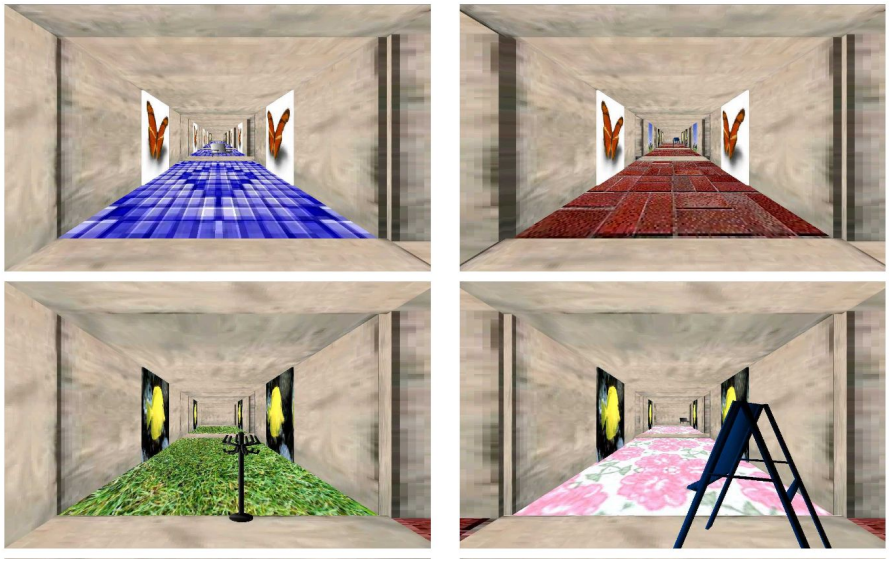
TACL 2013

Overview

- What to learn
  - Task: Spatial navigation using natural language instructions
  - General: Understanding of spatial relations and instructions
- How to learn
  - Weak supervision signal
  - Joint model of understanding and action
  - Coarse-to-fine lexicon
  - Perceptron update

What to Learn: Task

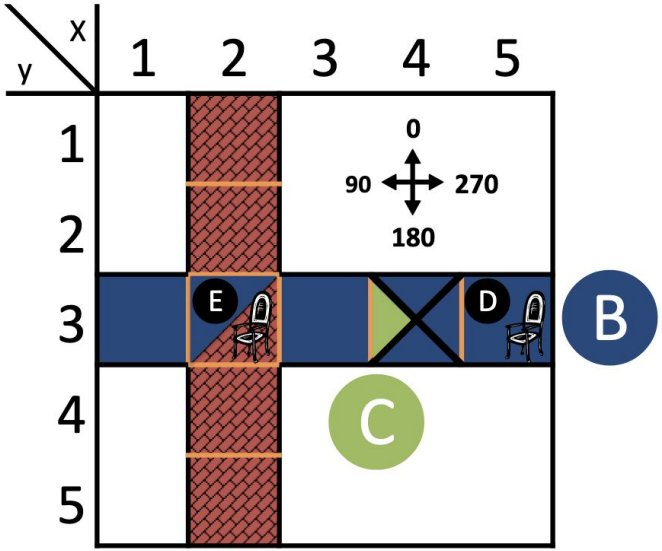




Instruction: Move the chair at the intersection of the hallways to the far end of the blue hall.

Instruction: Facing the lamp, go until you reach a chair.

What to Learn: Task



“Instruction: Move the chair at the intersection of the hallways to the far end of the blue hall.”

Language Understanding

- Spatial relation
- Disambiguation

Instruction-action Mapping

- Executable actions
- Implicit requests

What to Learn: Formulation

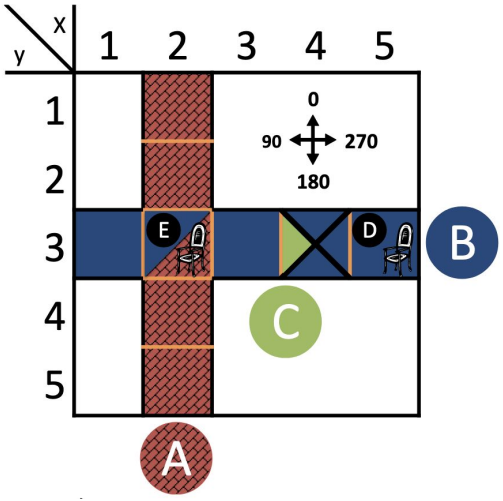
Task Let $\mathcal{S}$ be the set of possible environment states and $\mathcal{A}$ be the set of possible actions. Given a start state $s \in \mathcal{S}$ and a natural language instruction x, we aim to generate a sequence of actions $\vec{a} = \langle a_1, \ldots, a_n \rangle$, with each $a_i \in \mathcal{A}$, that performs the steps described in x.

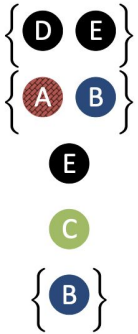
Lexicon
(Language Understanding)

Parameter
(Instruction-Action Mapping)

What to Learn: Formulation

- **Lexicon** (Language Understanding)
  - Mapping words/phrases to categories of logical forms
  - Combination of atomic maps

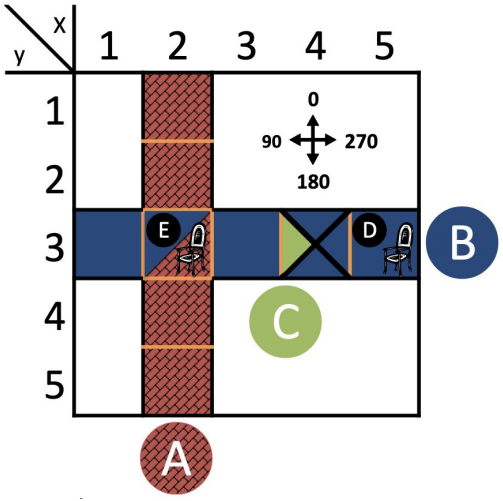




(a) chair
$\lambda x.chair(x)$
(b) hall
$\lambda x.hall(x)$
(c) the chair
$\iota x.chair(x)$
(d) you
you
(e) blue hall
$\lambda x.hall(x) \wedge blue(x)$

What to Learn: Formulation

- **Parameter** (Instruction-Action Mapping)
  - Mapping instructions to logical expressions
  - Logical expressions lead to executable actions



Imp.: move from the sofa to the chair

LF: $\lambda a.move(a) \wedge to(a, \iota x.chair(x)) \wedge from(a, \iota y.sofa(y))$

How to Learn: CCG

- **Combinatory Categorical Grammar** (CCG)
  - Lexicon
  - Combinators: Pre-defined rules to combine lexical categories

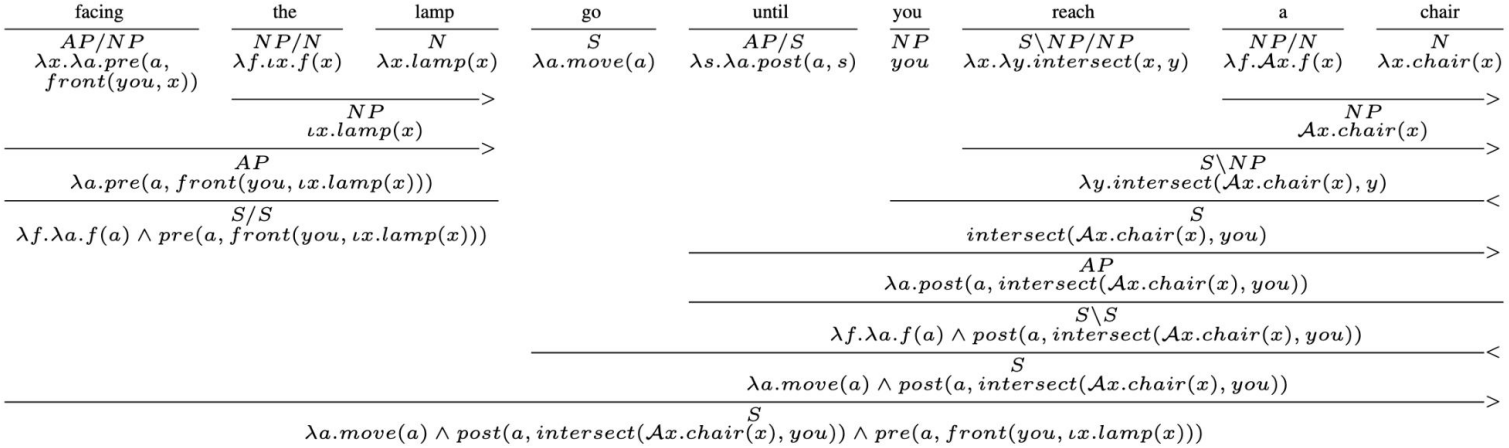


Figure 4: A CCG parse showing adverbial phrases and topicalization.

Forward combination: →
Backward combination: ←

How to Learn: CCG

- **Weighted CCG**
  - For a given input, rank different CCG parses with linear weights

$$y^*(x) = \arg \max_{y \in \mathbf{GEN}(x;\Lambda)} \theta \cdot \phi(x, y)$$

Lexicon

Parameter

How to Learn: Weak Supervision

- **Weak Supervision**
  - Validation Function: Mapping the action sequence to binary reward
  - Strong supervision
    - We have ground truth actions AND intermediate logical forms
  - Weak supervision
    - Trace
    - Final state

How to Learn: Algorithm

Algorithm:

Initialize θ using Λ_0 , $\Lambda \leftarrow \Lambda_0$

For $t = 1 \ldots T, i = 1 \ldots n$:

Step 1: (Lexical generation)

a. Set $\lambda_G \leftarrow GENLEX(x_i, s_i, \mathcal{V}_i; \Lambda, \theta)$, $\lambda \leftarrow \Lambda \cup \lambda_G$

b. Let E be the k highest scoring executions from $GEN(x_i, s_i; \lambda)$ which use at most one entry from λ_G

c. Select lexical entries from the highest scoring valid parses: $\lambda_i \leftarrow \bigcup_{e \in MAXV_i(E;\theta)} LEX(e^y)$

d. Update lexicon: $\Lambda \leftarrow \Lambda \cup \lambda_i$

Step 2: (Update parameters)

a. Set $G_i \leftarrow MAXV_i(GEN(x_i, s_i; \Lambda); \theta)$ and $B_i \leftarrow \{e | e \in GEN(x_i, s_i; \Lambda) \wedge \mathcal{V}_i(e^{\vec{a}}) \neq 1\}$

b. Construct sets of margin violating good and bad parses:

$R_i \leftarrow \{g | g \in G_i \wedge \exists b \in B_i \; s.t. \; \langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$

$E_i \leftarrow \{b | b \in B_i \wedge \exists g \in G_i \; s.t. \; \langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$

c. Apply the additive update:

$\theta \leftarrow \theta + \frac{1}{|R_i|} \sum_{r \in R_i} \Phi_i(r) - \frac{1}{|E_i|} \sum_{e \in E_i} \Phi_i(e)$

Output: Parameters θ and lexicon Λ

- **Lexical Generation**
  - Start from a coarse lexicon
  - Add entries from the highest scoring parses to the lexicon

How to Learn: Algorithm

Algorithm:

Initialize θ using Λ_0 , $\Lambda \leftarrow \Lambda_0$

For $t = 1 \ldots T, i = 1 \ldots n$:

Step 1: (Lexical generation)

a. Set $\lambda_G \leftarrow GENLEX(x_i, s_i, \mathcal{V}_i; \Lambda, \theta), \lambda \leftarrow \Lambda \cup \lambda_G$

b. Let E be the k highest scoring executions from $GEN(x_i, s_i; \lambda)$ which use at most one entry from λ_G

c. Select lexical entries from the highest scoring valid parses: $\lambda_i \leftarrow \bigcup_{e \in MAXV_i(E;\theta)} LEX(e^y)$

d. Update lexicon: $\Lambda \leftarrow \Lambda \cup \lambda_i$

Step 2: (Update parameters)

a. Set $G_i \leftarrow MAXV_i(GEN(x_i, s_i; \Lambda); \theta)$
and $B_i \leftarrow \{e | e \in GEN(x_i, s_i; \Lambda) \wedge \mathcal{V}_i(e^{\vec{a}}) \neq 1\}$

b. Construct sets of margin violating good and bad parses:
$R_i \leftarrow \{g | g \in G_i \wedge$
$\exists b \in B_i \; s.t. \; \langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$
$E_i \leftarrow \{b | b \in B_i \wedge$
$\exists g \in G_i \; s.t. \; \langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$

c. Apply the additive update:
$\theta \leftarrow \theta + \frac{1}{|R_i|} \sum_{r \in R_i} \Phi_i(r) - \frac{1}{|E_i|} \sum_{e \in E_i} \Phi_i(e)$

Output: Parameters θ and lexicon Λ

- **Parameter Update**
  - Select good and bad parses based n the validation function
  - Apply perceptron updates

Results

- **Key Takeaways**
  - Weak supervision is good enough!

	Single Sentence	Sequence
Chen and Mooney (2011)	54.4	16.18
Chen (2012)	57.28	19.18
+ additional data	57.62	20.64
Kim and Mooney (2012)	57.22	20.17
Trace validation	**65.28** (5.09)	**31.93** (3.26)
Final state validation	**64.25** (5.12)	**30.9** (2.16)

	Single Sentence	Sequence
Final state validation		
Complete system	81.98 (2.33)	59.32 (6.66)
No implicit actions	77.7 (3.7)	38.46 (1.12)
No joint execution	73.27 (3.98)	31.51 (6.66)
Trace validation		
Complete system	82.74 (2.53)	58.95 (6.88)
No implicit actions	77.64 (3.46)	38.34 (6.23)
No joint execution	72.85 (4.73)	30.89 (6.08)

Discussion

- **Key Takeaways**
  - Weak supervision is enough!
    - Can we scale up the data?
  - CCG still requires hand-crafted aspects.
    - Can we use learn generalizable rules through data without handcrafting?
  - Language understanding and action can be jointly modeled.
    - Can we learn them jointly through interaction with the environment?

Discussion

- **Relevance to Recent Work**
  - Weak supervision is enough!
    - Can we scale up the data?
      - **Pre-trained language models**
  - CCG still requires hand-crafted rules.
    - Can we use learn such rules with proper supervision?
      - **Instruction tuning & Supervised fine-tuning**
  - Language understanding and action can be jointly modeled.
    - Can we learn them jointly through interaction with the environment?
      - **Reinforcement learning**

Gorilla: Large Language Model Connected with Massive APIs

Shishir G. Patil, Tianjun Zhang, Xin Wang, Joseph E. Gonzalez

NeurIPS 2024

Background

- **Relevance to Recent Work**
  - Weak supervision is enough!
    - Can we scale up the data?
      - **Pre-trained language models**
  - CCG still requires hand-crafted rules.
    - Can we use learn such rules with proper supervision?
      - **Instruction tuning & Supervised fine-tuning**
  - Language understanding and action can be jointly modeled.
    - Can we learn them jointly through interaction with the environment?
      - **Reinforcement learning**

Background

- **Pre-trained LLMs are Tool-use Agents**
  - Pre-trained LLMs can use tools given few-shot demonstrations in context
    - Tools: python programs; program solvers
    - PAL (Gao et al., 2023); SAT LM (Ye et al., 2023)

Q: I have a chair, two potatoes, a cauliflower, a lettuce head, two tables, a cabbage, two onions, and three fridges. How many vegetables do I have?

```
# note: I'm not counting the chair, tables,
    or fridges
vegetables_to_count = {
    'potato': 2,
    'cauliflower': 1,
    'lettuce head': 1,
    'cabbage': 1,
    'onion': 2
}
answer = sum(vegetables_to_count.values())
```

PAL (Gao et al., 2023)

Motivation

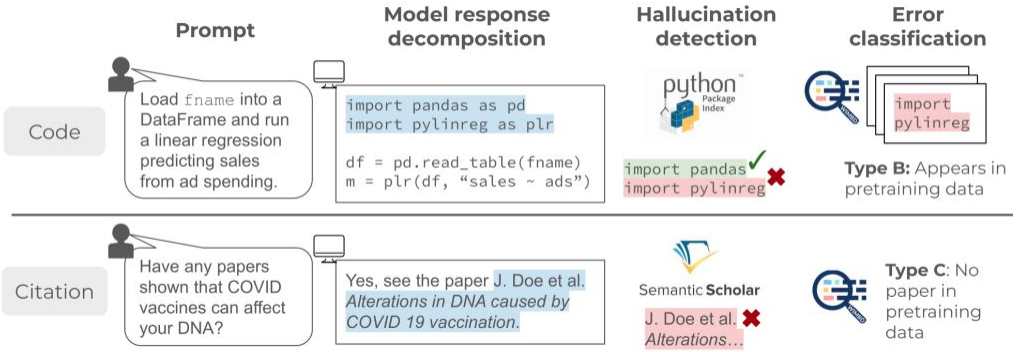
- **Pre-trained LLMs can Hallucinate**
  - LLMs are not trained to ground, so they can hallucinate
  - Particularly dangerous for tool-use cases

"Help me find an API to convert the spoken language in a recorded audio to text using Torch Hub."

Correct (Gorilla)	`<domain>: Speech-to-Text`
	`<api_provider>: TorchHub`
	`<code>:`
	`asr_model = torch.hub.load('snakers4/silero-models', 'silero_sst')`
	`result = asr_model.transcribe(audio_path)`

API Argument Hallucination (GPT-4)	`<domain>: Speech-to-Text`
	`<api_provider>: TorchHub`
	`<code>:`
	`asr_model = torch.hub.load('snakers4/silero-models', 'asr', source='local')`
	`result = asr_model.transcribe(audio_path)`

Incorrect API Call (Claude)	`<domain>: Audio-Translation`
	`<api_provider>: Pytorch`
	`<code>:`
	`import torchaudio`
	`translation = torchaudio.pipelines.WAV2VEC2_ASR_PIPELINE("audio.wav")`



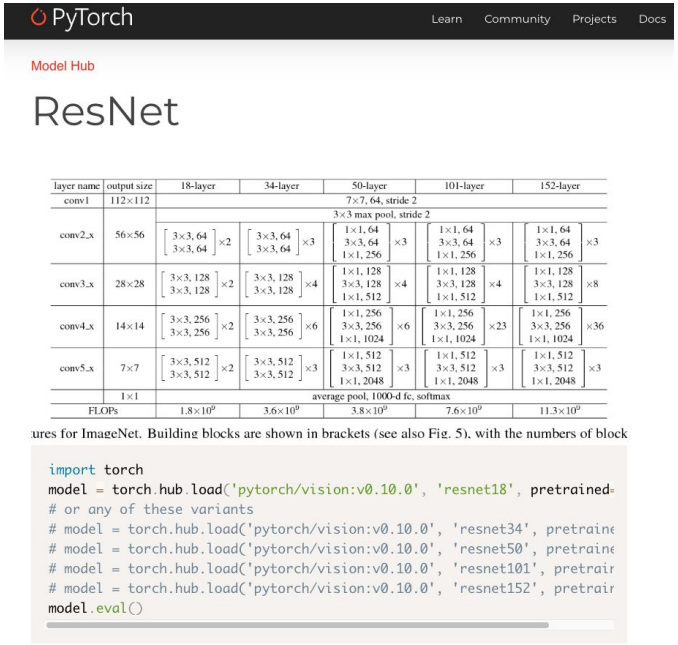
HaLoGen; Ravichander et al., 2025

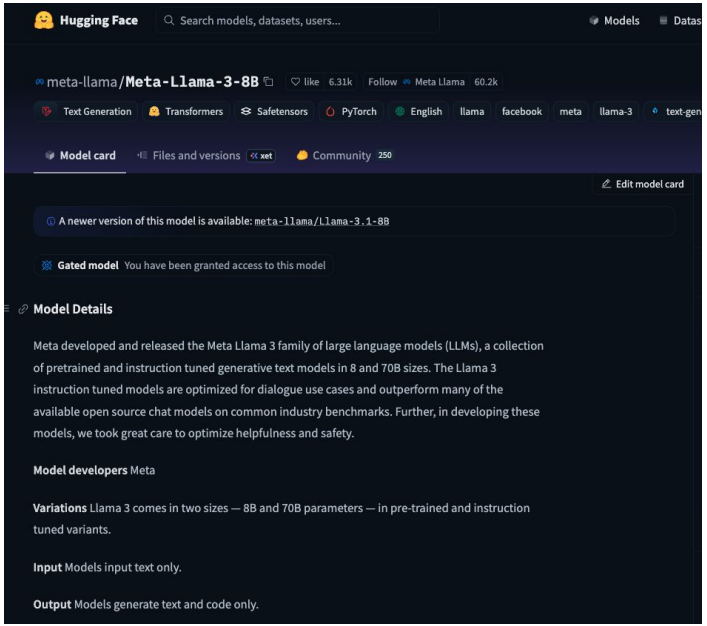
Motivation

- **Reducing Hallucinations of Tool-Use LLM Agents**
  - CCG still requires hand-crafted rules.
    - Can we use learn such rules with proper supervision?
      - **Instruction tuning & Supervised fine-tuning**
  - **Can we train LLM agents to ground their tool uses in reliable sources?**
  - **Can we provide proper supervision for such learning?**

What to Learn: Task

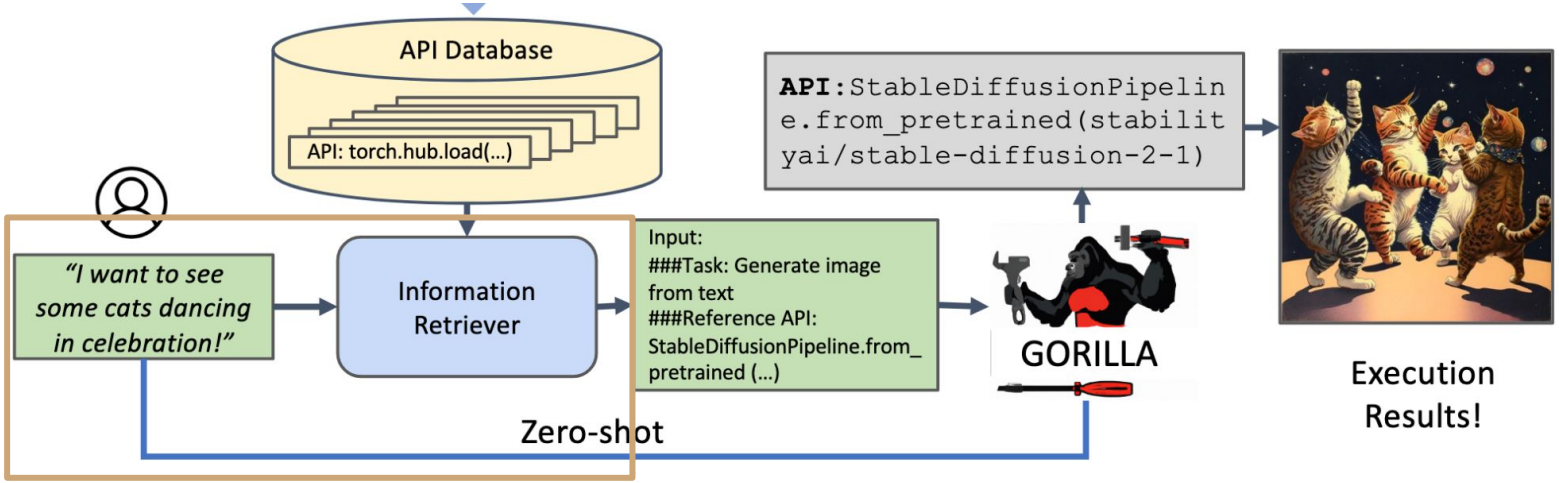
- **API Calls**
  - APIBench: 1600 ML APIs (TorchHub, TensorFlow Hub, HuggingFace)





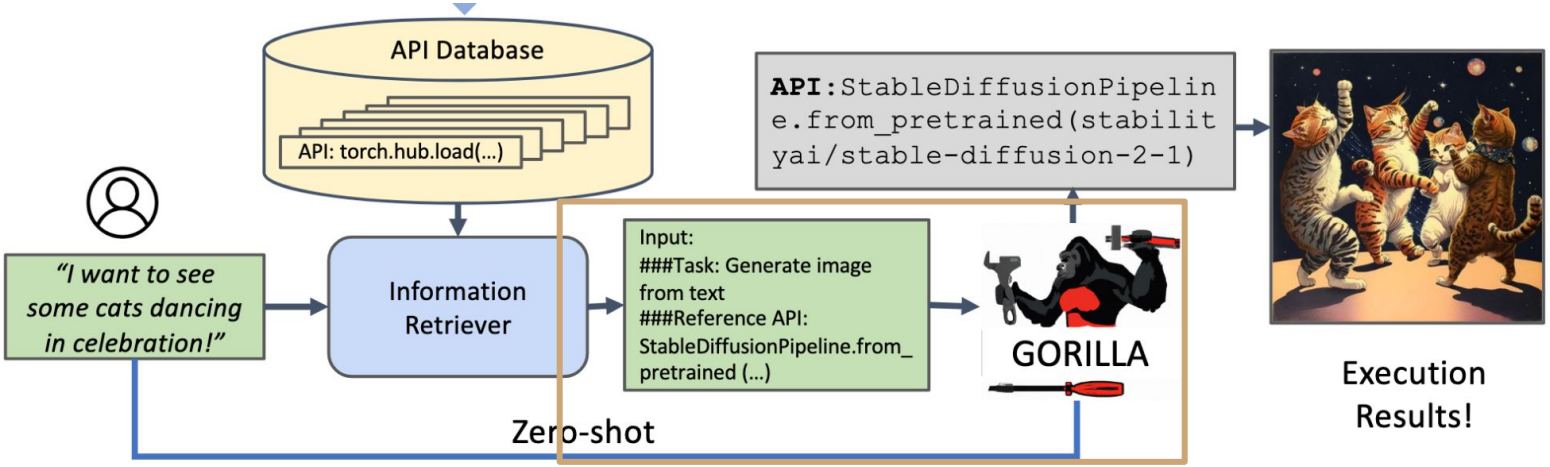
What to Learn: Grounding

- **Grounding**
  - Train LLMs to ground tool use in API documentation retrieved by retrievers



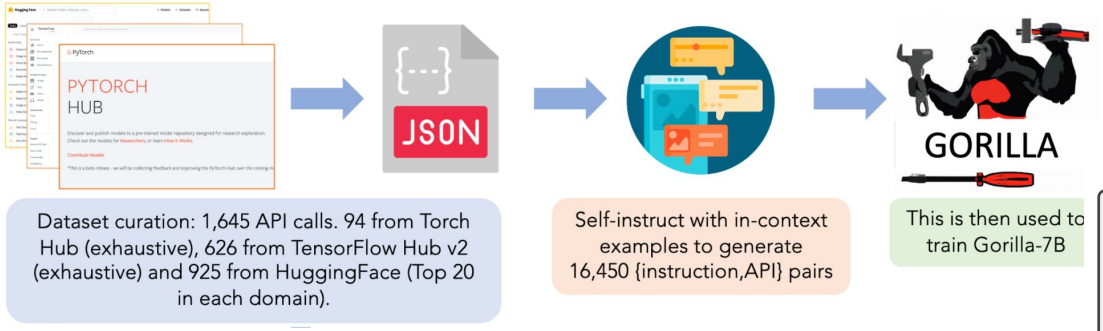
How to Learn: RAT

- **Retriever-Aware Training (RAT)**
  - Fine-tune LLMs to use retrieved documentation by retrievers in the context to make correct tool calls
    - The retrieved documentation might or might not be relevant!



How to Learn: Supervised Fine-tuning

- **Supervised Fine-tuning**
  - Curate SFT data for RAT by few-shot prompting GPT4
  - ~ 13K instruction-response pairs for training
  - Base model: Llama-7B



Dataset curation: 1,645 API calls. 94 from Torch Hub (exhaustive), 626 from TensorFlow Hub v2 (exhaustive) and 925 from HuggingFace (Top 20 in each domain).

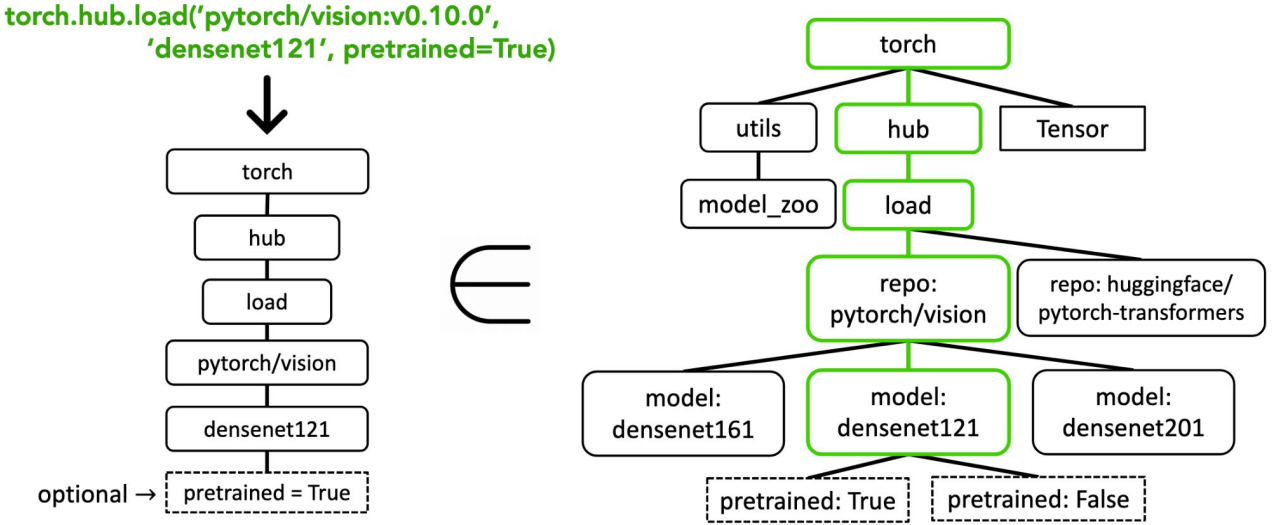
Self-instruct with in-context examples to generate 16,450 {instruction,API} pairs

This is then used to train Gorilla-7B

User: Our customer is a zoo and we want to help them detect movement of different animals. Write a python program in 1 to 2 lines to call API in TensorFlowHub.
Assistant:
<domain>: image-object-detection,
<api_call>: detector = hub.load('https://tfhub.dev/google/openimages_v4/ssd_mobilenet_v2/1'),
<api_provider>: TensorFlow Hub,
<explanation>: This API call loads the pre-trained Single Shot Detector (SSD) MobileNet V2 model from TensorFlow Hub, which has been trained on the Open Images v4 dataset. This model is used for object detection in images. Once loaded, you can pass an image (and an optional score threshold for detection confidence) to the detector to get back the detected objects along with their bounding boxes, class labels, and associated confidence scores.
<code>: import tensorflow as tf; import tensorflow_hub as hub; # Load the SSD MobileNet V2 model; detector = hub.load('https://tfhub.dev/google/openimages_v4/ssd_mobilenet_v2/1'); # Use the model to detect objects in an image (assuming 'image' is a TensorFlow tensor representing your image); result = detector(image, score_threshold=0.5)

Evaluation

- **AST-based Evaluation**
  - Evaluation based on the matching of AST Sub-tree
  - Capture hallucinations and errors



Results

- **Key Takeaways**
  - With SFT, Gorilla can match large close-source models in zero-shot and retriever-augmented setting
  - Gorilla reduces hallucinations, even in 0-shot setting

Table 1: **Evaluating LLMs on Torch Hub, HuggingFace, and Tensorflow Hub APIs**

LLM (retriever)	TorchHub			HuggingFace			TensorFlow Hub		
	overall ↑	hallu ↓	err ↓	overall ↑	hallu ↓	err ↓	overall ↑	hallu ↓	err ↓
LLAMA (0-shot)	0	100	0	0.00	97.57	2.43	0	100	0
GPT-3.5 (0-shot)	48.38	18.81	32.79	16.81	35.73	47.46	41.75	47.88	10.36
GPT-4 (0-shot)	38.70	36.55	24.7	19.80	37.16	43.03	18.20	78.65	3.13
Claude (0-shot)	18.81	65.59	15.59	6.19	77.65	16.15	9.19	88.46	2.33
Gorilla (0-shot)	**59.13**	**6.98**	33.87	**71.68**	**10.95**	17.36	**83.79**	**5.40**	10.80
LLAMA (BM-25)	8.60	76.88	14.51	3.00	77.99	19.02	8.90	77.37	13.72
GPT-3.5 (BM-25)	38.17	6.98	54.83	**17.26**	8.30	74.44	**54.16**	3.64	42.18
GPT-4 (BM-25)	35.48	11.29	53.22	16.48	15.93	67.59	34.01	37.08	28.90
Claude (BM-25)	39.78	5.37	54.83	14.60	15.82	69.58	35.18	21.16	43.64
Gorilla (BM-25)	**40.32**	**4.30**	55.37	17.03	**6.42**	76.55	41.89	**2.77**	55.32
LLAMA (GPT-Index)	14.51	75.8	9.67	10.18	75.66	14.20	15.62	77.66	6.71
GPT-3.5 (GPT-Index)	60.21	1.61	38.17	29.08	7.85	44.80	**65.59**	3.79	30.50
GPT-4 (GPT-Index)	59.13	1.07	39.78	44.58	11.18	44.25	43.94	31.53	24.52
Claude (GPT-Index)	60.21	3.76	36.02	41.37	18.81	39.82	55.62	16.20	28.17
Gorilla (GPT-Index)	**61.82**	**0**	38.17	**47.46**	**8.19**	44.36	64.96	**2.33**	32.70
LLAMA (Oracle)	16.12	79.03	4.83	17.70	77.10	5.20	12.55	87.00	0.43
GPT-3.5 (Oracle)	66.31	1.60	32.08	89.71	6.64	3.65	**95.03**	**0.29**	4.67
GPT-4 (Oracle)	66.12	0.53	33.33	85.07	10.62	4.31	55.91	37.95	6.13
Claude (Oracle)	63.44	3.76	32.79	77.21	19.58	3.21	74.74	21.60	3.64
Gorilla (Oracle)	**67.20**	**0**	32.79	**91.26**	**7.08**	1.66	94.16	1.89	3.94

Results

- **Is SFT better than in-context learning?**
  - Yes!

Table 5: **Evaluating Gorilla 0-shot with GPT 3-shot incontext examples**

	HF (Acc ↑)	HF (Hall ↓)	TH (Acc ↑)	TH (Hall ↓)	TF (Acc ↑)	TF (Hall ↓)
GPT-3.5 (0-shot)	16.81	35.73	41.93	10.75	41.75	47.88
GPT-4 (0-shot)	19.80	37.16	54.30	34.40	18.20	78.65
GPT-3.5 (3 incont)	25.77	32.30	73.11	72.58	71.82	11.09
GPT-4 (3 incont)	26.32	35.84	**75.80**	**13.44**	77.37	11.97
Gorilla (0-shot)	**58.05**	**28.32**	**75.80**	16.12	**83.79**	**5.40**

Results

- **How strong of supervision the SFT data needs to have?**
  - Strong supervision (oracle retriever) is better, but weaker supervision is decent in most cases

	Gorilla without Retriever				Gorilla with Oracle retriever			
	zero-shot	BM25	GPT-Index	Oracle	zero-shot	BM25	GPT-Index	Oracle
Torch Hub (overall) ↑	59.13	37.63	60.21	54.83	0	40.32	61.82	67.20
HuggingFace (overall) ↑	71.68	11.28	28.10	45.58	0	17.04	47.46	91.26
TensorHub (overall) ↑	83.79	34.30	52.40	82.91	0	41.89	64.96	94.16
Torch Hub (Hallu) ↓	6.98	11.29	4.30	15.59	100	4.30	0	0
HuggingFace (Hallu) ↓	10.95	46.46	41.48	52.77	99.67	6.42	8.19	7.08
TensorHub (Hallu) ↓	5.40	20.43	19.70	13.28	100	2.77	2.33	1.89

Discussion

- **Does the approach scale to long-context retrieval augmentation?**
  - The current setting only retrieves k = 1 API documentation
  - In practice, we would like to retrieve much more documentations and add to the context
- **Does the approach generalize to other domains of APIs?**
  - SFT might overfit the set of APIs in the training data

DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y.K. Li, Y. Wu, Daya Guo

2024

Background

- **Relevance to Recent Work**
  - Weak supervision is enough!
    - Can we scale up the data?
      - **Pre-trained language models**
  - CCG still requires hand-crafted rules.
    - Can we use learn such rules with proper supervision?
      - **Instruction tuning & Supervised fine-tuning**
  - Language understanding and action can be jointly modeled.
    - Can we learn them jointly through interaction with the environment?
      - **Reinforcement learning**

Issues of SFT

- SFT might overfit the training data

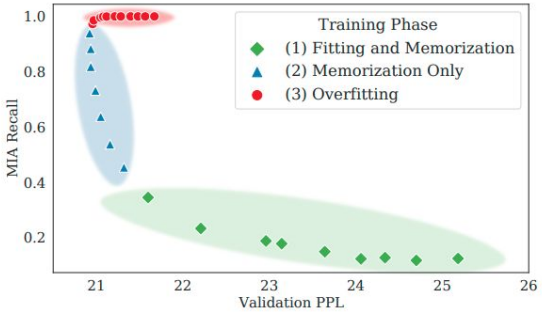


Figure 1: Each point in the graph shows the given metric values at the end of each training epoch. The rightmost lower points show the beginning, and as we move to left and upwards training progresses. We identify three separate phases within the learning process, distinguished by their memorization and generalization trends.

(Mireshghallah et al., 2022)

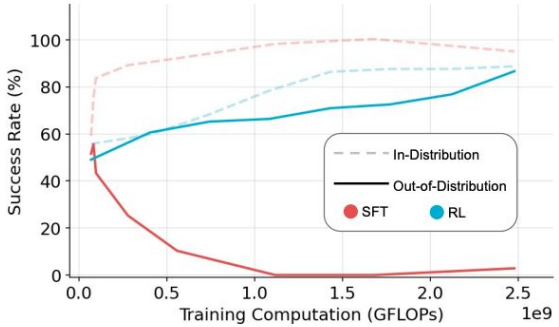
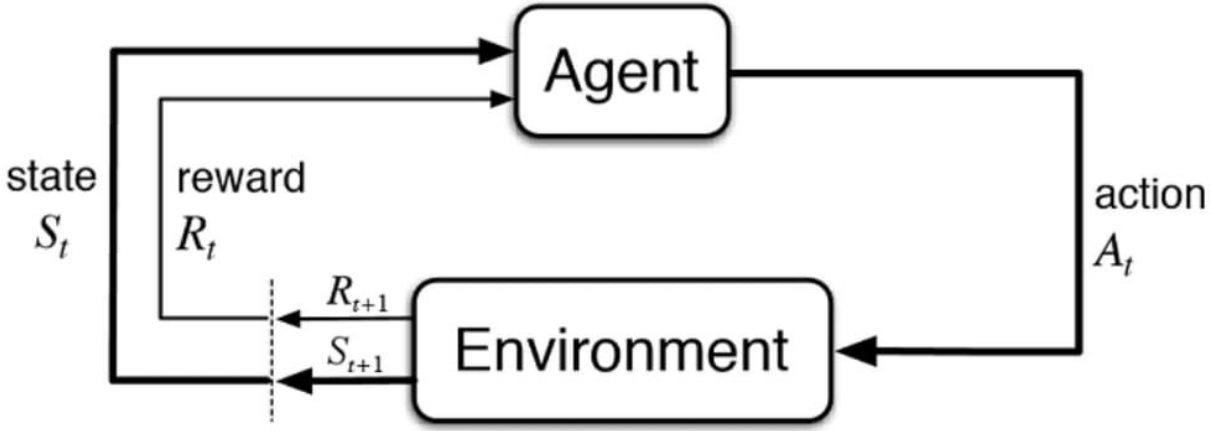


Figure 1: **A comparative study of RL and SFT on the visual navigation environment** `V-IRL` **(Yang et al., 2024a) for OOD generalization.** OOD curves represent performance on the same task, using *a different textual action space*. See detailed descriptions of the task in Section 5.1.

(Chu et al., 2025)

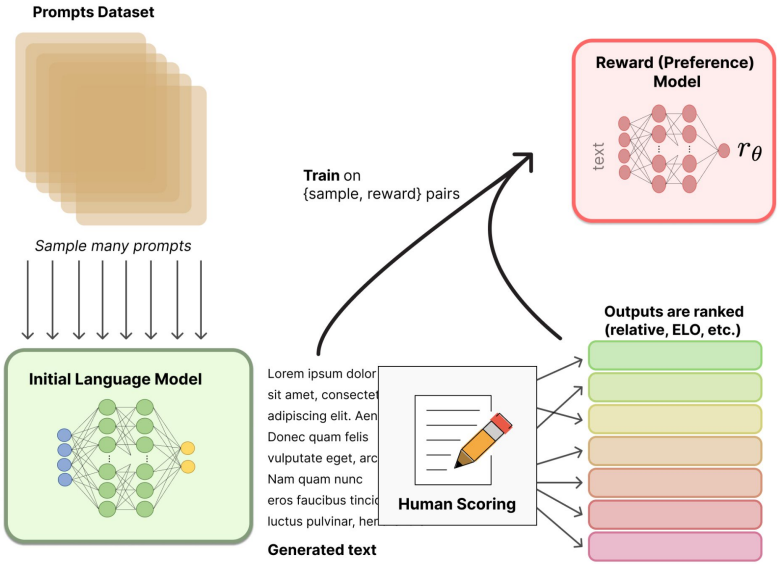
Background

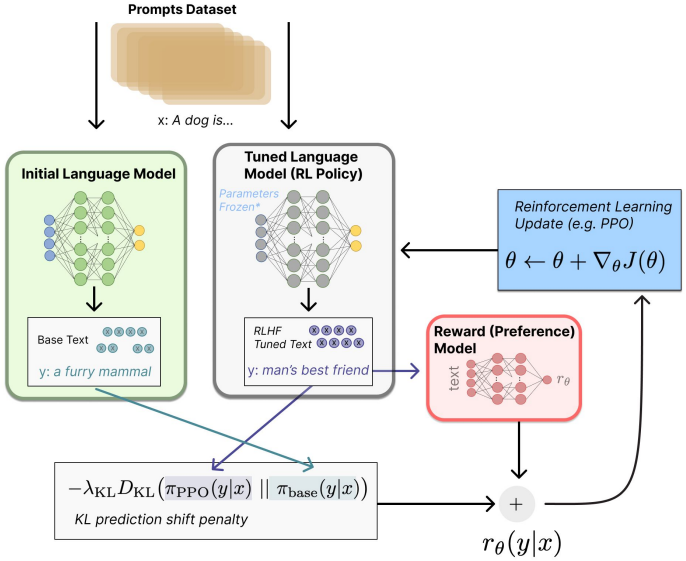
- Alternative: Reinforcement Learning
  - Agent: LLM
  - Action: Sequence generated by the LLM
  - Core differences: Reward & Environment



Background

- Alternative: Reinforcement Learning from Human Feedback (RLHF)

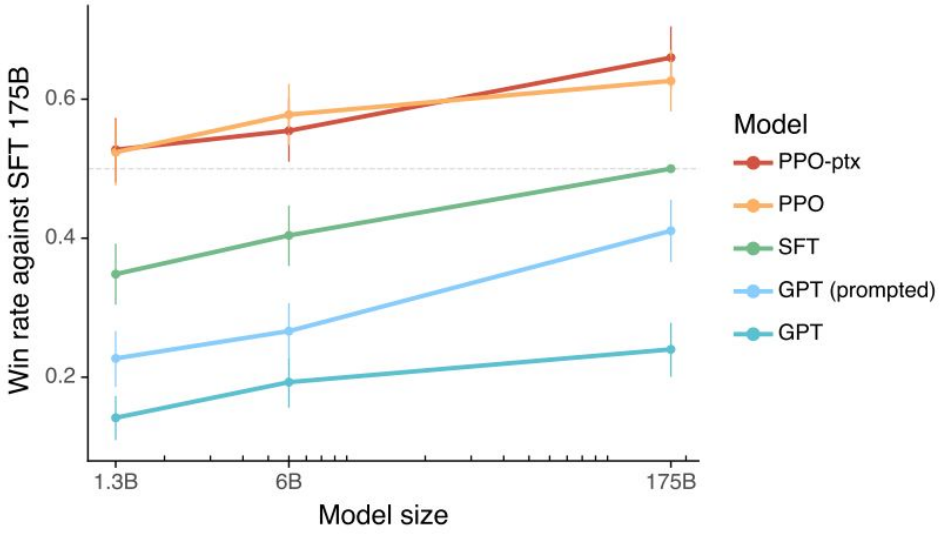




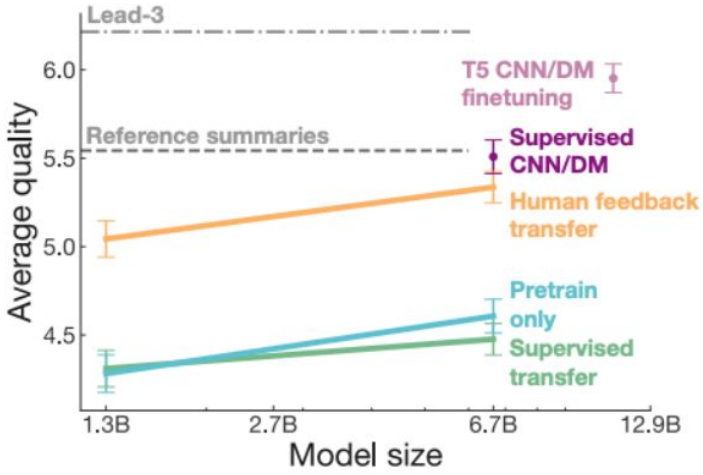
https://huggingface.co/blog/rlhf

Background

- Alternative: Reinforcement Learning from Human Feedback (RLHF)



(Ouyang et al., 2022)



(Stiennon et al., 2022)

Background

- Learning: Proximal Policy Optimization (PPO)

Algorithm 1 PPO, Actor-Critic Style

for iteration=$1, 2, \ldots$ **do**
  for actor=$1, 2, \ldots, N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

LLM

Estimated benefit of generating a sequence compared with others

Loss function

Background

- Learning: Proximal Policy Optimization (PPO)

$$\mathcal{J}_{PPO}(\theta) = \mathbb{E}[q \sim P(Q), o \sim \pi_{\theta_{old}}(O|q)] \frac{1}{|o|} \sum_{t=1}^{|o|} \min \left[\frac{\pi_\theta(o_t|q,o_{<t})}{\pi_{\theta_{old}}(o_t|q,o_{<t})} A_t, \text{clip}\left(\frac{\pi_\theta(o_t|q,o_{<t})}{\pi_{\theta_{old}}(o_t|q,o_{<t})}, 1-\varepsilon, 1+\varepsilon \right) A_t \right],$$

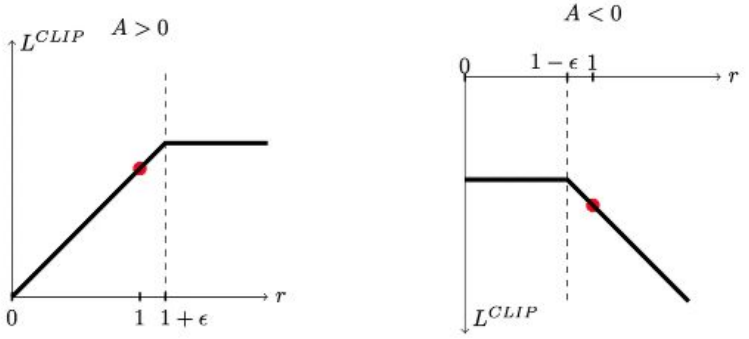


Figure 1: Plots showing one term (i.e., a single timestep) of the surrogate function L^{CLIP} as a function of the probability ratio r, for positive advantages (left) and negative advantages (right). The red circle on each plot shows the starting point for the optimization, i.e., $r = 1$. Note that L^{CLIP} sums many of these terms.

Motivation

- Supervision for RLHF
  - Preference data is not easy to collect for a lot of tasks
- PPO is not efficient to optimize
  - Require four models: reference, policy, value, and reward

What to Learn: Task

- Mathematical reasoning had been hard for LLMs

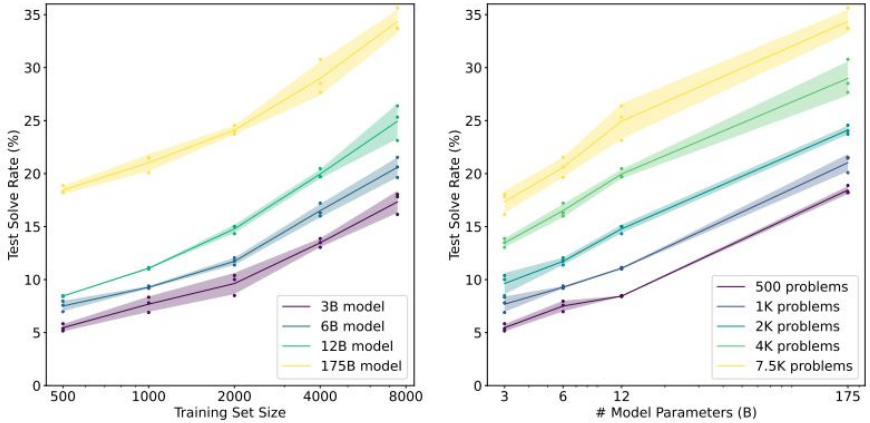
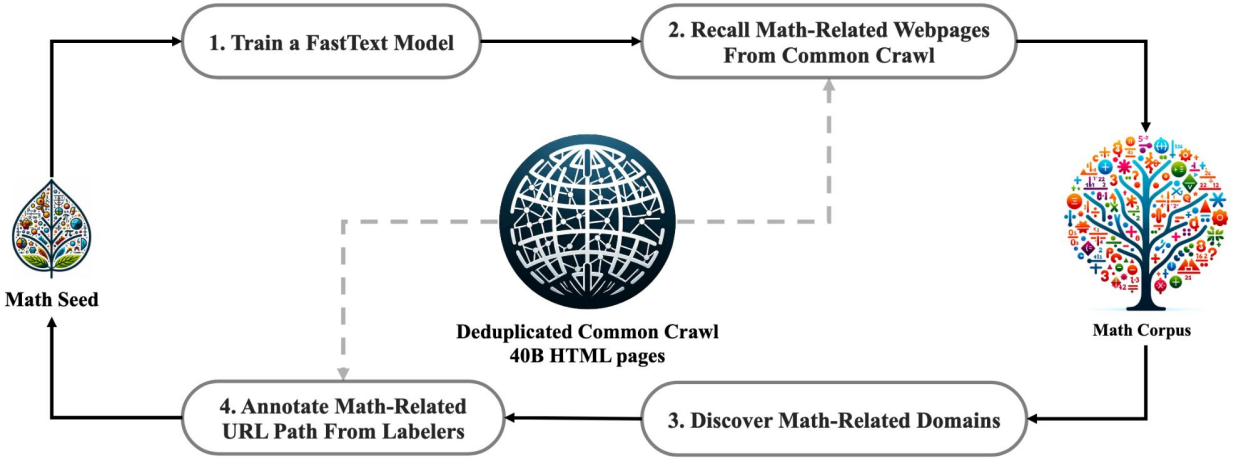


Figure 2: Final test performance for various GPT-3 model sizes after finetuning on training sets of different sizes. Mean and standard deviation is shown across 3 runs.

(Cobbe et al., 2021)

How to Learn: Pre-training Data

- Large-scale High-quality Math Pre-training Data
  - Up to 160B high-quality math data
- Mixture with Code Data
  - Code data is important for mathematical and logical reasoning



How to Learn: Math SFT

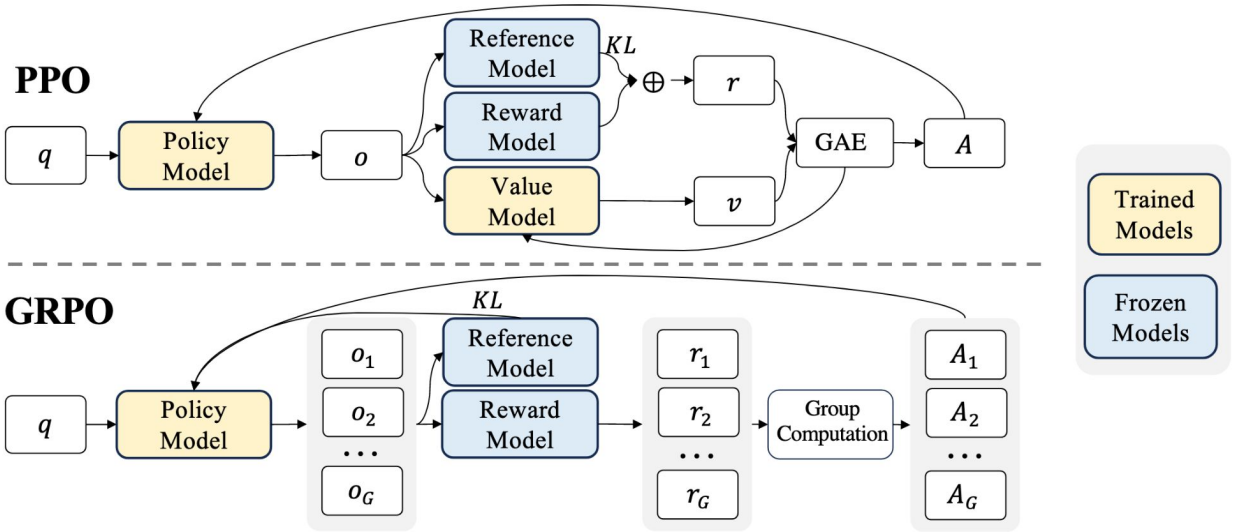
- Tool-use oriented SFT

We construct a mathematical instruction-tuning dataset covering English and Chinese problems from different mathematical fields and of varying complexity levels: problems are paired with solutions in chain-of-thought (CoT) (Wei et al., 2022), program-of-thought (PoT) (Chen et al., 2022; Gao et al., 2023), and tool-integrated reasoning format (Gou et al., 2023). The total number of training examples is 776K.

- **English mathematical datasets**: We annotate GSM8K and MATH problems with tool-integrated solutions, and adopt a subset of MathInstruct (Yue et al., 2023) along with the training set of Lila-OOD (Mishra et al., 2022) where problems are solved with CoT or PoT. Our English collection covers diverse fields of mathematics, e.g., algebra, probability, number theory, calculus, and geometry.
- **Chinese mathematical datasets**: We collect Chinese K-12 mathematical problems spanning 76 sub-topics such as linear equations, with solutions annotated in both CoT and tool-integrated reasoning format.

How to Learn: GRPO

- Group Relative Policy Optimization (GRPO)
  - Remove the value model
  - Estimate advantage based on relative rewards
  - Adding KL to the loss directly



How to Learn: GRPO

- Group Relative Policy Optimization (GRPO)
  - Remove the value model
  - Estimate advantage based on relative rewards
  - Adding KL to the loss directly

$$\mathcal{J}_{PPO}(\theta) = \mathbb{E}[q \sim P(Q), o \sim \pi_{\theta_{old}}(O|q)] \frac{1}{|o|} \sum_{t=1}^{|o|} \min\left[\frac{\pi_\theta(o_t|q,o_{<t})}{\pi_{\theta_{old}}(o_t|q,o_{<t})} A_t, \text{clip}\left(\frac{\pi_\theta(o_t|q,o_{<t})}{\pi_{\theta_{old}}(o_t|q,o_{<t})}, 1-\varepsilon, 1+\varepsilon\right) A_t\right],$$

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$

$$\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left\{\min\left[\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}\hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right] - \beta\mathbb{D}_{KL}\left[\pi_\theta||\pi_{ref}\right]\right\}, \quad (3)$$

How to Learn: GRPO

- Group Relative Policy Optimization (GRPO)
  - Reward
    - Outcome supervision
    - Process supervision

Results: Pre-Training

- Quality and scale matter

Math Corpus	Size	English Benchmarks					Chinese Benchmarks		
		GSM8K	MATH	OCW	SAT	MMLU STEM	CMATH	Gaokao MathCloze	Gaokao MathQA
No Math Training	N/A	2.9%	3.0%	2.9%	15.6%	19.5%	12.3%	0.8%	17.9%
MathPile	8.9B	2.7%	3.3%	2.2%	12.5%	15.7%	1.2%	0.0%	2.8%
OpenWebMath	13.6B	11.5%	8.9%	3.7%	31.3%	29.6%	16.8%	0.0%	14.2%
Proof-Pile-2	51.9B	14.3%	11.2%	3.7%	43.8%	29.2%	19.9%	5.1%	11.7%
DeepSeekMath Corpus	**120.2B**	**23.8%**	**13.6%**	**4.8%**	**56.3%**	**33.1%**	**41.5%**	**5.9%**	**23.6%**

Results: Pre-Training

- Code training helps

Training Setting	Training Tokens			w/o Tool Use			w/ Tool Use	
	General	Code	Math	GSM8K	MATH	CMATH	GSM8K+Python	MATH+Python
No Continual Training	–	–	–	2.9%	3.0%	12.3%	2.7%	2.3%
Two-Stage Training								
Stage 1: General Training	400B	–	–	2.9%	3.2%	14.8%	3.3%	2.3%
Stage 2: Math Training	–	–	150B	19.1%	14.4%	37.2%	14.3%	6.7%
Stage 1: Code Training	–	400B	–	5.9%	3.6%	19.9%	12.4%	10.0%
Stage 2: Math Training	–	–	150B	**21.9%**	**15.3%**	**39.7%**	17.4%	9.4%
One-Stage Training								
Math Training	–	–	150B	20.5%	13.1%	37.6%	11.4%	6.5%
Code & Math Mixed Training	–	400B	150B	17.6%	12.1%	36.3%	**19.7%**	**13.5%**

Table 6 | Investigation of how code affects mathematical reasoning under different training settings. We experiment with DeepSeek-LLM 1.3B, and evaluate its mathematical reasoning performance without and with tool use via few-shot chain-of-thought prompting and few-shot program-of-thought prompting, respectively.

Results: RL

- GRPO is better than other RL algorithms

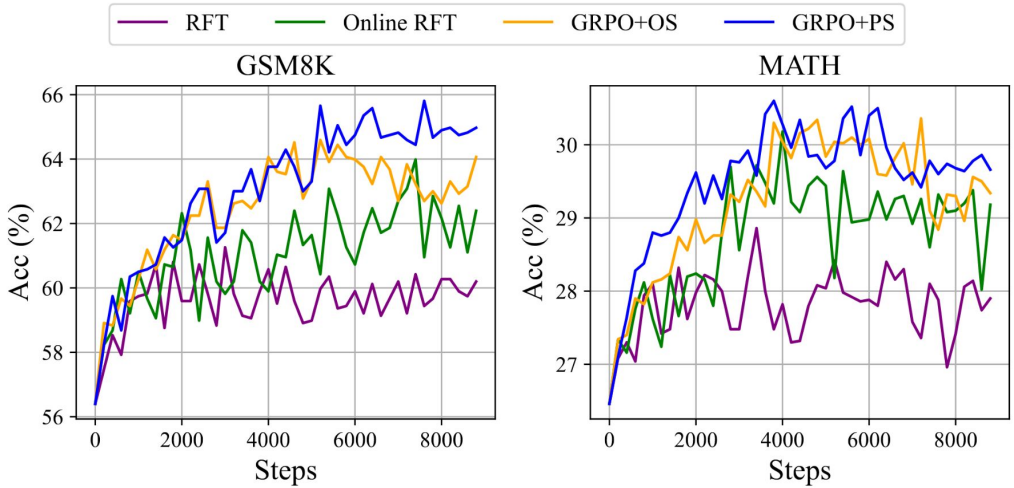


Figure 5 | Performance of the DeepSeekMath-Instruct 1.3B model, which was further trained using various methods, on two benchmarks.

Discussion: Why RL works?

- RL boosts correct response from TopK of the policy model

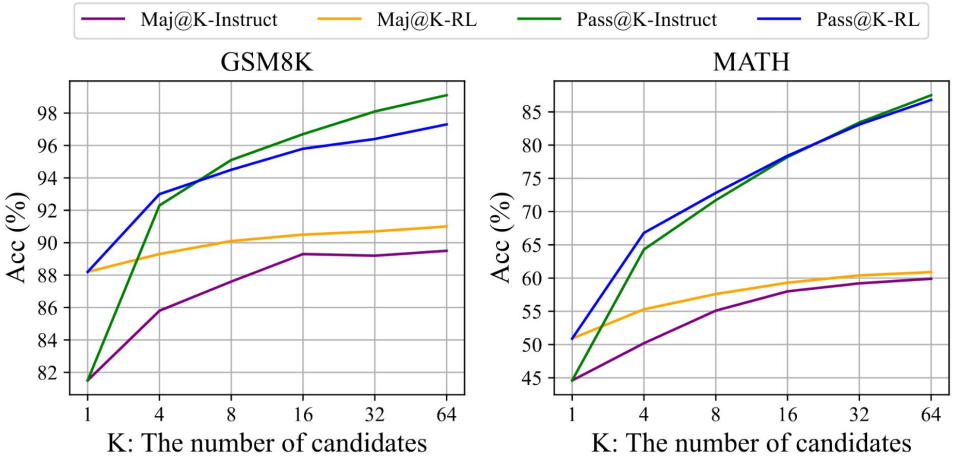
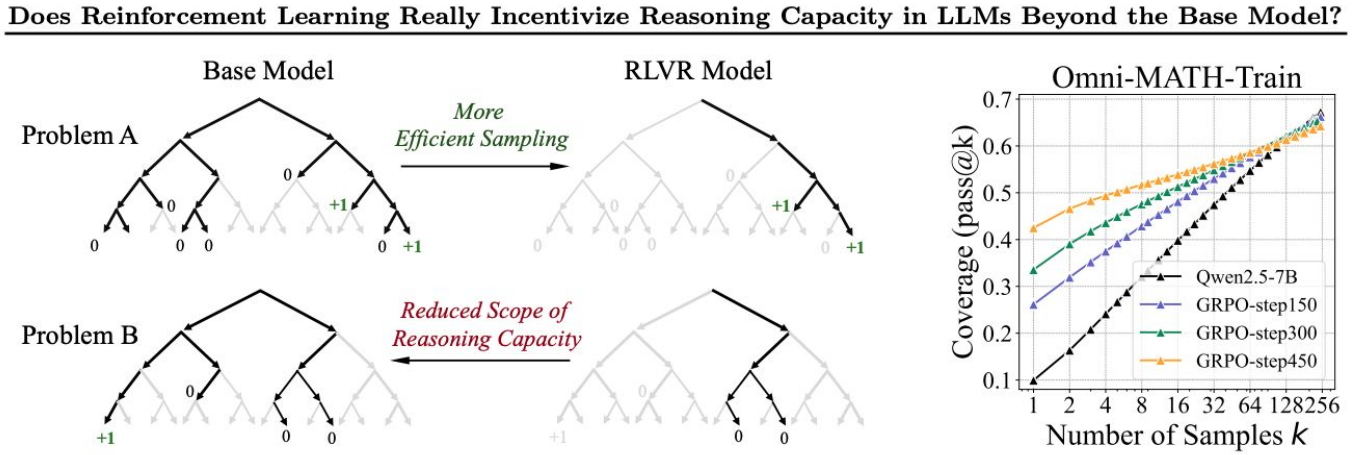


Figure 7 | The Maj@K and Pass@K of SFT and RL DeepSeekMath 7B on GSM8K and MATH (temperature 0.7). It was noted that RL enhances Maj@K but not Pass@K.

Discussion: Why RL works?

- RL improves sampling efficiency of the base model
- Instead of instilling new capabilities into the model



(Yue et al., 2025)

Discussion: Is GRPO Special?

- Yes! It is better than other RL algorithms?
- No… A strong base model is more important?

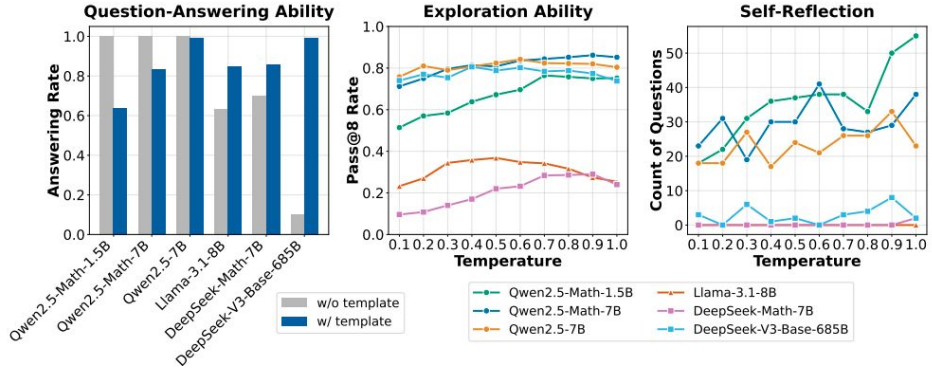


Figure 3: Model attributes across three aspects. **Question-Answering Ability**: the extent to which a pretrained language model provides a direct answer to a question rather than continuing or expanding upon it; **Exploration Ability**: pass@8 measures how well base models explore; **Self-Reflection**: counts are obtained through cross-validation between keyword-based detection and LLM-based detection, as detailed in Appendix C.

(Dr. GRPO; Liu et al., 2025)

Agentic Reasoning and Tool Integration for LLMs via Reinforcement Learning

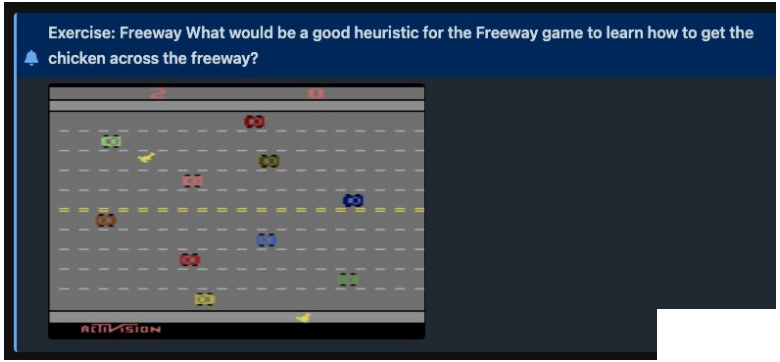
Joykirat Singh, Raghav Magazine, Yash Pandya, Akshay Nambi

2025

Discussion: What Else Matters?

- Learning data…
- Learning algorithms…
- How about supervision signal/reward?
  - Very important in RL!

Background: Reward Shaping



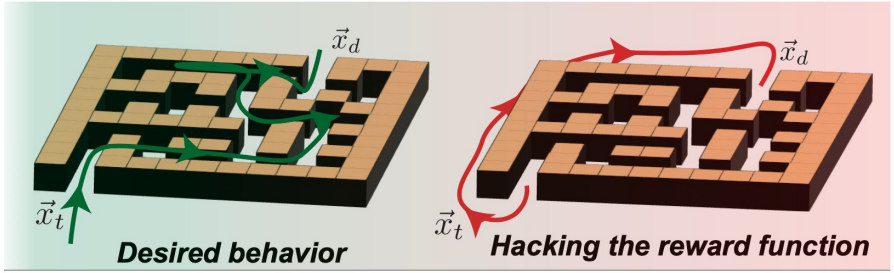
Sparse Rewards

4					+1k
3					
2					
1					
0	I				
	0	1	2	3	4

Dense Rewards

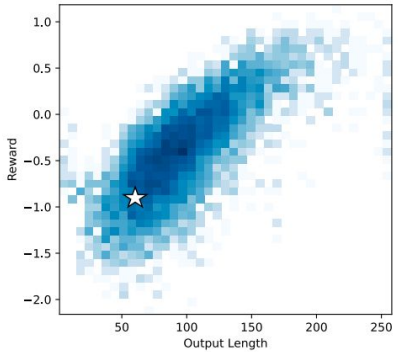
4					+1k
3	+3	⋰			
2	+2	+3	⋰		
1	+1	+2	+3	⋰	
0	I	+1	+2	+3	
	0	1	2	3	4

Background: Reward Hacking



$$r(s_t, a_t) = -\|\vec{x}_t - \vec{x}_d\|^2$$

(Reward is a form of "*Minimize distance to goal*")



Question: *Why don't adults roll off the bed?*

☆ **SFT (Before); 59 tokens**

Adults typically do not roll off of the bed because they have developed the muscle memory to keep their bodies from involuntarily moving during sleep and maintaining proper posture.

RLHF (After); 243 tokens: **Similar output, but much longer / more details**

Adults generally do not roll off of the bed because they have developed muscle control and awareness which helps them to remain upright while sleeping. Additionally, most adults find it uncomfortable or even painful to move around in their sleep, so rolling off the bed would be difficult without waking up first. There may also be safety concerns such as falling out of bed and potentially hurting oneself if one were to lose balance or fall from the ...

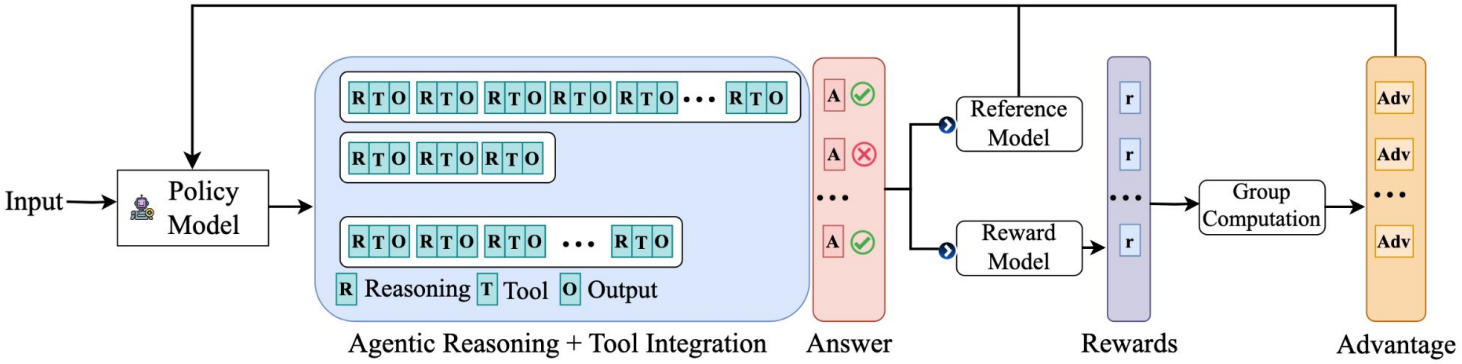
(Singhal et al., 2024)

Motivation

- What is the best reward design for LLM agents that can reason and use tools adaptively?
  - Simplest: Answer
  - A bit sophisticated: Answer and format (e.g., DeepSeek-R1)
  - How to train the agent for the adaptive strategy?

What to Learn

- Adaptive use of chain-of-thought reasoning and tool interactions
- Tasks
  - Mathematical reasoning
  - Multi-turn API calls



How to Learn: Reward

- Answer reward
- Format reward
  - Tags in order: reasoning, tool call, and tool output
  - Final answer tag is present
- Tool execution reward
  - Potential issue: discourage exploration

$$\text{Tool Execution Reward} = \frac{Tool_{success}}{Tool_{total}}$$

How to Learn: Reward

- Specific reward for API call task
  - State reward
    - State variables of the API call match the ground truth
  - Function reward
    - The number of function calls that match the ground truth calls
    - Final answer tag is present

$$R_{\text{state}} = SR_{\text{max}} \times \frac{State_{\text{match}}}{State_{\text{total}}} \qquad R_{\text{function}} = FR_{\text{max}} \times \frac{Functions_{\text{matched}}}{Functions_{\text{total}}}$$

How to Learn: GRPO

- Small Modification to GRPO
  - Tool outputs are masked from the loss computations

Results

Method	MATH-500	AIME	AMC	Olympiad
Frontier LLMs				
GPT-4o	0.630	0.080	0.430	0.290
Frontier Open-Source LLMs				
DeepSeek-R1-Distill-Qwen-7B	0.858	0.211	0.675	0.395
DeepSeek-R1	0.850	0.300	0.810	0.460
Open-Source Tool-Augmented LLMs				
NuminaMath-TIR-7B	0.530	0.060	0.240	0.190
ToRA-7B	0.410	0.000	0.070	0.130
ToRA-Code-7B	0.460	0.000	0.100	0.160
Qwen2.5-7B (PAL)	0.100	0.000	0.050	0.020
Base LLMs				
Qwen2.5-7B-Instruct	0.620	0.040	0.350	0.210
Qwen2.5-14B-Instruct	0.700	0.060	0.330	0.240
Base LLMs + Tools via Prompt				
Qwen2.5-7B-Instruct + Python Tool	0.629	0.122	0.349	0.366
Qwen2.5-14B-Instruct + Python Tool	0.671	0.100	0.410	0.371
ARTIST				
Qwen2.5-7B-Instruct + ARTIST	**0.676**	**0.156**	**0.470**	**0.379**
Qwen2.5-14B-Instruct + ARTIST	**0.726**	**0.122**	**0.550**	**0.420**

Table 1: Pass@1 accuracy on four mathematical reasoning benchmarks. ARTIST consistently outperforms all baselines, especially on complex tasks.

Benchmark / Method	τ bench		BFCL v3		
	Airline	Retail	Missing Function	Missing Parameters	Long Context
Frontier LLMs					
GPT-4o	0.460	0.604	0.410	0.355	0.545
Frontier Open-Source LLMs					
Llama-3-70B*	0.148	0.144	0.130	0.105	0.095
Deepseek-R1	–	–	0.155	0.110	0.115
Qwen2.5-72B-Instruct	–	–	0.245	0.200	0.155
Base LLMs					
Qwen2.5-7B-Instruct	0.120	0.180	0.085	0.060	0.040
Base LLMs + Reasoning via Prompt					
Qwen2.5-7B-Instruct + Reasoning	0.120	0.200	0.105	0.055	0.055
ARTIST					
Qwen2.5-7B-Instruct + ARTIST	**0.260**	**0.240**	**0.105**	**0.065**	**0.130**

ToolRL: Reward is All Tool Learning Needs

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, Heng Ji

2025

How to Learn: Reward

- Answer reward
- Format reward
  - All tags are present in the correct order
- Correctness reward
  - Tool name matching
  - Parameter name matching
  - Parameter content matching

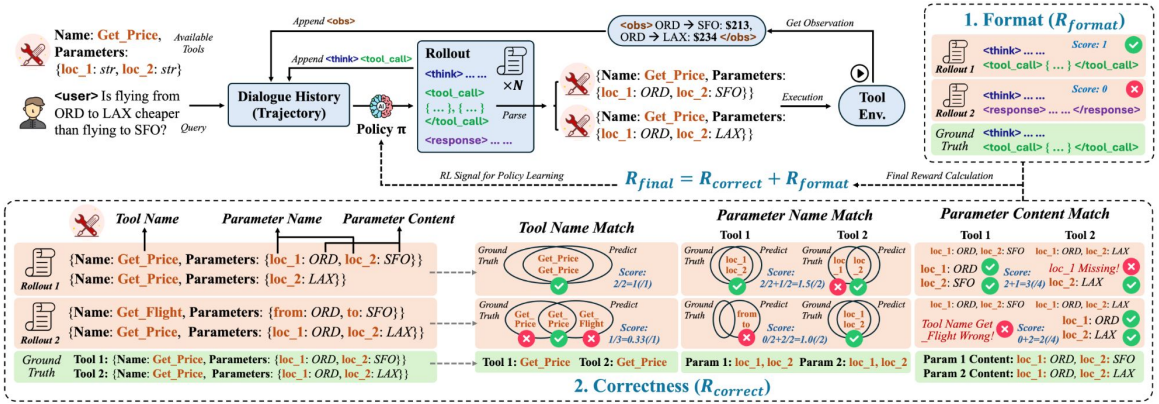
$$r_{\text{param}} = \sum_{G_j \in G} \frac{|\text{keys}(P_G) \cap \text{keys}(P_P)|}{|\text{keys}(P_G) \cup \text{keys}(P_P)|} \in [0, |G|]$$

$$r_{\text{name}} = \frac{|N_G \cap N_P|}{|N_G \cup N_P|} \in [0, 1]$$

$$r_{\text{value}} = \sum_{G_j \in G} \sum_{k \in \text{keys}(G_j)} \mathbb{1}[P_G[k] = P_P[k]] \in [0, \sum_{G_j \in G} |\text{keys}(G_j)|]$$

How to Learn: Reward

- Answer reward
- Format reward
  - All tags are present in the correct order
- Correctness reward
  - Tool name matching
  - Parameter name matching
  - Parameter content matching

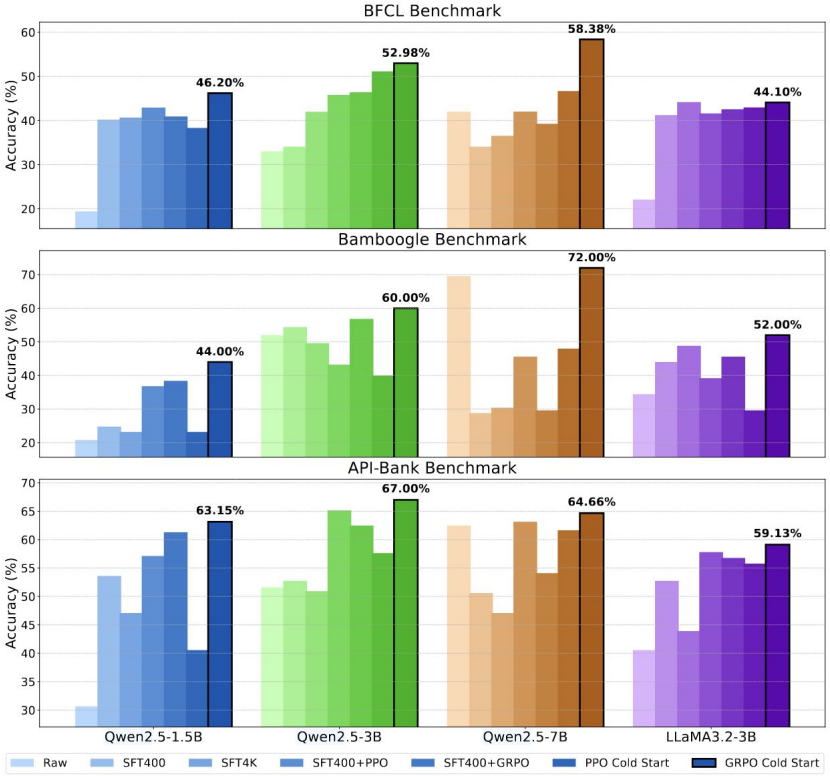


How to Learn: GRPO

- Small Modification to GRPO
  - KL term is omitted

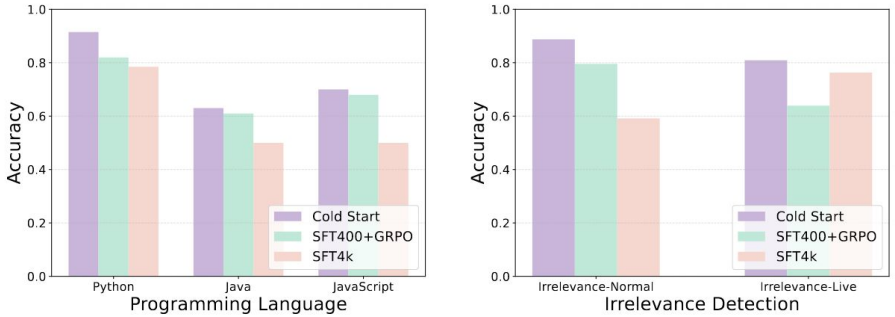
Results

- SFT is not necessary before RL for learning tool use



Results

- SFT fails to generalize to unseen scenario/goal

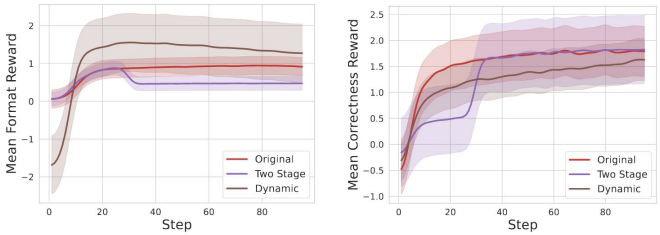


(a) Unfamiliar Scenario (b) Unfamiliar Goal

Figure 7: Qwen2.5-3B-Instruct's performance across unfamiliar programming language scenarios (left) and novel relevance detection task goals (right), evaluated under different training settings.

Results

- Reward scaling is very important



(a) Format Reward (b) Correctness Reward

Figure 10: Format (left) and correctness (right) reward trends across training steps for Qwen2.5-3B-Instruct with different reward scale dynamics.

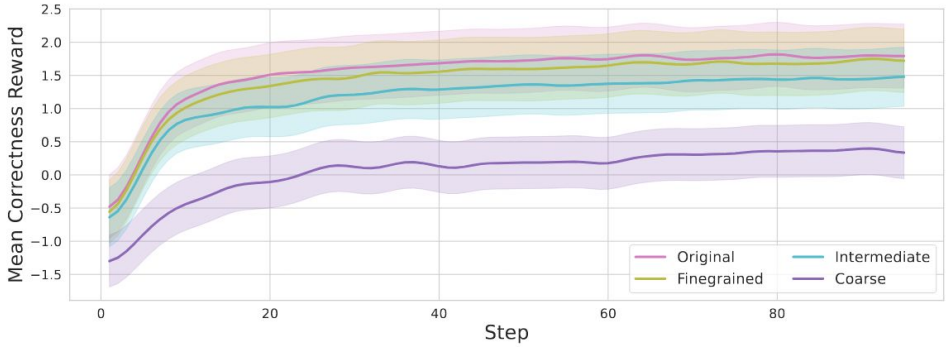


Figure 11: Correctness reward trends across training steps for Qwen2.5-3B-Instruct with different reward granularity.

Discussion: What’s Next?

Discussion: What to Learn

- Task-specific knowledge
  - How to balance the trade-off between forgetting the general and overfitting the specific?
- General knowledge
  - How to acquire general knowledge by interacting with environments and/or human/agent feedback?
    - Grounded learning as in Artzi and Zettlemoyer 2013?

Discussion: How to Learn

- Pretrained + SFT
  - Large scale data seems to have been exploited for this approach
  - How can we keep scaling up high-quality data with potentially generalizable knowledge/skills?
- RL
  - Is RL injecting new capabilities, or solely enhancing existing ones?

Thank you!