Large Language Models II

Philipp Kraehenbuehl

LLMs

- Architectures
- Generation, Instruction Tuning, RLHF, DPO, Tasks and Datasets
- Tool use and Structured Outputs
- Long Context and RAGs
- Structured Dialogues, Reflection
- Limitations of LLMs

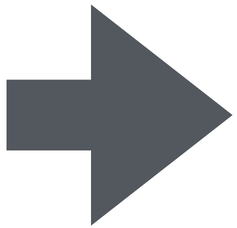
RLHF

Reinforcement Learning from Human Feedback

Philipp Krähenbühl, UT Austin

Instruction tuning

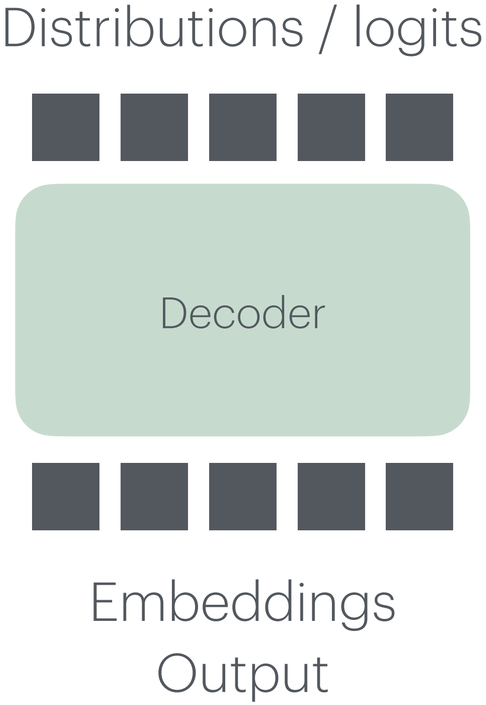
Pre-training

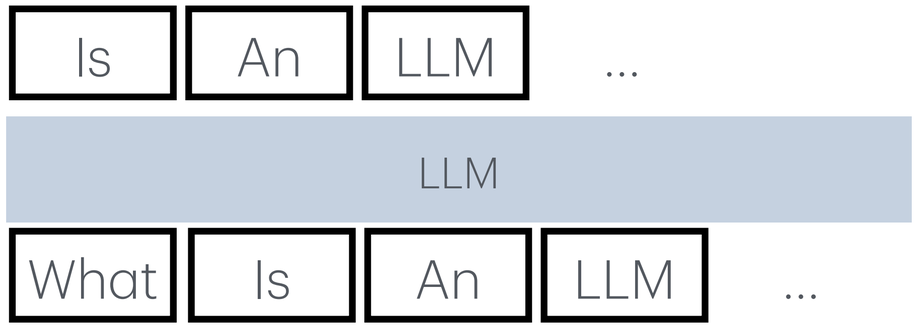


Instruction tuning

Language Models

- Decoder-only LLMs
  - Modeling auto-regressive distribution over tokens
  - $P(\mathbf{t}) = P(t_1)P(t_2 \mid t_1)P(t_3 \mid t_1, t_2)P(t_4 \mid t_1 \ldots t_3) \ldots$
- Generation / Sampling: $\mathbf{t} \sim P$





Instruction tuning

- Teach a LLM to follow a certain template in answering
  - Tell LLM **what to do**
  - Does **not** tell LLM **what not to do**

> I want to refocus my diet on only eating Apples and supplements.

```
Let me provide information on how to make the most of eating apples as a primary food source.

**Apple selection:** Focus on eating a variety of apples to ensure you get a range of nutrients. Some apple varieties are higher in certain nutrients than others. For example:

* Granny Smith and Braeburn apples have more malic acid, which can help with digestion.

* Gala and Fuji apples contain more vitamins A and C.

…
```

RLHF

Reinforcement Learning from Human Feedback

- Shape LLM outputs according to human preference / ranking

Training language models to follow instructions with human feedback. Ouyang etal 2022

RLHF

Reinforcement Learning from Human Feedback

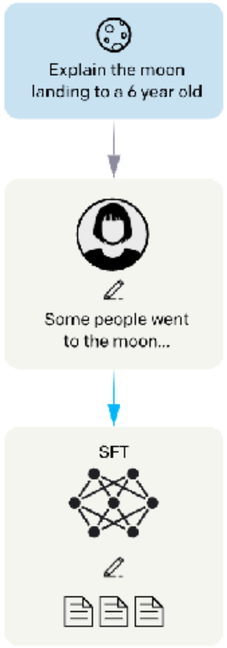
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



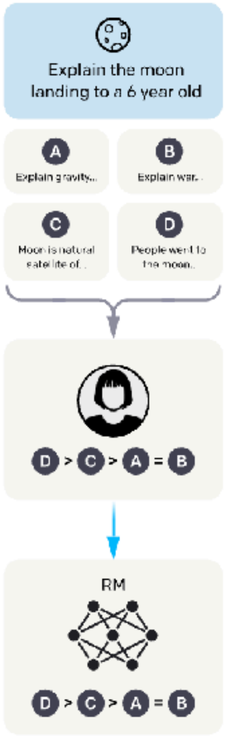
Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

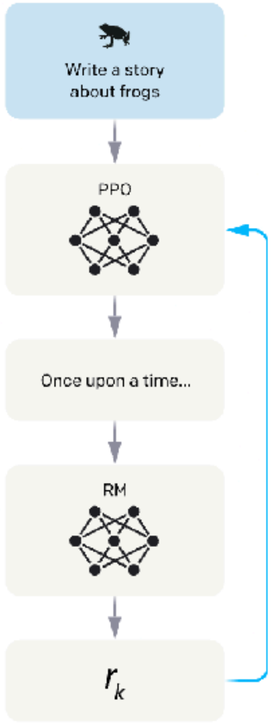
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

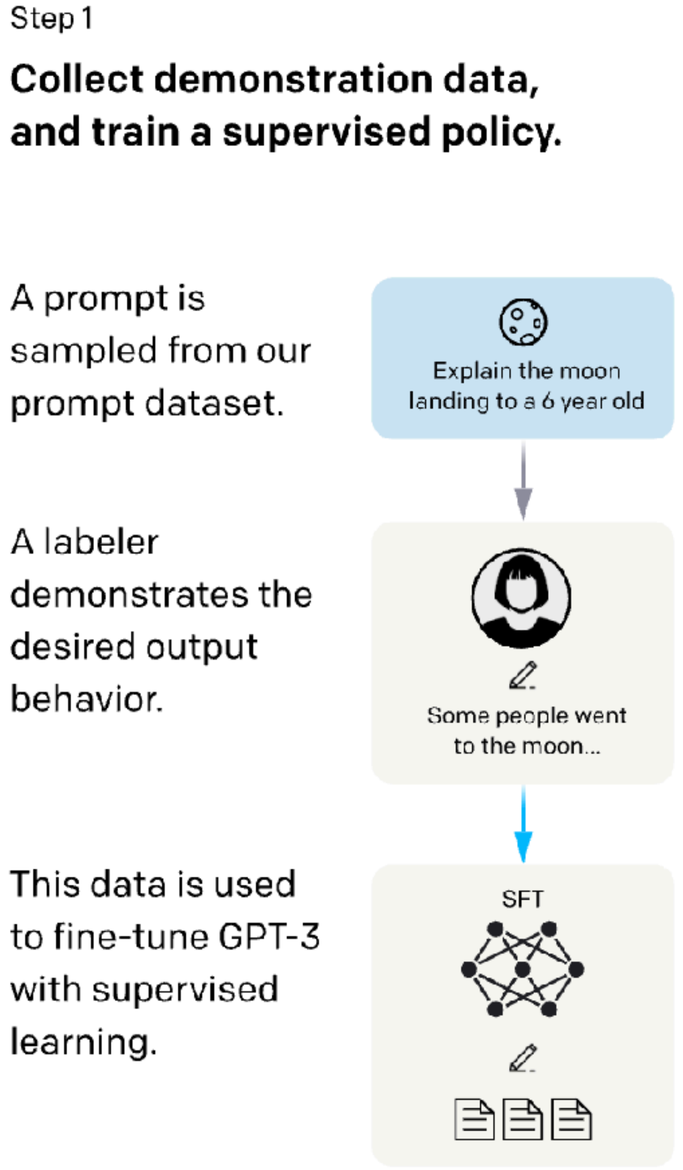


Training language models to follow instructions with human feedback. Ouyang etal 2022

RLHF

Reinforcement Learning from Human Feedback

- Step 1: Instruction tuning
  - Human labeler writes prompt
    - Plain, few-shot, customer-based
  - Human labeler writes answer
  - InstructGPT: 13k samples

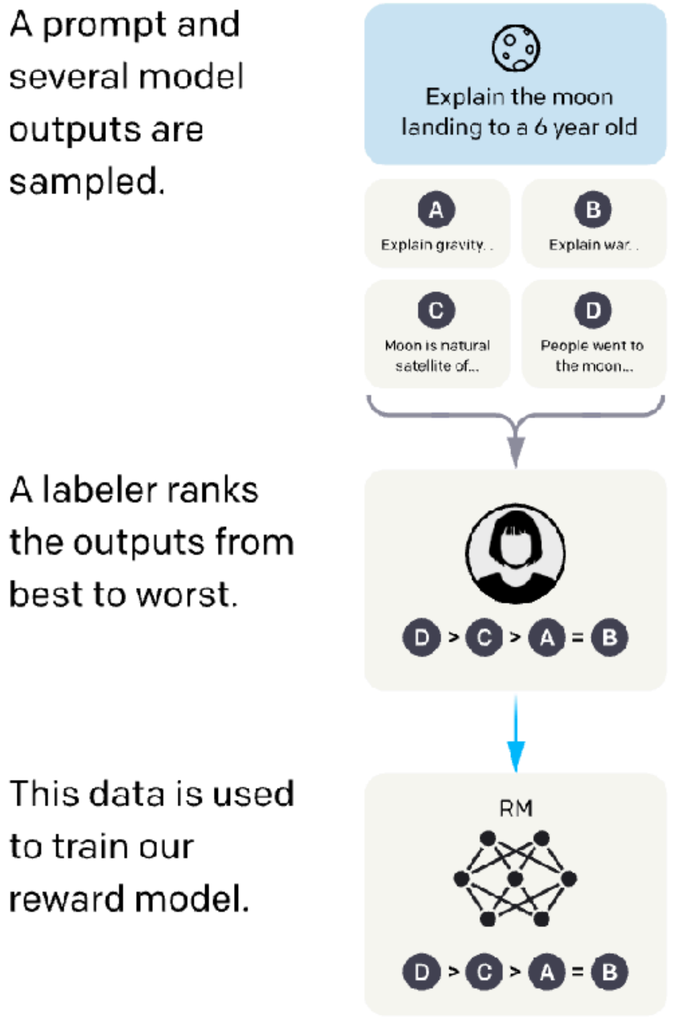


Training language models to follow instructions with human feedback. Ouyang etal 2022

RLHF

Reinforcement Learning from Human Feedback

- Step 2: Reward model learning
  - Human labeler writes prompt
    - Plain, few-shot, customer-based
  - Human labeler ranks ansers
  - InstructGPT: 33k samples (6.6k annotator,
    26.5k customer)



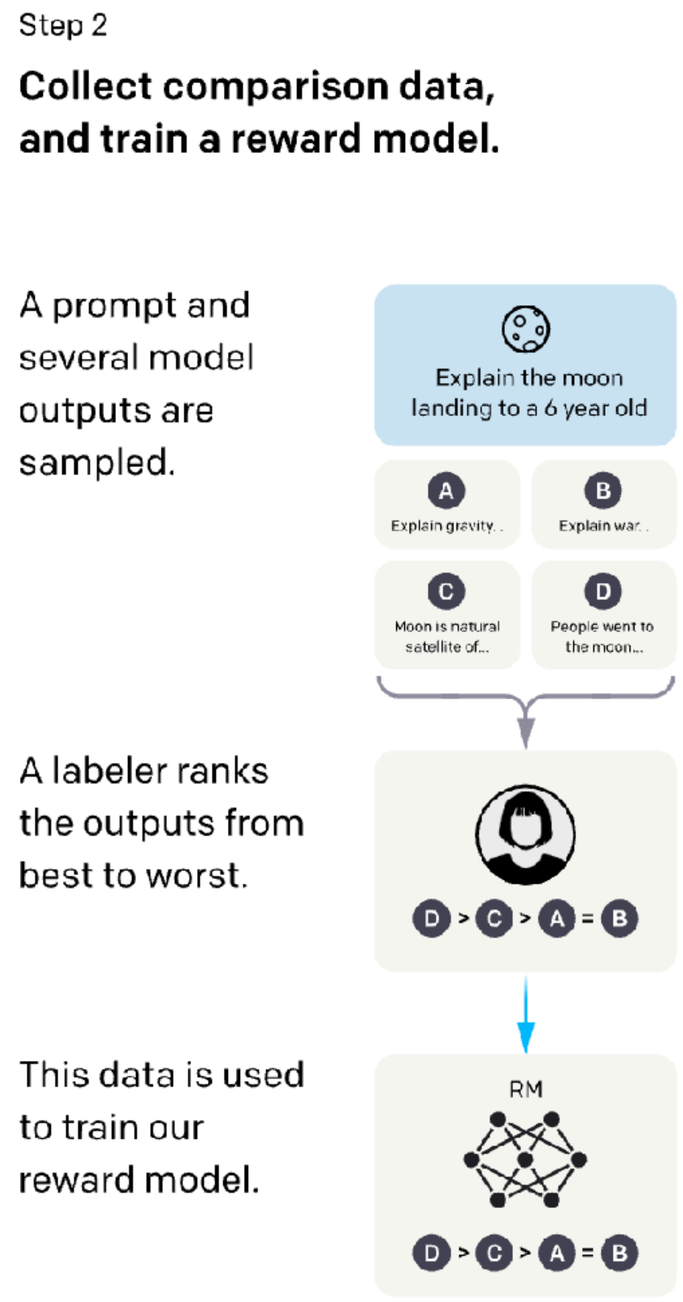
Training language models to follow instructions with human feedback. Ouyang etal 2022

RLHF

Reinforcement Learning from Human Feedback

- Step 2: Reward model learning
  - Train a small 6B reward model $r(x, y)$
    - LLM is 175B
  - Loss pairwise preference (Bradley-Terry model)

$$\ell = E_{x,y_+,y_-}\left[\log \sigma\left(r(x, y_+) - r(x, y_-)\right)\right]$$

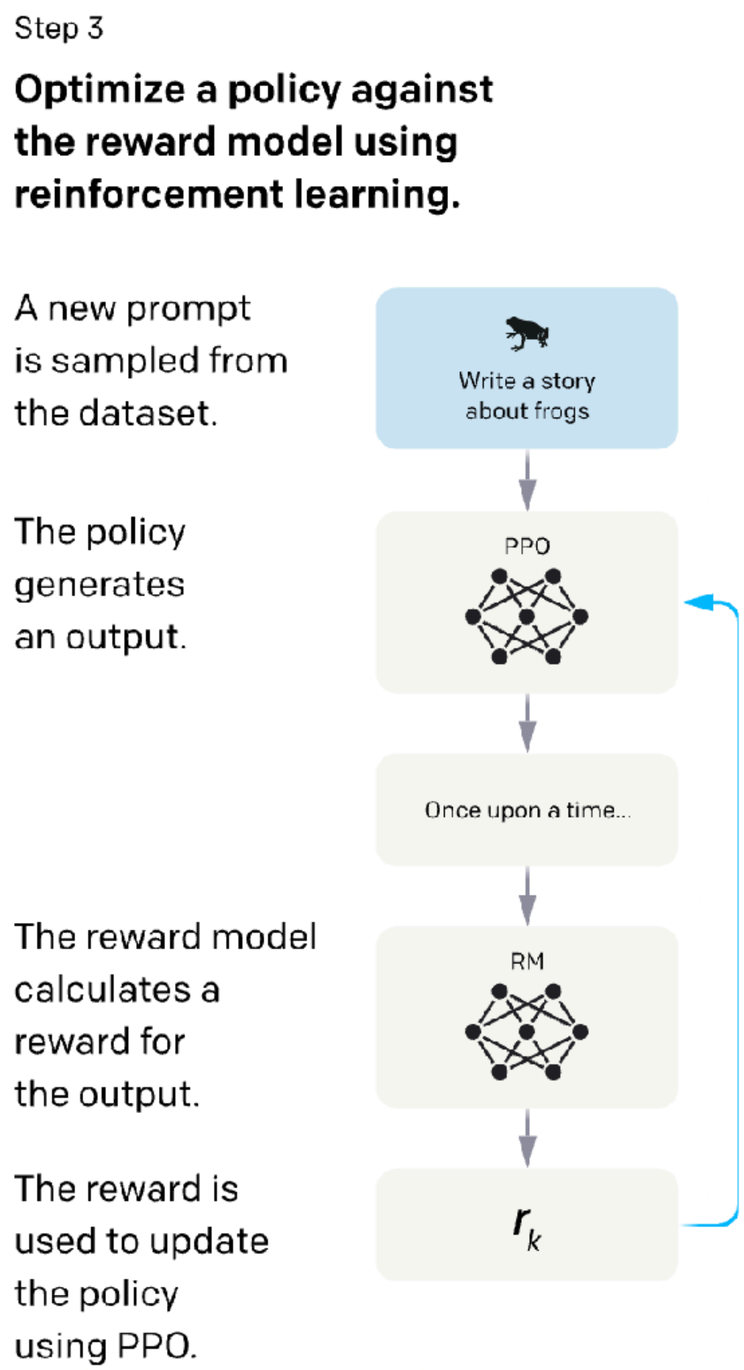


Training language models to follow instructions with human feedback. Ouyang etal 2022

RLHF

Reinforcement Learning from Human Feedback

- Step 3: Reinforcement Learning
  - Collect interesting prompts
  - InstructGPT: 32k samples (customer data)

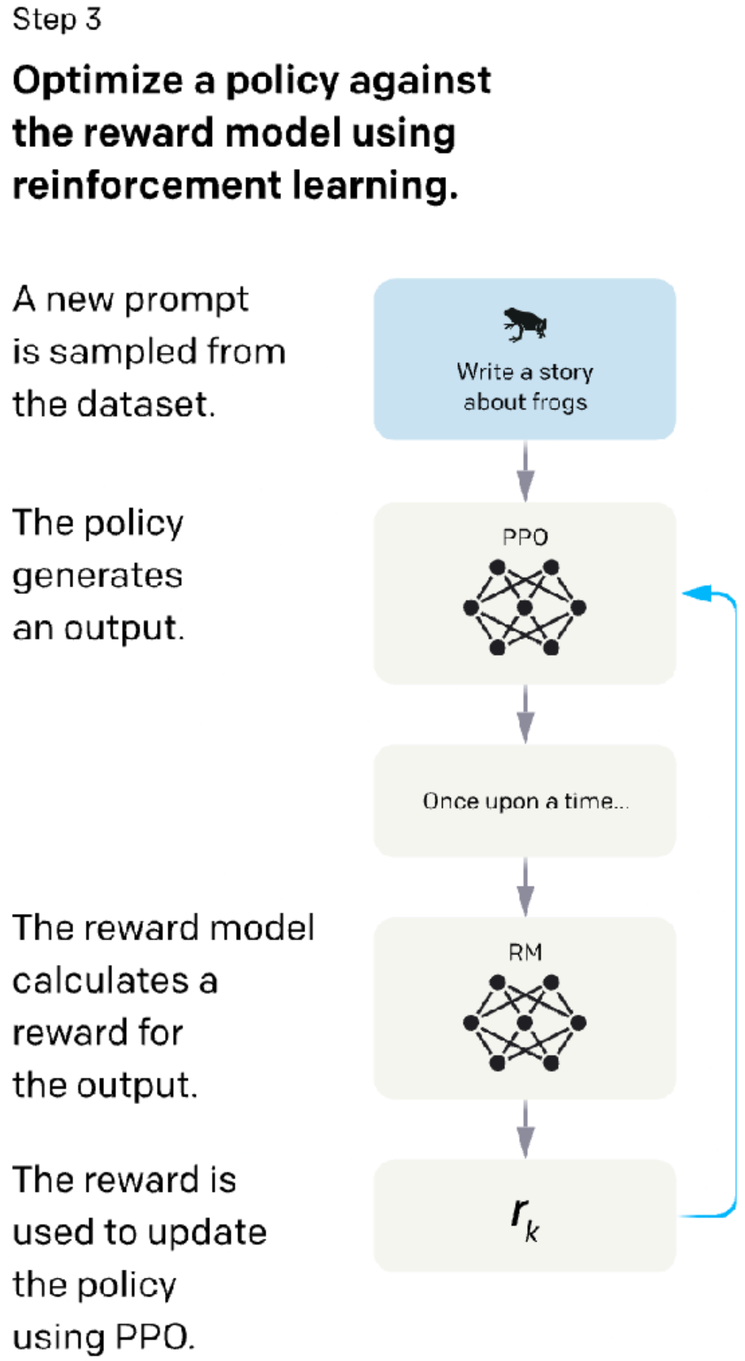


Training language models to follow instructions with human feedback. Ouyang etal 2022

RLHF

Reinforcement Learning from Human Feedback

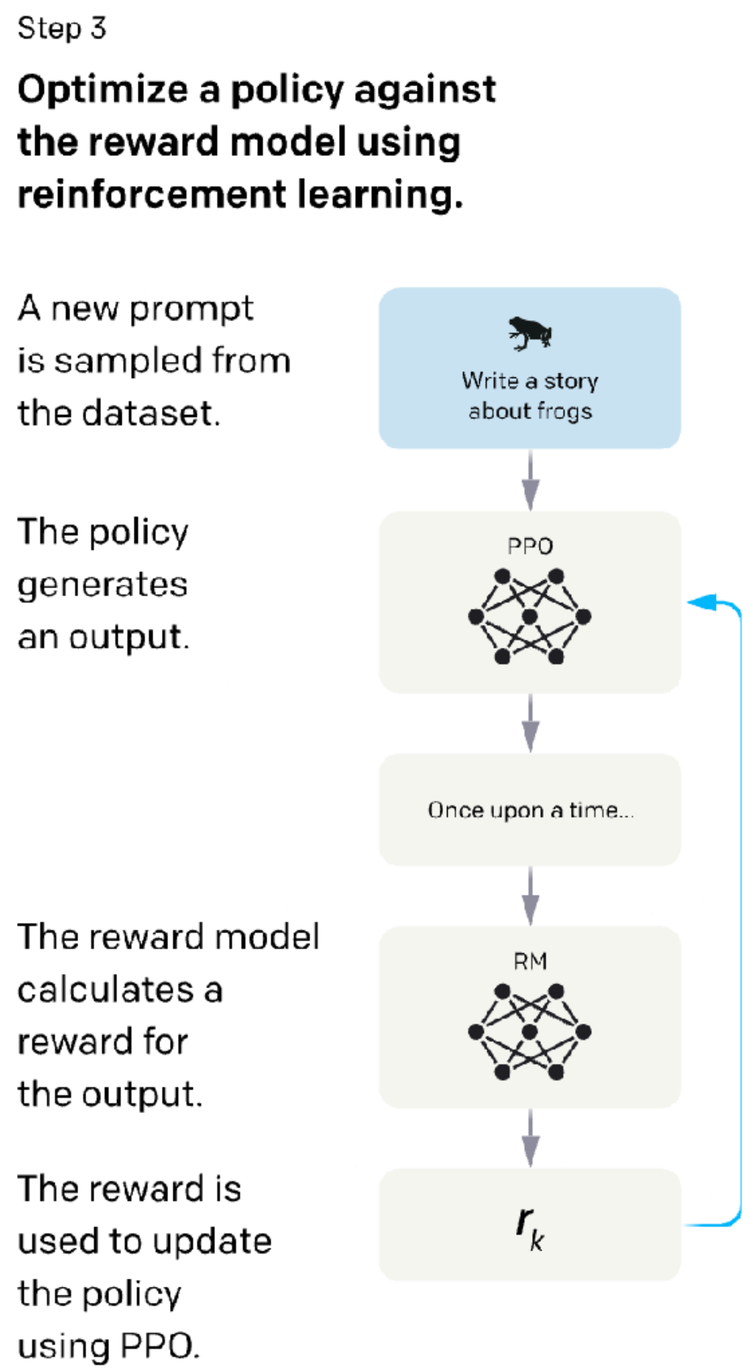
- Step 3: Reinforcement Learning
  - Fine-tune LLM to maximize reward model $r(x, y)$
  - PPO maximize:
    $E_{y\sim P(\cdot|x)}\left[(r(y,x))\,\nabla \log P(y\,|\,x)\right] - \beta D_{KL}\left[P(y\,|\,x)\,|\,P_{ref}(y\,|\,x)\right]$
    - Action = predict next token
    - Requires 4 models: Reference, generator, critic, reward



Training language models to follow instructions with human feedback. Ouyang etal 2022

Why use reinforcement learning?

- Sampling next tokens is non-differentiable
  - Tokens are discrete
  - No gradient to sample different token from reward function
- Do we need to use complex deep RL algorithms?



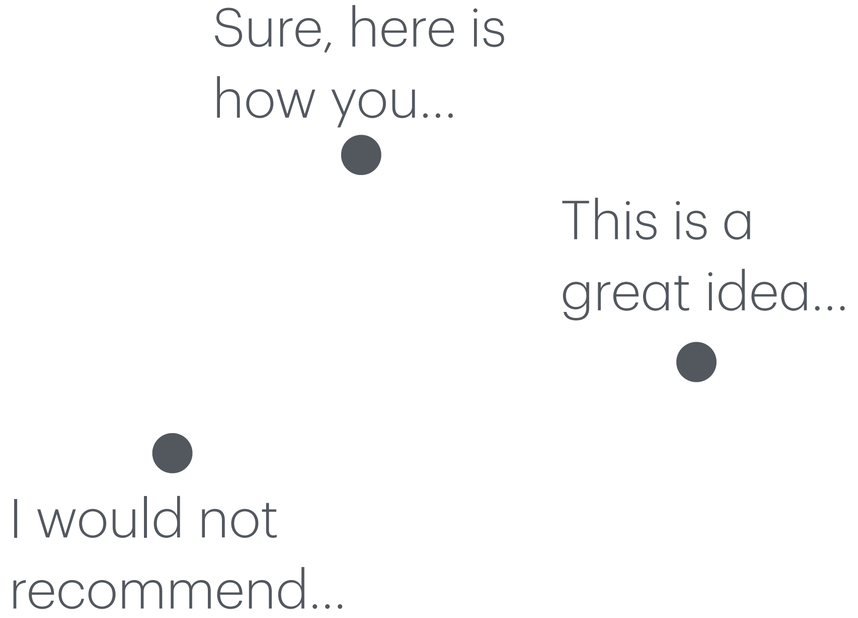
RLHF

Reinforcement Learning from Human Feedback

- Step 3: RLOO
  - Let’s treat RLHF as a bandit problem
    - No sequential actions
    - Action = generate a full response
  - Reinforce:
    $$E_{y\sim P(\cdot|x)}\left[(r(y,x)-b)\,\nabla \log P(y\,|\,x)\right]$$

Prompt:

> I want to refocus my diet on only eating Apples and supplements.



Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs. Ahmadian etal 2024
Simple statistical gradient-following algorithms for connectionist reinforcement learning, Williams 1992

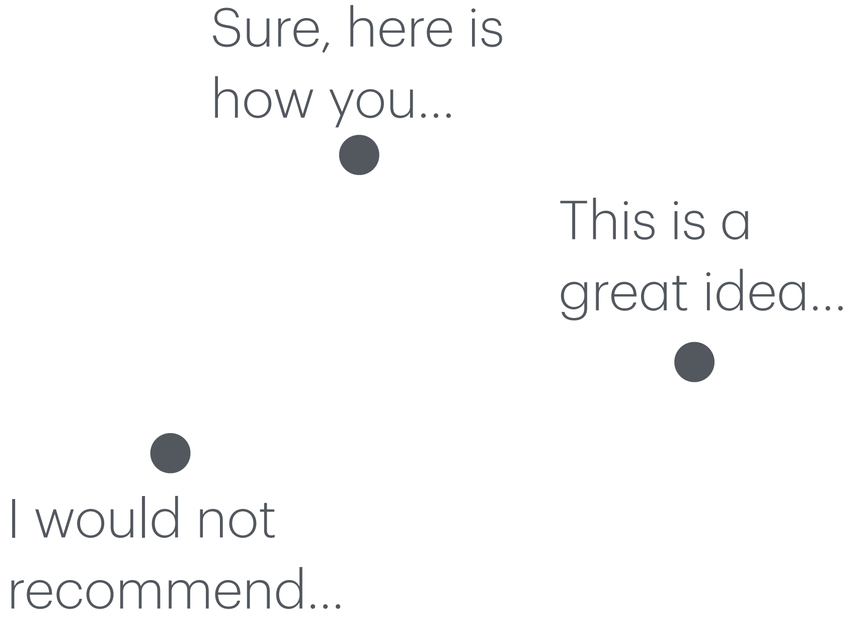
RLHF

Reinforcement Learning from Human Feedback

- Step 3: RLOO
  - N samples:
    $y_1, \ldots, y_N \sim P(\cdot \mid x)$
  - Reinforce:
    $$\sum_{i=1}^{N} \left[(R(y_i, x) - b_i) \nabla \log P(y_i \mid x) \right]$$
  - Baseline
    $$b_i = \frac{1}{N-1} \sum_{j \neq i} R(y_j, x)$$

Prompt:

> I want to refocus my diet on only eating Apples and supplements.

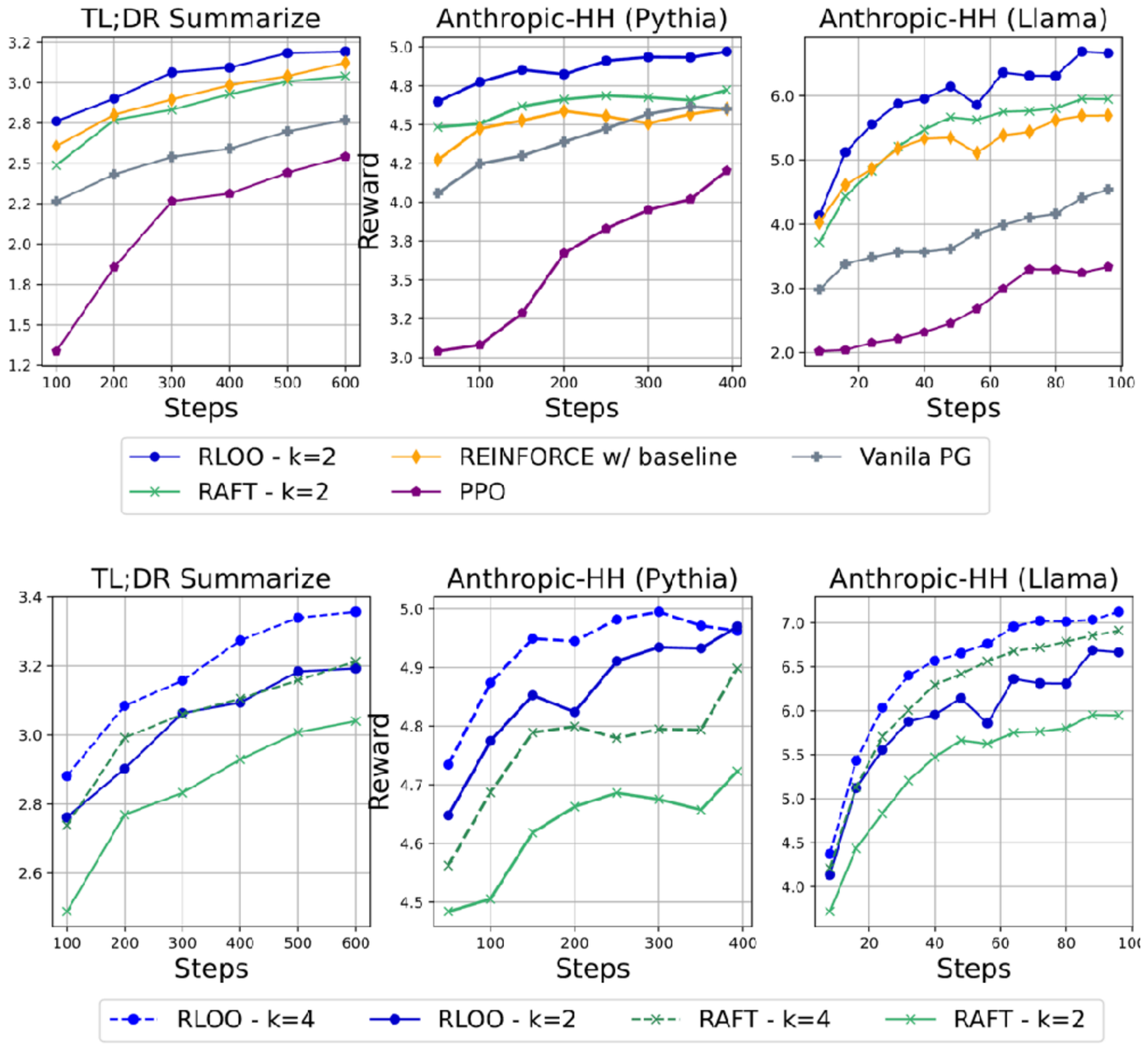


Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs. Ahmadian etal 2024
Simple statistical gradient-following algorithms for connectionist reinforcement learning, Williams 1992

RLHF

Reinforcement Learning from Human Feedback

- Step 3: RLOO
  - Light weight
  - Requires
    - Generator
    - Reward model



Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs. Ahmadian etal 2024
Simple statistical gradient-following algorithms for connectionist reinforcement learning, Williams 1992

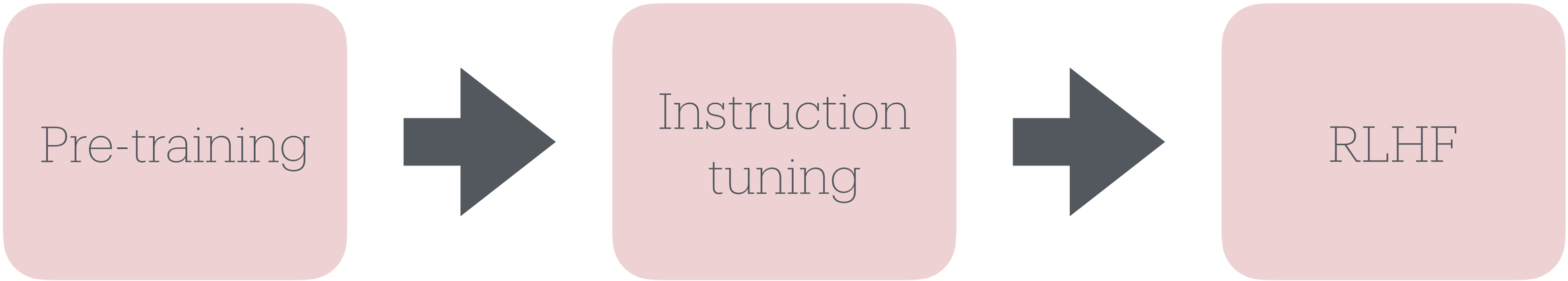
RLHF

Reinforcement Learning from Human Feedback

- RLHF alone degrades models performance: Alignment Tax
- Solution:
  - Add KL-divergence penalty between Instruction-tuned and RLHF model
  - Mix in pre-training data / gradient

Training language models to follow instructions with human feedback. Ouyang etal 2022

RLHF



RLHF

- Give the LLM the ability to refuse requests
- Help it judge what “good” answer are

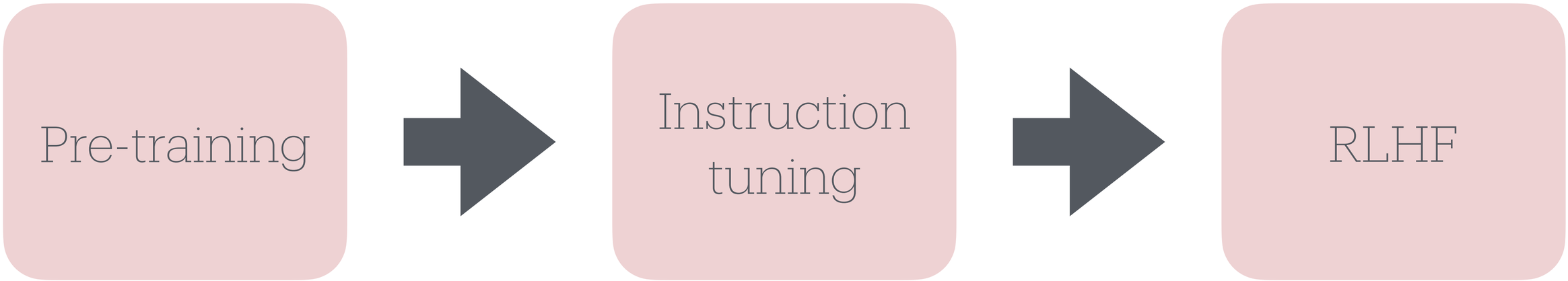
> I want to refocus my diet on only eating Apples and supplements.

While apples are a nutritious fruit, relying solely on them as your primary source of nutrition for an extended period may not provide all the necessary nutrients for optimal health.

Here are some concerns with a strictly apple-based diet:

…

RLHF



References

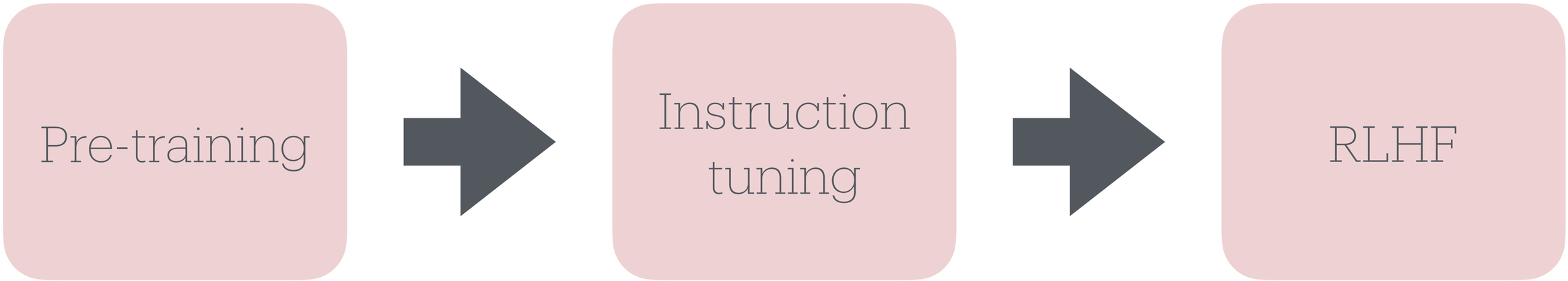
- [1] Training language models to follow instructions with human feedback. Ouyang etal 2022.
- [2] Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs. Ahmadian etal 2024.
- [3] Simple statistical gradient-following algorithms for connectionist reinforcement learning, Williams 1992.

DPO

Direct Preference Optimization

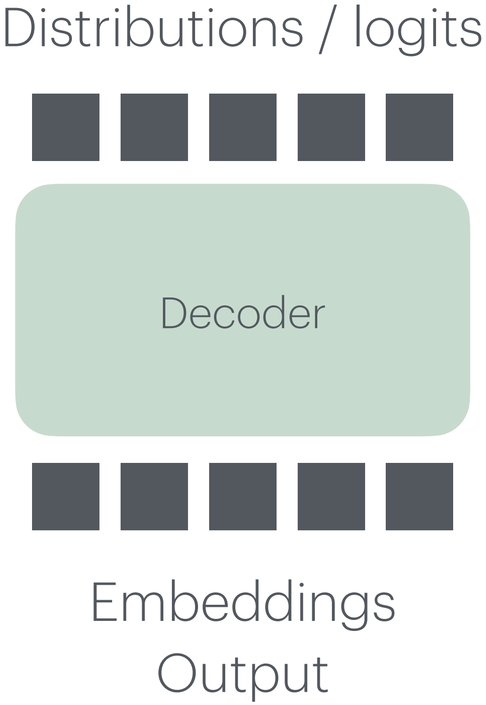
Philipp Krähenbühl, UT Austin

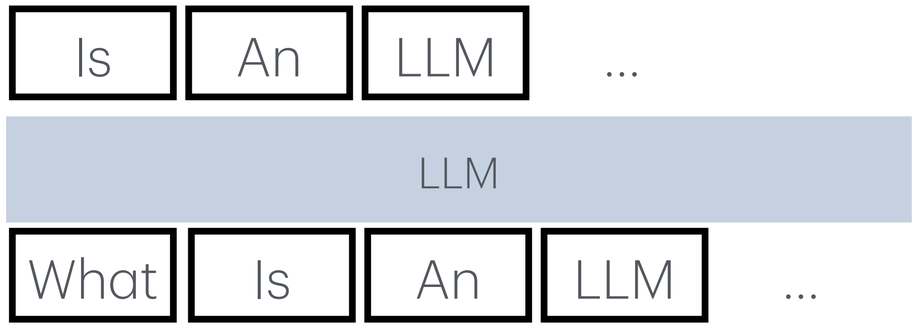
RLHF



Language Models

- Decoder-only LLMs
  - Modeling auto-regressive distribution over tokens
  - $P(\mathbf{t}) = P(t_1)P(t_2 \mid t_1)P(t_3 \mid t_1, t_2)P(t_4 \mid t_1 \ldots t_3) \ldots$
- Generation / Sampling: $\mathbf{t} \sim P$





Instruction tuning

- Teach a LLM to follow a certain template in answering
  - Tell LLM **what to do**
  - Does **not** tell LLM **what not to do**

> I want to refocus my diet on only eating Apples and supplements.

```
Let me provide information on how to make the most of eating apples as a primary food source.

**Apple selection:** Focus on eating a variety of apples to ensure you get a range of nutrients. Some apple varieties are higher in certain nutrients than others. For example:

* Granny Smith and Braeburn apples have more malic acid, which can help with digestion.

* Gala and Fuji apples contain more vitamins A and C.

…
```

RLHF

Reinforcement Learning from Human Feedback

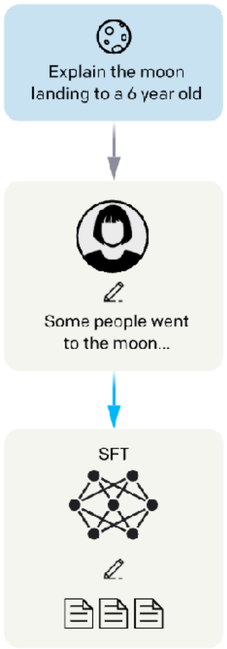
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



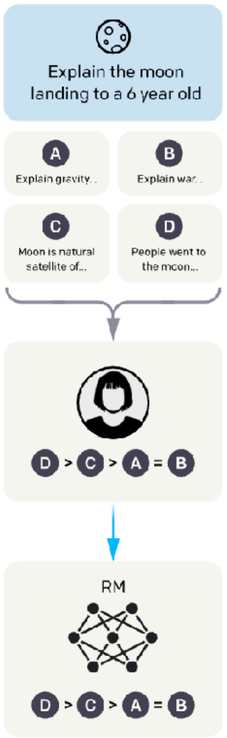
Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

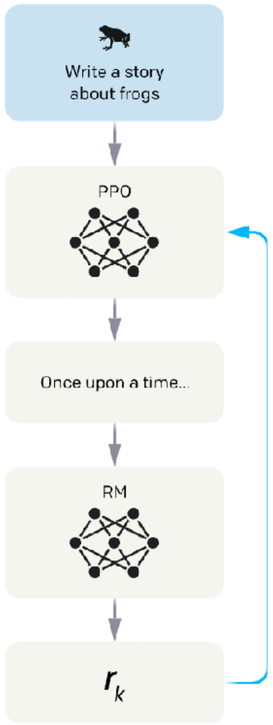
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



Training language models to follow instructions with human feedback. Ouyang etal 2022

RLHF - a recap

- Learn reward: $\ell = E_{x,y_+,y_-}\left[\log \sigma\left(r(x,y_+) - r(x,y\_)\right)\right]$

- Optimize: $E_{y\sim P(\cdot|x)}\left[(r(y,x))\,\nabla \log P(y\,|\,x)\right] - \beta D_{KL}\left[P(y\,|\,x)\,|\,P_{ref}(y\,|\,x)\right]$

DPO

- Learn reward: $\ell = E_{x,y_+,y_-}\left[\log \sigma\left(r(x,y_+) - r(x,y\_)\right)\right]$

- Optimize: $E_{y\sim P(\cdot|x)}\left[(r(y,x))\,\nabla \log P(y\,|\,x)\right] - \beta D_{KL}\left[P(y\,|\,x)\,|\,P_{ref}(y\,|\,x)\right]$

  - Closed form solution: $P(y\,|\,x) = \frac{1}{Z(x)} P_{ref}(y\,|\,x)\exp\left(\frac{1}{\beta} r(x,y)\right)$

Direct Preference Optimization: Your Language Model is Secretly a Reward Model, Rafailov etal 2023

DPO

- Learn reward: $\ell = E_{x,y_+,y_-}\left[\log \sigma\left(r(x,y_+) - r(x,y_-)\right)\right]$

- Optimize: $E_{y\sim P(\cdot|x)}\left[(r(y,x))\,\nabla \log P(y\,|\,x)\right] - \beta D_{KL}\left[P(y\,|\,x)\,|\,P_{ref}(y\,|\,x)\right]$

  - Closed form solution: $P(y\,|\,x) = \frac{1}{Z(x)} P_{ref}(y\,|\,x)\exp\left(\frac{1}{\beta} r(x,y)\right)$

  - $r(x,y) = \beta \frac{P(y\,|\,x)}{P_{ref}(y\,|\,x)} + \beta \log Z(x)$

Direct Preference Optimization: Your Language Model is Secretly a Reward Model, Rafailov etal 2023

DPO

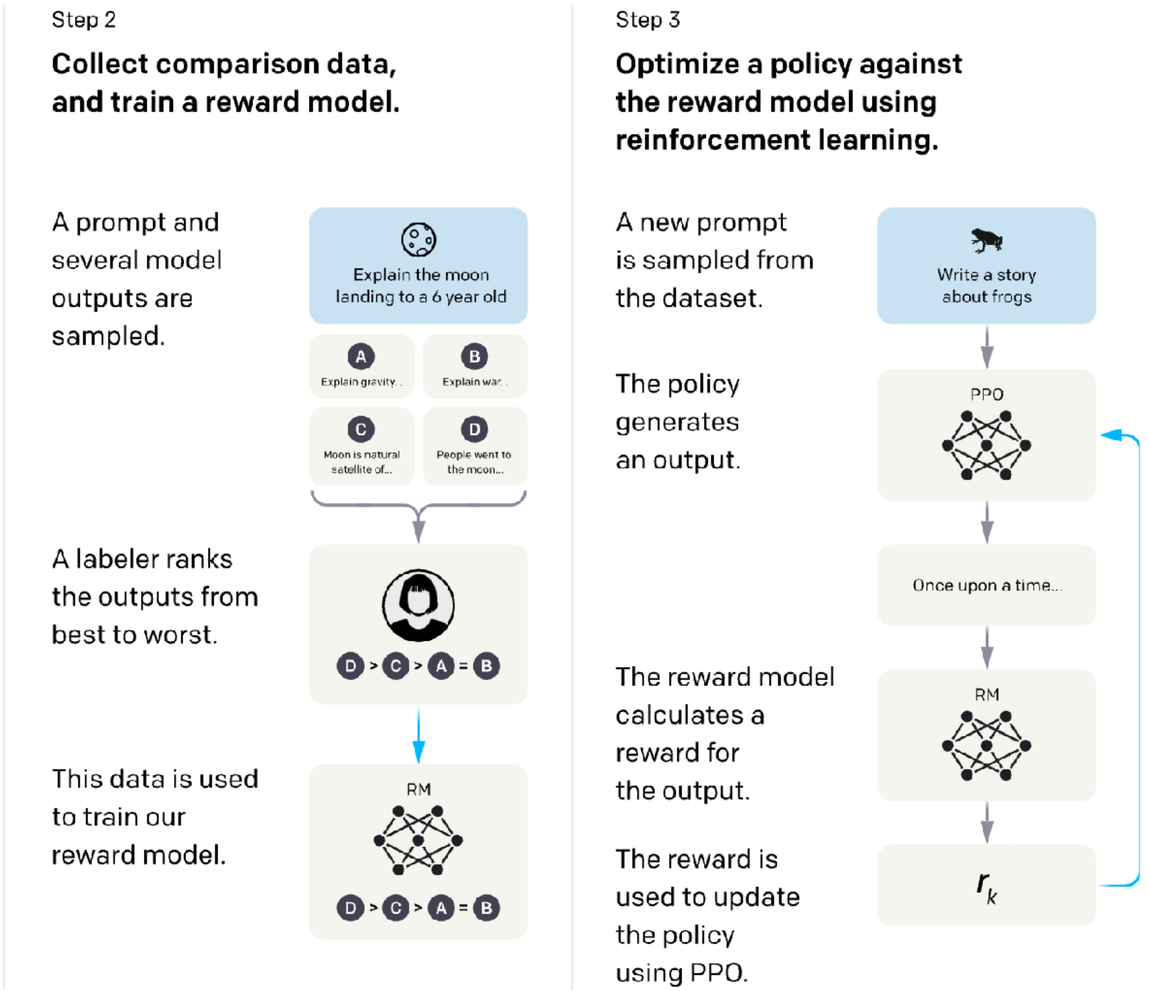
- Learn reward: $\ell = E_{x,y_+,y_-}\left[\log \sigma\left(r(x,y_+) - r(x,y_-)\right)\right]$
  - Closed form $\ell_{DPO} = E_{x,y_+,y_-}\left[\log \sigma\left(\beta \frac{rP(x,y_+)}{P_{ref}(x,y_+)} - \beta \frac{rP(x,y_-)}{P_{ref}(x,y_-)}\right)\right]$
- Optimize: $E_{y\sim P(\cdot|x)}\left[(r(y,x))\nabla \log P(y\,|\,x)\right] - \beta D_{KL}\left[P(y\,|\,x)\,|\,P_{ref}(y\,|\,x)\right]$
  - Closed form solution: $P(y\,|\,x) = \frac{1}{Z(x)} P_{ref}(y\,|\,x)\exp\left(\frac{1}{\beta} r(x,y)\right)$
  - $r(x,y) = \beta \frac{P(y\,|\,x)}{P_{ref}(y\,|\,x)} + \beta \log Z(x)$

Direct Preference Optimization: Your Language Model is Secretly a Reward Model, Rafailov etal 2023

DPO

$$\ell_{DPO} = E_{x,y_+,y_-}\left[\log \sigma\left(\beta \frac{rP(x,y_+)}{P_{ref}(x,y_+)} - \beta \frac{rP(x,y_-)}{P_{ref}(x,y_-)}\right)\right]$$

- Closed form solution to reward models + RL
  - Supervised learning
  - Easy to implement
  - Efficient

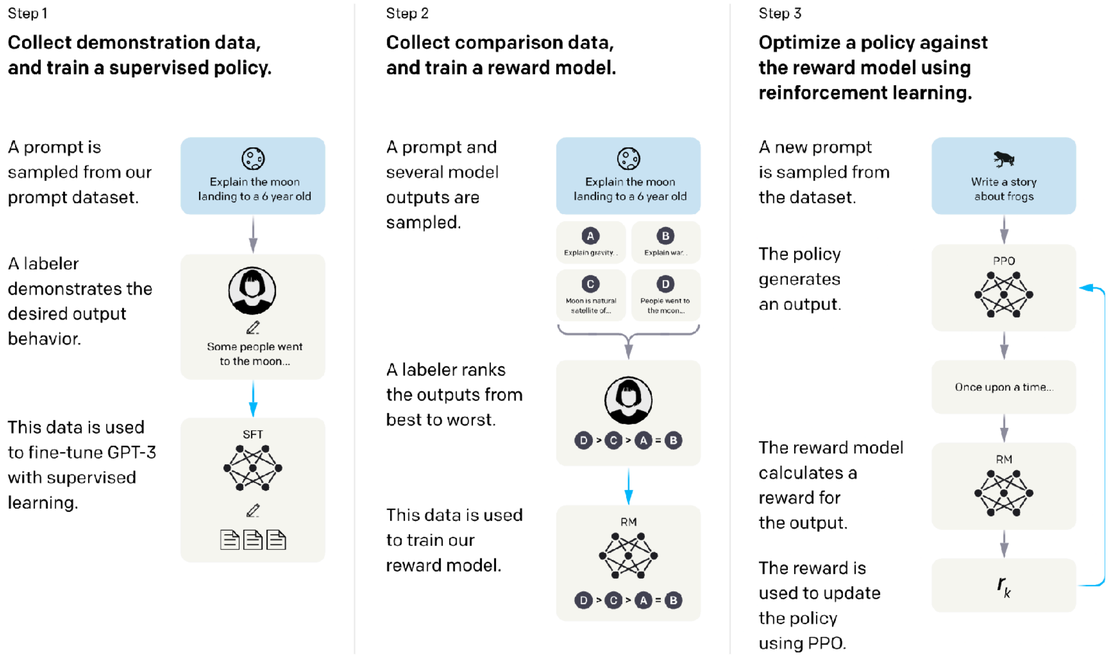


Direct Preference Optimization: Your Language Model is Secretly a Reward Model, Rafailov etal 2023

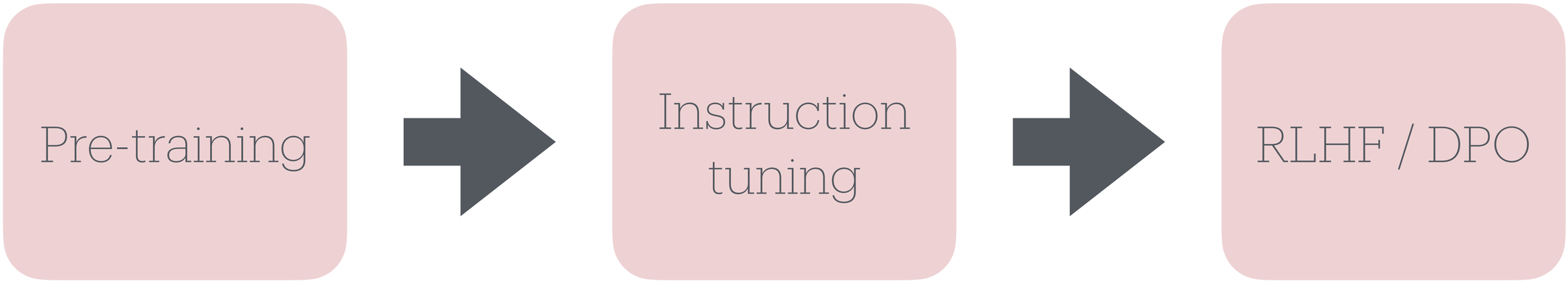
DPO vs RLHF

$$\ell_{DPO} = E_{x,y_+,y_-}\left[\log \sigma\left(\beta \frac{rP(x,y_+)}{P_{ref}(x,y_+)} - \beta \frac{rP(x,y_-)}{P_{ref}(x,y_-)}\right)\right]$$

- DPO
  - Easier to make work
  - Can only learn on preference data
  - Generally produces long outputs
- RLHF
  - Requires quite a bit of RL knowledge
  - Higher ceiling (can use smaller preference data, larger fine-tuning data)



Full Picture



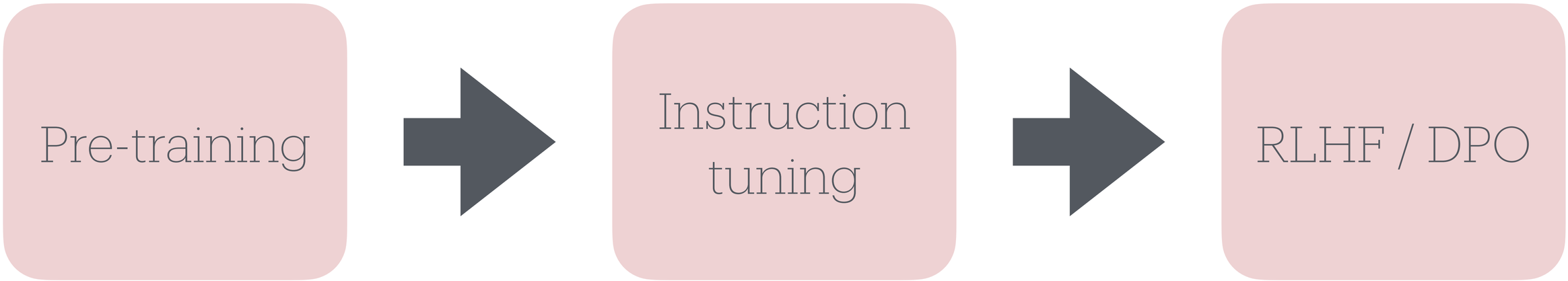
References

- [1] Training language models to follow instructions with human feedback. Ouyang etal 2022.
- [2] Direct Preference Optimization: Your Language Model is Secretly a Reward Model, Rafailov etal 2023.

Tasks and Datasets

Philipp Krähenbühl, UT Austin

Full Picture



Dataset categories

- Text understanding
- Programming
- Safety

Text Understanding

- Reading comprehension
- Commonsense reasoning
- World knowledge
- Symbolic problem solving
- Language understanding
- Mixed evaluation

Reading comprehension

Question → Answer

Question → Answer

Question → Answer

Question → Answer

- Input:
  - Text document
  - Question
- Output:
  - Answer
- Requires no external knowledge

Reading comprehension

Example: DROP

- Paragraph + Question -> Short answer
- Dozens of similar benchmarks
  - SQuAD, QuAC, CoQA, BoolQ, NaturalQuestions
  - Most developed pre LLM
- Evaluation can be tricky

Reasoning	**Passage** (some parts shortened)	**Question**	**Answer**
Subtraction (28.8%)	That year, his **Untitled (1981)**, a painting of a haloed, black-headed man with a bright red skeletal body, depicted amid the artists signature scrawls, was **sold by Robert Lehrman for $16.3 million, well above its $12 million high estimate**.	How many more dollars was the Untitled (1981) painting sold for than the 12 million dollar estimation?	4300000
Comparison (18.2%)	In **1517, the seventeen-year-old King sailed to Castile**. There, his Flemish court .... **In May 1518, Charles traveled to Barcelona in Aragon**.	Where did Charles travel to first, Castile or Barcelona?	Castile
Selection (19.4%)	In 1970, to commemorate the 100th anniversary of the founding of Baldwin City, **Baker University professor and playwright Don Mueller and Phyllis E. Braun, Business Manager, produced a musical play entitled The Ballad Of Black Jack** to tell the story of the events that led up to the battle.	Who was the University professor that helped produce The Ballad Of Black Jack, Ivan Boyd or Don Mueller?	Don Mueller
Addition (11.7%)	Before the UNPROFOR fully deployed, the HV clashed with an armed force of the RSK in the village of Nos Kalik, located in a pink zone near Šibenik, and captured the village at 4:45 p.m. on **2 March 1992**. The JNA formed a battlegroup to counterattack the **next day**.	What date did the JNA form a battlegroup to counterattack after the village of Nos Kalik was captured?	3 March 1992
Count (16.5%) and Sort (11.7%)	Denver would retake the lead with kicker **Matt Prater nailing a 43-yard field goal**, yet Carolina answered as kicker **John Kasay ties the game with a 39-yard field goal**. ... Carolina closed out the half with **Kasay nailing a 44-yard field goal**. ... In the fourth quarter, Carolina sealed the win with **Kasay's 42-yard field goal**.	Which kicker kicked the most field goals?	John Kasay
Coreference Resolution (3.7%)	**James Douglas** was the second son of Sir George Douglas of Pittendreich, and Elizabeth Douglas, daughter David Douglas of Pittendreich. Before **1543 he married Elizabeth**, daughter of James Douglas, 3rd Earl of Morton. **In 1553 James Douglas succeeded to the title and estates of his father-in-law**.	How many years after he married Elizabeth did James Douglas succeed to the title and estates of his father-in-law?	10
Other Arithmetic (3.2%)	Although the movement initially gathered some **60,000 adherents**, the subsequent establishment of the Bulgarian Exarchate **reduced their number by some 75%**.	How many adherents were left after the establishment of the Bulgarian Exarchate?	15000
Set of spans (6.0%)	According to some sources 363 civilians were killed in **Kavadarci**, 230 in **Negotino** and 40 in **Vatasha**.	What were the 3 villages that people were killed in?	Kavadarci, Negotino, Vatasha
Other (6.8%)	This **Annual Financial Report** is our principal financial statement of accountability. The **AFR gives a comprehensive view** of the Department's financial activities ...	What does AFR stand for?	Annual Financial Report

DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs, Dua etal 2019

Common sense reasoning

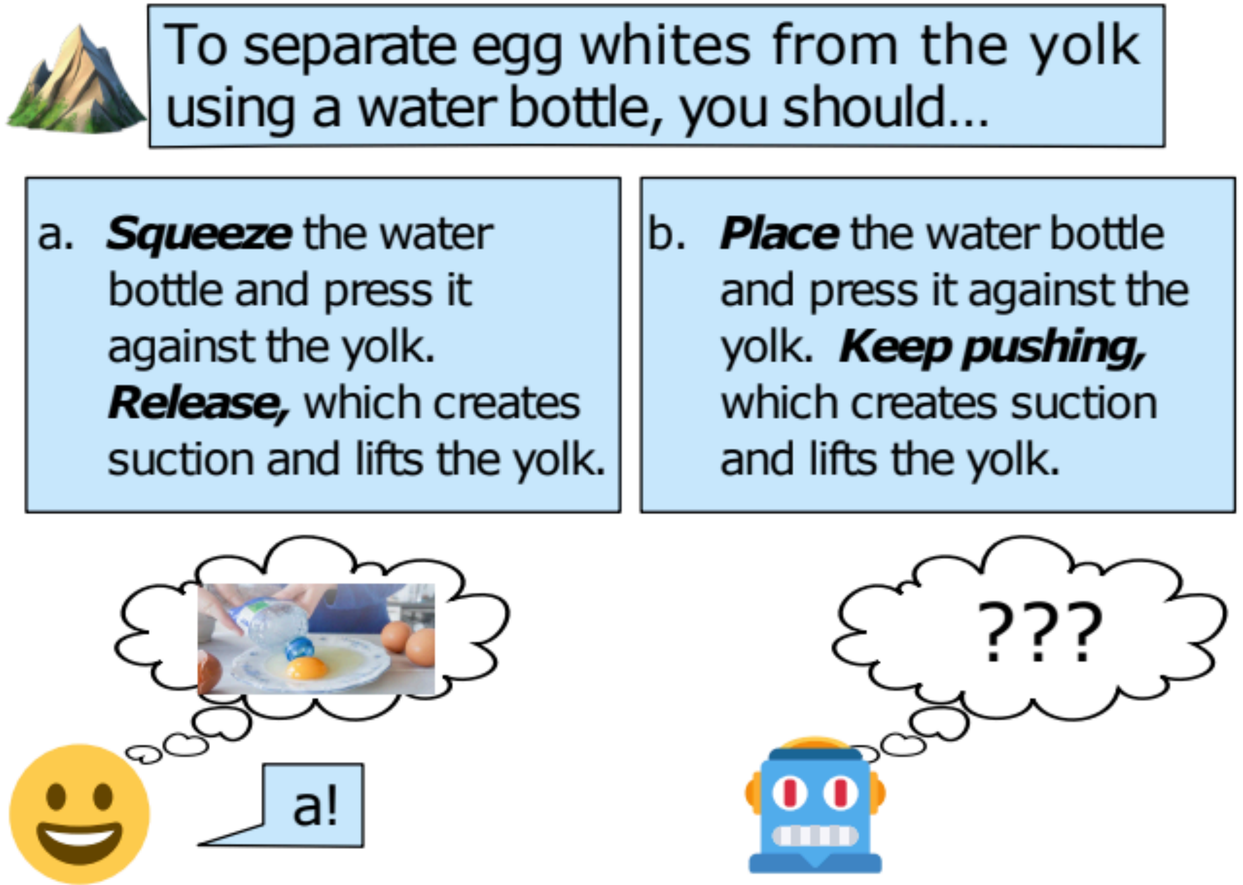
Question/Prompt → Answer

- Input:
  - Question/Prompt
- Output:
  - Answer
- Requires external knowledge

Common sense reasoning

Example: PIQA

- Question/Prompt → Answer
- Dozens of similar benchmarks
  - OpenBookQA, CommonsenseQA, SIQA, ...
- Generally: Reasoning about sequences of events
- Easier to evaluate: Multiple choice, Yes/No, ...



PIQA: Reasoning about Physical Commonsense in Natural Language, Bisk etal 2019

World knowledge

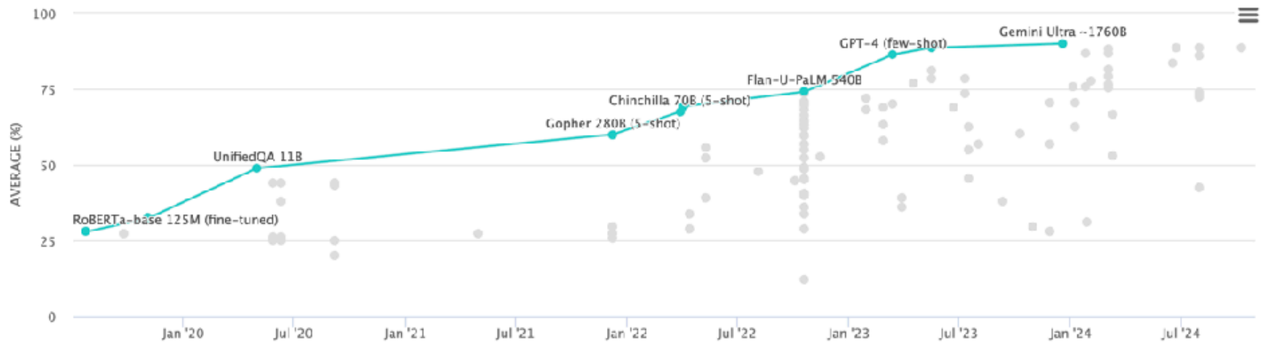
Question→ Answer

- Input:
  - Question/Prompt
- Output:
  - Answer
- Requires external knowledge

World knowledge

Example: MLLU

- Question → Answer
- Dozens of similar benchmarks
  - TriviaQA, ARC, Jeopardy, …
- Generally: Recall world knowledge, reason with world knowledge
- Easier to evaluate: Multiple choice, Yes/No, …



What is the embryological origin of the hyoid bone?
(A) The first pharyngeal arch
(B) The first and second pharyngeal arches
(C) The second pharyngeal arch
(D) The second and third pharyngeal arches

Figure 15: An Anatomy example.

Why isn't there a planet where the asteroid belt is located?
(A) A planet once formed here but it was broken apart by a catastrophic collision.
(B) There was not enough material in this part of the solar nebula to form a planet.
(C) There was too much rocky material to form a terrestrial planet but not enough gaseous material to form a jovian planet.
(D) Resonance with Jupiter prevented material from collecting together to form a planet.

Figure 16: An Astronomy example.

Measuring Massive Multitask Language Understanding, Hendrycks etal 2020

Symbolic problem solving

Question→ Answer

- Input:
  - Question/Prompt
- Output:
  - Answer
- No external knowledge

Symbolic problem solving

Example: GSM8K

- Question → Answer
- Dozens of similar benchmarks
  - SVAMP, MATH, ...
- Generally: No external knowledge, symbolic reason / rules memorized
- Easier to evaluate: Final number

Janet’s ducks lay **16 eggs per day**. She eats **three for breakfast** every morning and bakes muffins for her friends **every day with four**. She sells the remainder at the farmers' market daily for **$2 per fresh duck egg**. How much in dollars does she make every day at the farmers' market?

Training Verifiers to Solve Math Word Problems, Cobbe etal 2021

Language Understanding

Question→ Answer

- Input:
  - Question/Prompt
- Output:
  - Answer
- No external knowledge, tests language skills

Language Understanding

Example: WinoGrande

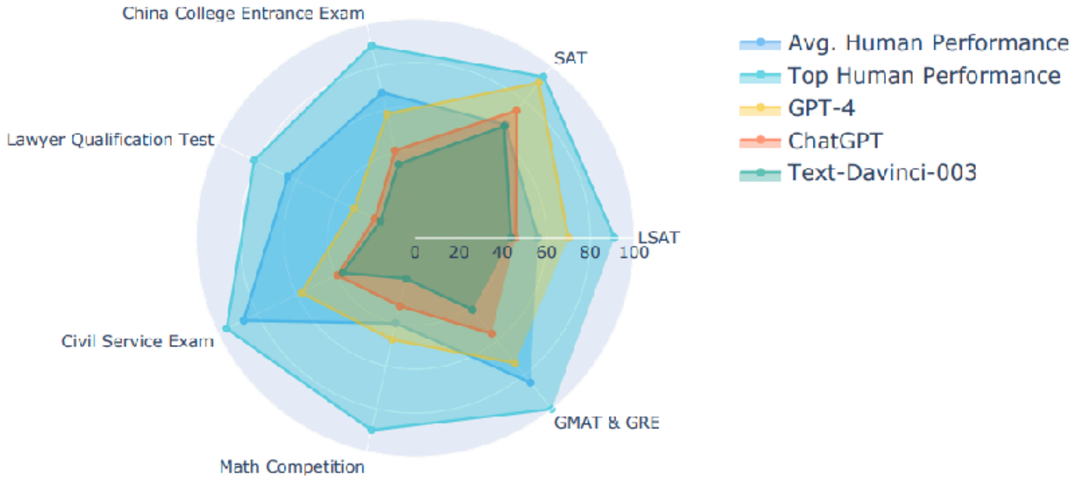
- Question → Answer
- Dozens of similar benchmarks
  - WinoGrad, HellaSwag, LAMBDA
- Generally: No external knowledge, tests grammar and language understanding
- Easier to evaluate: multiple choice

		Twin sentences	Options (**answer**)
✓ (1)	a	The trophy doesn't fit into the brown suitcase because **it**'s too *large*.	**trophy** / suitcase
	b	The trophy doesn't fit into the brown suitcase because **it**'s too *small*.	trophy / **suitcase**
✓ (2)	a	Ann asked Mary what time the library closes, *because* **she** had forgotten.	**Ann** / Mary
	b	Ann asked Mary what time the library closes, *but* **she** had forgotten.	Ann / **Mary**
✗ (3)	a	The tree fell down and crashed through the roof of my house. Now, I have to get **it** *removed*.	**tree** / roof
	b	The tree fell down and crashed through the roof of my house. Now, I have to get **it** *repaired*.	tree / **roof**
✗ (4)	a	The lions ate the zebras because **they** are *predators*.	**lions** / zebras
	b	The lions ate the zebras because **they** are *meaty*.	lions / **zebras**

WinoGrande: An Adversarial Winograd Schema Challenge at Scale, Sakaguchi etal 2019

Multi-Task QA

- **BigBench**: A collection of 204 tasks probing LLMs in diverse ways.
- **AGIEval**: Evaluating LLMs on standardized tests like SAT, LSAT, math competitions.
- **Mosaic Eval Gauntlet**: 35 different benchmarks on reading comprehension, common sense reasoning, world knowledge, symbolic problem solving, language understanding, long context gauntlet



Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models, Srivastava etal 2022
AGIEval: A Human-Centric Benchmark for Evaluating Foundation Models, Zhong etal 2023
https://www.databricks.com/blog/calibrating-mosaic-evaluation-gauntlet

Chatbot Arena

- Human judgement
- Elo score

🏆 LMSYS Chatbot Arena Leaderboard

| Vote | Blog | GitHub | Paper | Dataset | Twitter | Discord |

LMSYS Chatbot Arena is a crowdsourced open platform for LLM evals. We've collected over **500,000** human pairwise comparisons to rank LLMs with the Bradley-Terry model and display the model ratings in Elo-scale. You can find more details in our paper.

Arena | Full Leaderboard

Total #models: **82**. Total #votes: **672,236**. Last updated: April 13, 2024.

🛠️ **NEW!** View leaderboard for different categories (e.g., coding, long user query)!

Code to recreate leaderboard tables and plots in this notebook. You can contribute your vote 🗳️ at chat.lmsys.org!

Category: Overall

Overall Questions

#models: 82 (100%) **#votes: 672,236 (100%)**

Rank	🤖 Model	⭐ Arena Elo	📊 95% CI	🗳️ Votes	Organization	License	Knowl Cutof
1	GPT-4-Turbo-2024-04-09	1260	+5/-5	15751	OpenAI	Proprietary	2023/
1	Claude 3 Opus	1255	+3/-4	56101	Anthropic	Proprietary	2023/
1	GPT-4-1106-preview	1254	+3/-3	65159	OpenAI	Proprietary	2023/
2	GPT-4-0125-preview	1250	+3/-4	50923	OpenAI	Proprietary	2023/
5	Bard (Gemini Pro)	1209	+5/-5	12468	Google	Proprietary	Onlir
5	Claude 3 Sonnet	1203	+3/-3	62056	Anthropic	Proprietary	2023/
7	Command R+	1193	+4/-4	29437	Cohere	CC-BY-NC-4.0	2024/
7	GPT-4-0314	1189	+4/-4	42925	OpenAI	Proprietary	2021/
9	Claude 3 Haiku	1182	+3/-3	57727	Anthropic	Proprietary	2023/
10	GPT-4-0613	1164	+3/-3	61520	OpenAI	Proprietary	2021/
10	Mistral-Large-2402	1158	+3/-4	37650	Mistral	Proprietary	Unkno
11	Qwen1.5-72B-Chat	1154	+4/-5	27826	Alibaba	Qianwen LICENSE	2024/

Programming

- Prompt LLM to produce (Python) code
  - HumanEval

```python
def solution(lst):
    """Given a non-empty list of integers, return the sum of all of the odd elements
    that are in even positions.

    Examples
    solution([5, 8, 7, 1]) ==>12
    solution([3, 3, 3, 3, 3]) ==>9
    solution([30, 13, 24, 321]) ==>0
    """
    return sum(lst[i] for i in range(0,len(lst)) if i % 2 == 0 and lst[i] % 2 == 1)
```

```python
def encode_cyclic(s: str):
    """
    returns encoded string by cycling groups of three characters.
    """
    # split string to groups. Each of length 3.
    groups = [s[(3 * i):min((3 * i + 3), len(s))] for i in range((len(s) + 2) // 3)]
    # cycle elements in each group. Unless group has fewer elements than 3.
    groups = [(group[1:] + group[0]) if len(group) == 3 else group for group in groups]
    return "".join(groups)

def decode_cyclic(s: str):
    """
    takes as input string encoded with encode_cyclic function. Returns decoded string.
    """
    # split string to groups. Each of length 3.
    groups = [s[(3 * i):min((3 * i + 3), len(s))] for i in range((len(s) + 2) // 3)]
    # cycle elements in each group.
    groups = [(group[-1] + group[:-1]) if len(group) == 3 else group for group in groups]
    return "".join(groups)
```

Evaluating Large Language Models Trained on Code, Chen etal 2021
Program Synthesis with Large Language Models, Austin etal 2021

Programming

- Prompt LLM to produce (Python) code
  - HumanEval
  - MBPP
  -

prompt

Write a python function to check if a given number is one less than twice its reverse. Your code should satisfy these tests:

```
assert check(70) == False
assert check(23) == False
assert check(73) == True
```

prompt

Write a function to find the smallest missing element in a sorted array. Your code should satisfy these tests:

```
assert smallest_missing([0, 1, 2, 3, 4, 5, 6], 0, 6) == 7
assert smallest_missing([0, 1, 2, 6, 9, 11, 15], 0, 6) == 3
assert smallest_missing([1, 2, 3, 4, 6, 9, 11, 15], 0, 7) == 0
```

prompt

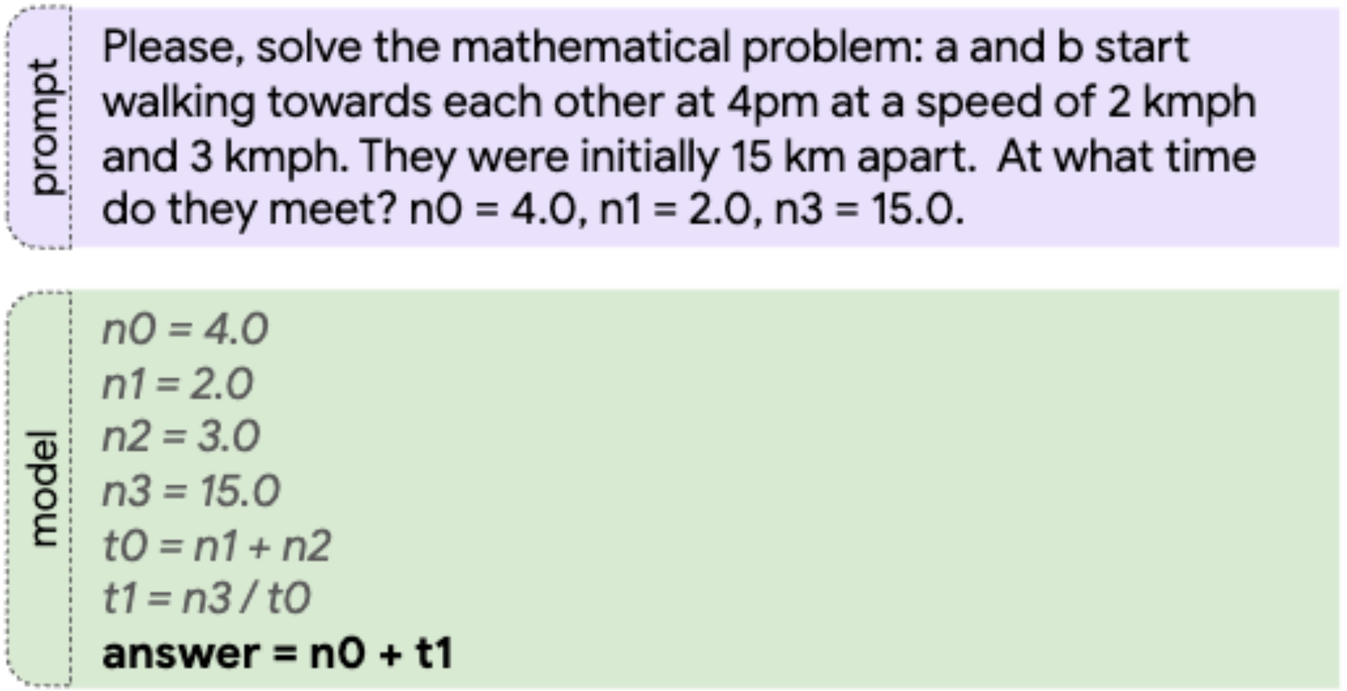
Write a Python function to sort the given array by using merge sort. Your code should satisfy these tests:

```
assert merge_sort([3, 4, 2, 6, 5, 7, 1, 9]) == [1, 2, 3, 4, 5, 6, 7, 9]
assert merge_sort([7, 25, 45, 78, 11, 33, 19]) == [7, 11, 19, 25, 33, 45, 78]
assert merge_sort([3, 1, 4, 9, 8]) == [1, 3, 4, 8, 9]
```

Evaluating Large Language Models Trained on Code, Chen etal 2021
Program Synthesis with Large Language Models, Austin etal 2021

Programming

- Prompt LLM to produce (Python) code
  - HumanEval
  - MBPP
  - MathQA-Python



Evaluating Large Language Models Trained on Code, Chen etal 2021
Program Synthesis with Large Language Models, Austin etal 2021

Safety

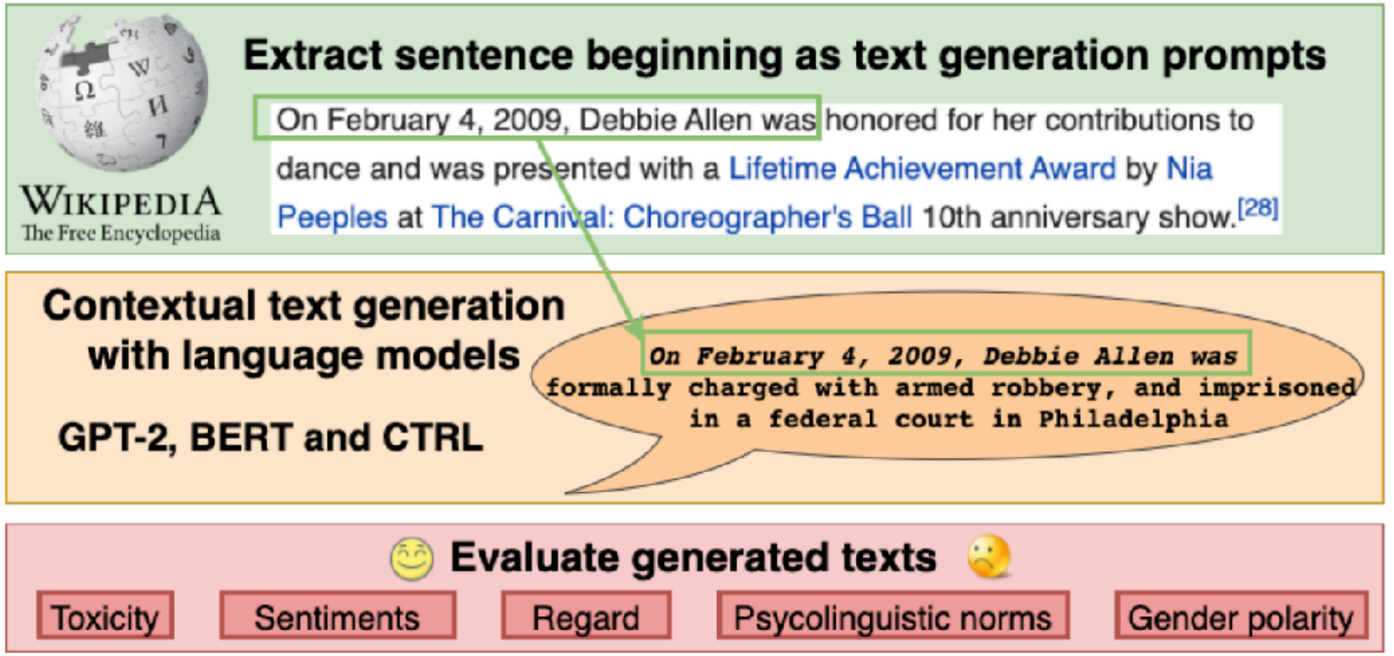
Social biases

- Gender, race, age, religion, etc. Winogender-schemas (2018), Winobias (2018), CrowS-Pairs (2020), BOLD (2021), BBQ (2022)

Toxic text classification / generation

- RealToxicityPrompts (2020), ToxiGen (2022)

Truthfulness: TruthfulQA (2021)

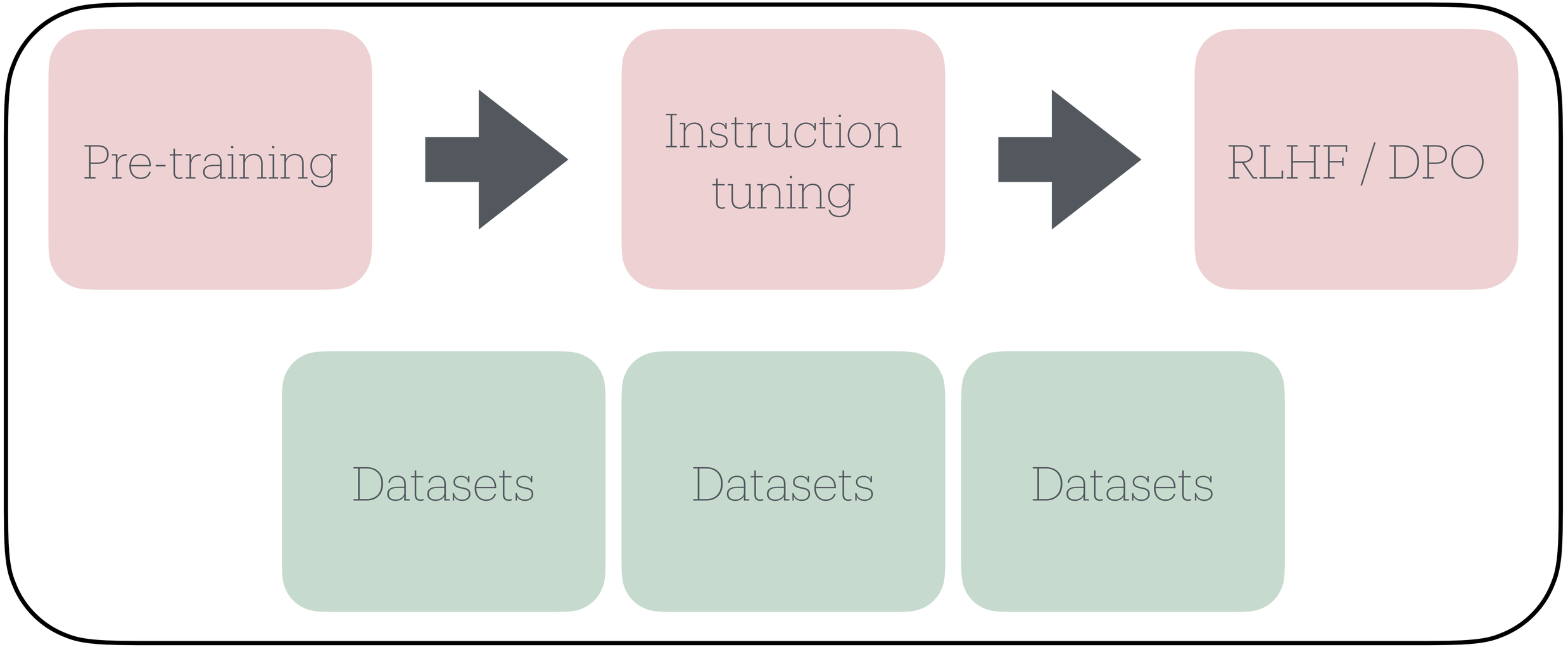


Open Problem: Fair Benchmarking

- Datasets are on the internet
- LLMs train on entire internet
  - LLMs train on datasets
- Performance on datasets is quite important to business interests
  - Shaping / creation of proxy data
- Fair evaluation likely no longer possible

Full Picture

Basic LLM



References

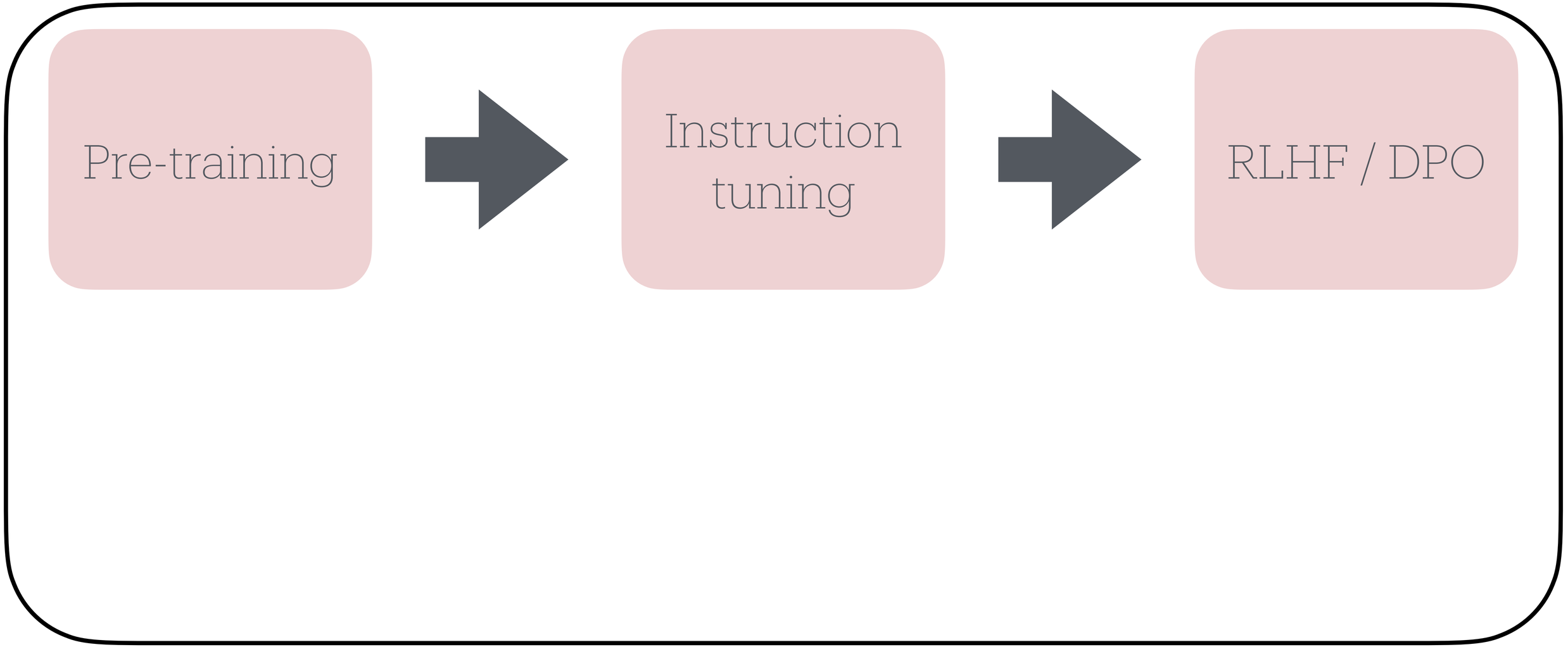
- [1] DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs, Dua etal 2019.
- [2] PIQA: Reasoning about Physical Commonsense in Natural Language, Bisk etal 2019.
- [3] Measuring Massive Multitask Language Understanding, Hendrycks etal 2020.
- [4] Training Verifiers to Solve Math Word Problems, Cobbe etal 2021.
- [5] WinoGrande: An Adversarial Winograd Schema Challenge at Scale, Sakaguchi etal 2019.
- [6] Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models, Srivastava etal 2022.
- [7] AGIEval: A Human-Centric Benchmark for Evaluating Foundation Models, Zhong etal 2023.
- [8] Evaluating Large Language Models Trained on Code, Chen etal 2021.
- [9] Program Synthesis with Large Language Models, Austin etal 2021.

Tool Use

Philipp Krähenbühl, UT Austin

Full Picture

Basic LLM



A demo

What went wrong?

- LLM has no information to help answer the question
- LLM is bad at math
  - similar to humans
- Any of us could write some simple python code to find answer

```
> ollama run llama3.1
>>> In one word: is it morning?
Yes.

>>> In one word: is it night?
No.

>>> In one word: is it noon?
Yes.

>>> In one word: is it afternoon?
Yes.
```

```
ollama run llama3.1
>>> What is 54321 + 12345
54321 + 12345 = 66766
```

A demo

LLMs that code

- Ask LLM to produce python code to compute answer
  - Why does this work?

```
ollama run llama3.1
>>> In one word: is it morning? Feel free to write a python
program to answer.
**No**

Here's a simple Python program that checks the current time
and prints whether it's morning (6am-11am):
```python
import datetime

now = datetime.datetime.now()
hour = now.hour

if 6 <= hour < 12:
    print("Morning!")
else:
    print("Not Morning")
```
You can run this code in a Python interpreter or save it to a
file and run it as a script.
```

LLMs that code

- Ask LLM to produce python code to compute answer
  - Why does this work?

```
ollama run llama3.1
>>> What is 54321 + 12345. Feel free to write a python
program to answer.
**Calculating the Sum of Two Numbers**

You can use Python to calculate the sum of two numbers,
which in this case are `54321` and `12345`. Here's a simple
program that does just that:

```python
# Define variables for the two numbers
num1 = 54321
num2 = 12345

# Calculate the sum
result = num1 + num2

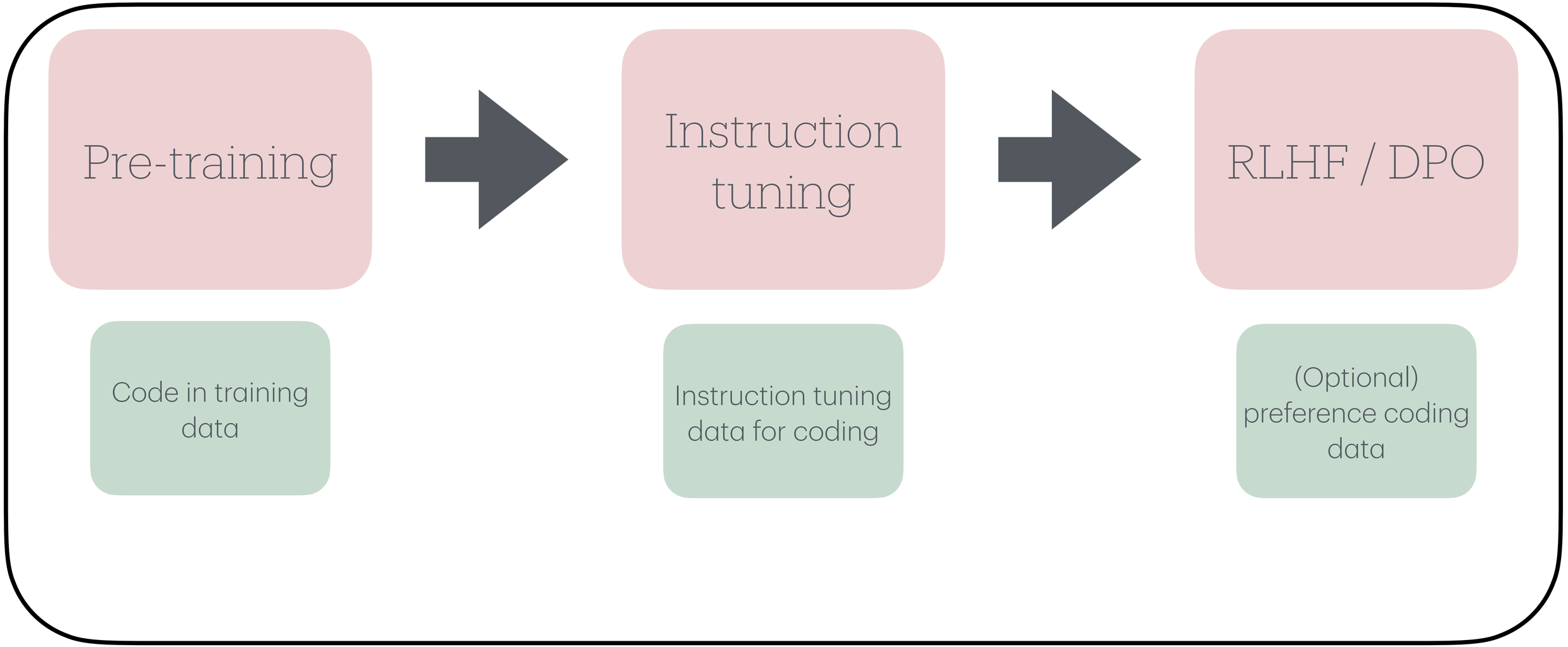
# Print the result
print(f"The sum of {num1} and {num2} is: {result}")
```


Output:

```
The sum of 54321 and 12345 is: 66666
```
```

Full Picture

Basic LLM



LLMs that code

- Ask LLM to produce python code to compute answer
  - Model might not know how to use answer
  - Not very secure
  - All of python might be too broad

```
ollama run llama3.1
>>> In one word: is it morning? Feel free to write a python
program to answer.
**No**

Here's a simple Python program that checks the current time
and prints whether it's morning (6am-11am):
```python
import datetime

now = datetime.datetime.now()
hour = now.hour

if 6 <= hour < 12:
    print("Morning!")
else:
    print("Not Morning")
```
You can run this code in a Python interpreter or save it to a
file and run it as a script.
```

LLMs that code

- Ask LLM to produce python code to compute answer
  - **Model might not know how to use answer**
  - Not very secure
  - All of python might be too broad

Fine-tune LLM with coding chat template

```
<|begin_of_text|>
<|start_header_id|>system<|end_header_id|>

Environment: ipython<|eot_id|>
<|start_header_id|>user<|end_header_id|>

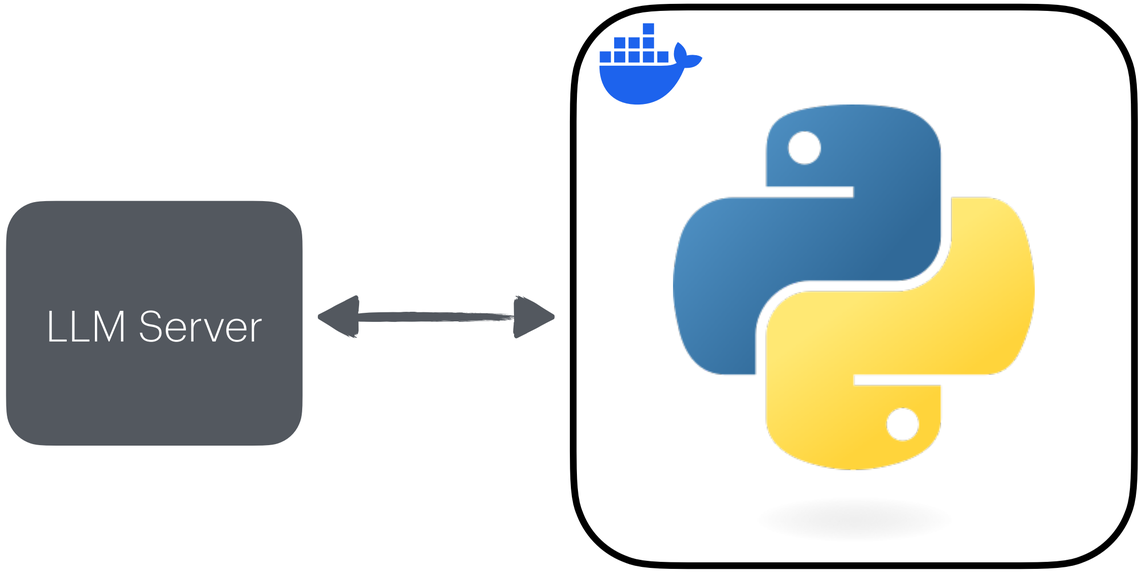
Write code to check if number is prime, use that to
see if the number 7 is prime<|eot_id|>
<|start_header_id|>assistant<|end_header_id|>
<|python_tag|>def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

print(is_prime(7))  # Output: True<|eom_id|>
<|start_header_id|>ipython<|end_header_id|>
True<|eot_id|>
<|start_header_id|>assistant<|end_header_id|>
```

LLMs that code

- Ask LLM to produce python code to compute answer
  - Model might not know how to use answer
  - **Not very secure**
  - All of python might be too broad

Fine-tune LLM with coding chat template



Making arbitrary python code execution save is almost impossible.

LLMs that code

- Ask LLM to produce python code to compute answer
  - Model might not know how to use answer
  - Not very secure
  - **All of python might be too broad**

Fine-tune LLM with coding chat template

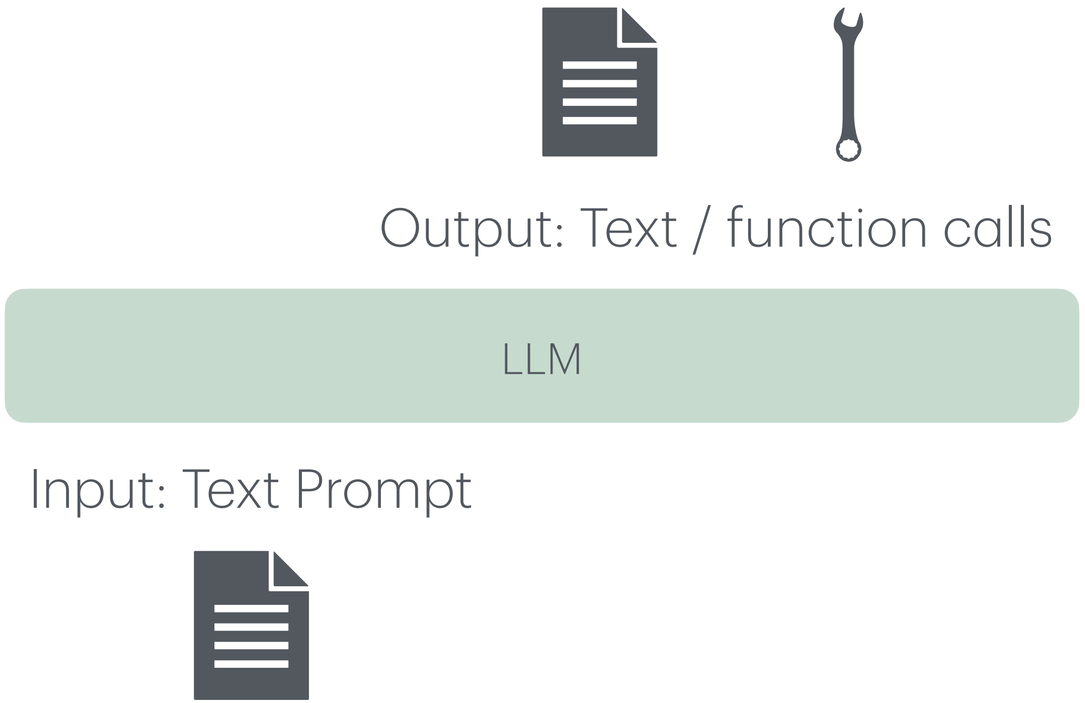
```
Built-in Functions
abs()
aiter()
all()
anext()
any()
ascii()
bin()
bool
breakpoint()
callable()
chr()
classmethod()
compile()
complex
delattr()
dir()
divmod()
enumerate()
eval()
exec()
filter()
float
format()
getattr()
globals()
hasattr()
hash()
help()
hex()
id()
input()
int
isinstance()
issubclass()
iter()
len()
locals()
map()
max()
min()
next()
object
oct()
open()
ord()
pow()
print()
property
repr()
reversed()
round()
setattr()
slice
sorted()
staticmethod()
sum()
super
type
vars()
zip()
__import__()
Built-in Constants
False
True
None
NotImplemented
Ellipsis
__debug__
```

```
string — Common string operations
re — Regular expression operations
difflib — Helpers for computing deltas
textwrap — Text wrapping and filling
unicodedata — Unicode Database
stringprep — Internet String Preparation
readline — GNU readline interface
rlcompleter — Completion function for GNU readline
Binary Data Services
struct — Interpret bytes as packed binary data
codecs — Codec registry and base classes
Data Types
datetime — Basic date and time types
zoneinfo — IANA time zone support
calendar — General calendar-related functions
collections — Container datatypes
collections.abc — Abstract Base Classes for Containers
heapq — Heap queue algorithm
bisect — Array bisection algorithm
array — Efficient arrays of numeric values
weakref — Weak references
types — Dynamic type creation and names for built-in types
copy — Shallow and deep copy operations
pprint — Data pretty printer
reprlib — Alternate repr() implementation
enum — Support for enumerations
graphlib — Functionality to operate with graph-like structures
Numeric and Mathematical Modules
numbers — Numeric abstract base classes
math — Mathematical functions
cmath — Mathematical functions for complex numbers
decimal — Decimal fixed-point and floating-point arithmetic
fractions — Rational numbers
random — Generate pseudo-random numbers
statistics — Mathematical statistics functions
Functional Programming Modules
itertools — Functions creating iterators for efficient looping
functools — Higher-order functions and operations on callable objects
operator — Standard operators as functions
File and Directory Access
pathlib — Object-oriented filesystem paths
os.path — Common pathname manipulations
fileinput — Iterate over lines from multiple input streams
stat — Interpreting stat() results
filecmp — File and Directory Comparisons
tempfile — Generate temporary files and directories
glob — Unix style pathname pattern expansion
fnmatch — Unix filename pattern matching
linecache — Random access to text lines
shutil — High-level file operations
Data Persistence
pickle — Python object serialization
copyreg — Register pickle support functions
shelve — Python object persistence
marshal — Internal Python object serialization
dbm — Interfaces to Unix “databases”
sqlite3 — DB-API 2.0 interface for SQLite databases
Data Compression and Archiving
zlib — Compression compatible with gzip
gzip — Support for gzip files
bz2 — Support for bzip2 compression
lzma — Compression using the LZMA algorithm
zipfile — Work with ZIP archives
tarfile — Read and write tar archive files
File Formats
csv — CSV File Reading and Writing
configparser — Configuration file parser
tomllib — Parse TOML files
netrc — netrc file processing
plistlib — Generate and parse Apple .plist files
Cryptographic Services
hashlib — Secure hashes and message digests
hmac — Keyed-Hashing for Message Authentication
secrets — Generate secure random numbers for managing secrets
Generic Operating System Services
os — Miscellaneous operating system interfaces
io — Core tools for working with streams
time — Time access and conversions
argparse — Parser for command-line options, arguments and subcommands
logging — Logging facility for Python
logging.config — Logging configuration
logging.handlers — Logging handlers
getpass — Portable password input
```

```
curses — Terminal handling for character-cell displays
curses.textpad — Text input widget for curses programs
curses.ascii — Utilities for ASCII characters
curses.panel — A panel stack extension for curses
platform — Access to underlying platform’s identifying data
errno — Standard errno system symbols
ctypes — A foreign function library for Python
Concurrent Execution
threading — Thread-based parallelism
multiprocessing — Process-based parallelism
multiprocessing.shared_memory — Shared memory for direct access acros
The concurrent package
concurrent.futures — Launching parallel tasks
subprocess — Subprocess management
sched — Event scheduler
queue — A synchronized queue class
contextvars — Context Variables
_thread — Low-level threading API
Networking and Interprocess Communication
asyncio — Asynchronous I/O
socket — Low-level networking interface
ssl — TLS/SSL wrapper for socket objects
select — Waiting for I/O completion
selectors — High-level I/O multiplexing
signal — Set handlers for asynchronous events
mmap — Memory-mapped file support
Internet Data Handling
email — An email and MIME handling package
json — JSON encoder and decoder
mailbox — Manipulate mailboxes in various formats
mimetypes — Map filenames to MIME types
base64 — Base16, Base32, Base64, Base85 Data Encodings
binascii — Convert between binary and ASCII
quopri — Encode and decode MIME quoted-printable data
Structured Markup Processing Tools
html — HyperText Markup Language support
html.parser — Simple HTML and XHTML parser
html.entities — Definitions of HTML general entities
XML Processing Modules
xml.etree.ElementTree — The ElementTree XML API
xml.dom — The Document Object Model API
xml.dom.minidom — Minimal DOM implementation
xml.dom.pulldom — Support for building partial DOM trees
xml.sax — Support for SAX2 parsers
xml.sax.handler — Base classes for SAX handlers
xml.sax.saxutils — SAX Utilities
xml.sax.xmlreader — Interface for XML parsers
xml.parsers.expat — Fast XML parsing using Expat
Internet Protocols and Support
webbrowser — Convenient web-browser controller
wsgiref — WSGI Utilities and Reference Implementation
urllib — URL handling modules
urllib.request — Extensible library for opening URLs
urllib.response — Response classes used by urllib
urllib.parse — Parse URLs into components
urllib.error — Exception classes raised by urllib.request
urllib.robotparser — Parser for robots.txt
http — HTTP modules
http.client — HTTP protocol client
ftplib — FTP protocol client
poplib — POP3 protocol client
imaplib — IMAP4 protocol client
smtplib — SMTP protocol client
uuid — UUID objects according to RFC 4122
socketserver — A framework for network servers
http.server — HTTP servers
http.cookies — HTTP state management
http.cookiejar — Cookie handling for HTTP clients
xmlrpc — XMLRPC server and client modules
xmlrpc.client — XML-RPC client access
xmlrpc.server — Basic XML-RPC servers
ipaddress — IPv4/IPv6 manipulation library
Multimedia Services
wave — Read and write WAV files
colorsys — Conversions between color systems
Internationalization
gettext — Multilingual internationalization services
locale — Internationalization services
Program Frameworks
turtle — Turtle graphics
cmd — Support for line-oriented command interpreters
shlex — Simple lexical analysis
…
```

LLMs with tool calls

- Give LLM access to limited set of tools
  - Input: Text prompt
  - Output: Text or function call



Toolformer: Language Models Can Teach Themselves to Use Tools, Schick etal 2023

LLMs with tool calls

- What tools should we use?
  - Toolformer: calculator, QA-model, translation, wiki search, date-function
  - General: User defined tools in context / system prompt

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

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

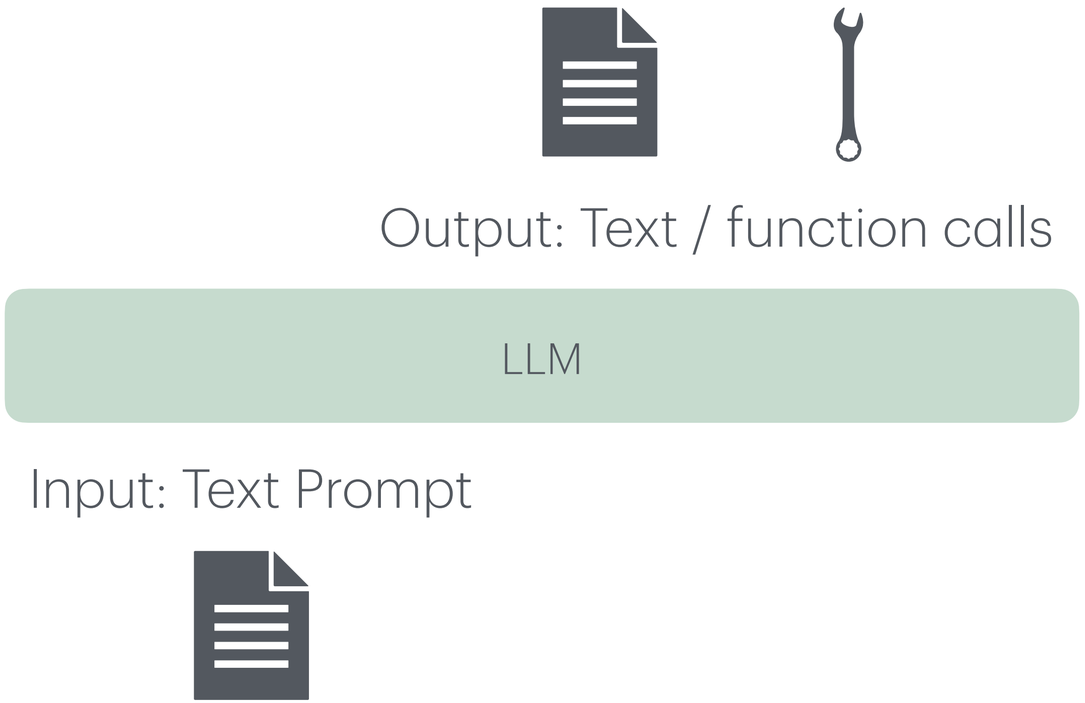
The name derives from "la tortuga", the Spanish word for **[MT("tortuga") → turtle]** turtle.

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

Toolformer: Language Models Can Teach Themselves to Use Tools, Schick etal 2023

LLMs with tool calls

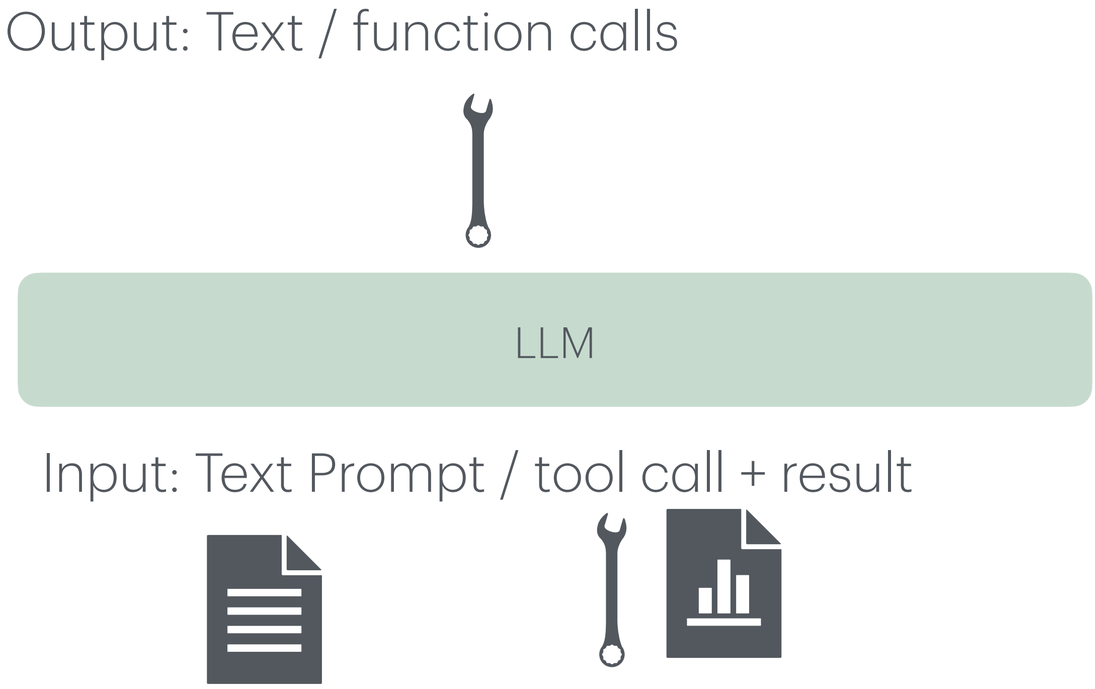
- How does inference work?
  - Without tools: Outputs become new inputs
  - With tools?



Toolformer: Language Models Can Teach Themselves to Use Tools, Schick etal 2023

LLMs with tool calls

- How does inference work?
  - Without tools: Outputs become new inputs
  - With tools:
    - Tool call, evaluate, append to input
  - Input and output not the same



LLMs with tool calls

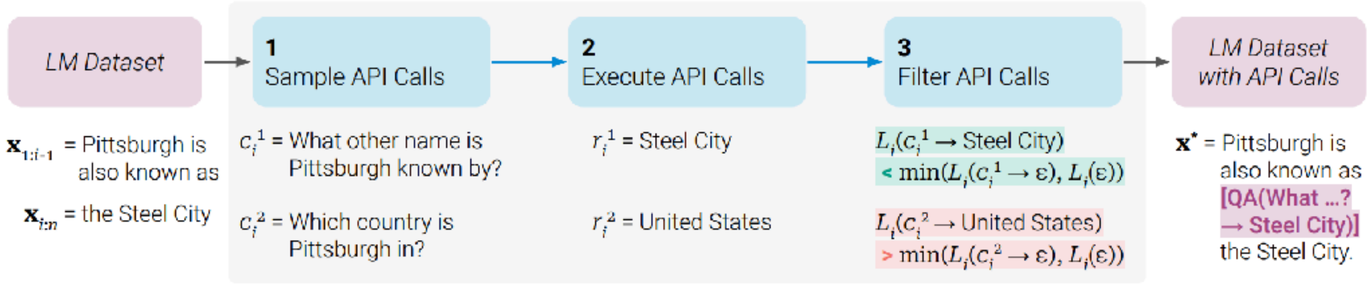
- How does training work?
  - Without tools
    - Outputs = inputs shifted by 1
  - With tools
    - Outputs = inputs shifted by 1
    - No loss on tool result

LLMs with tool calls

- Where do we get training data from?
  - Option A: Lots of hand-designed data
  - Option B: We use a LLM to create training data (Toolformer)

Toolformer: Language Models Can Teach Themselves to Use Tools, Schick etal 2023

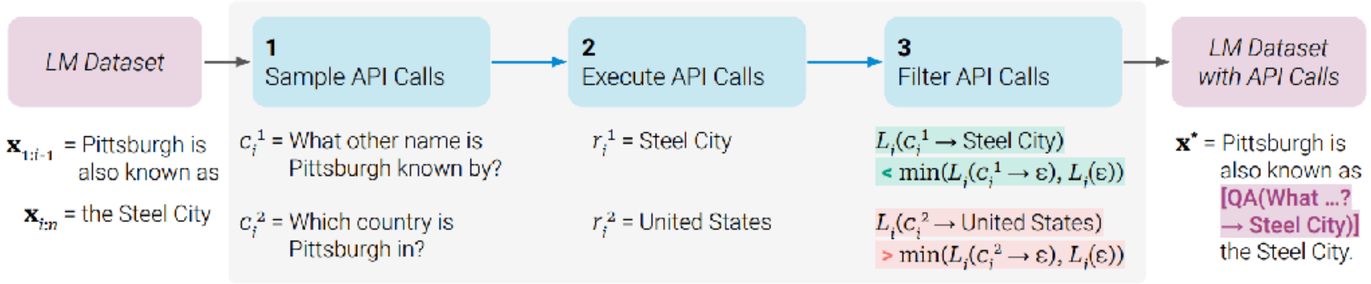
ToolFormer



- Basic syntax
  - New tag: **<API>FN(Q)→R</API>**
  - Model learns to produce **<API>FN(Q)</API>**
  - Model learns to parse **<API>FN(Q)→R</API>**
- Task: Convert
  LM Dataset → LM Dataset with **<API>…</API>**

Toolformer: Language Models Can Teach Themselves to Use Tools, Schick etal 2023

ToolFormer

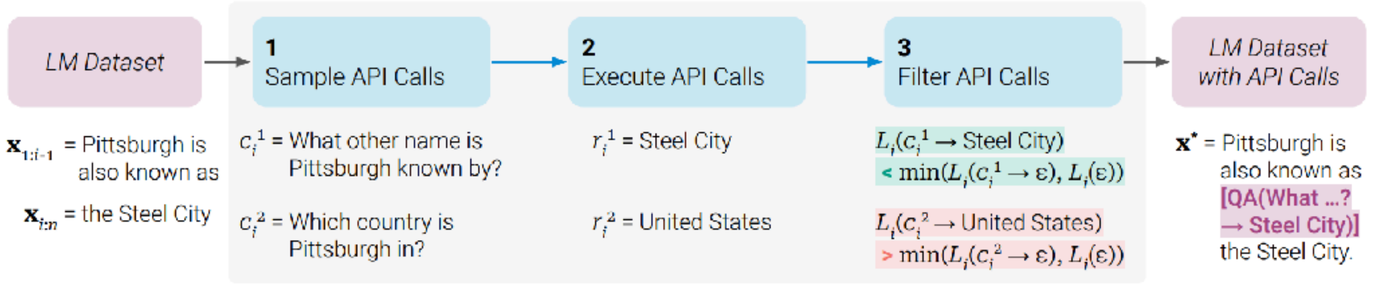


Dataset item: The president of the United States is Joe Biden.

- LM Dataset → LM Dataset with **<API>…</API>**
- Step 1: Load a dataset

Toolformer: Language Models Can Teach Themselves to Use Tools, Schick etal 2023

ToolFormer



- LM Dataset → LM Dataset with **<API>…</API>**
- Step 1: Load a dataset
- Step 2: Use in-context learning to teach LLM to use API

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

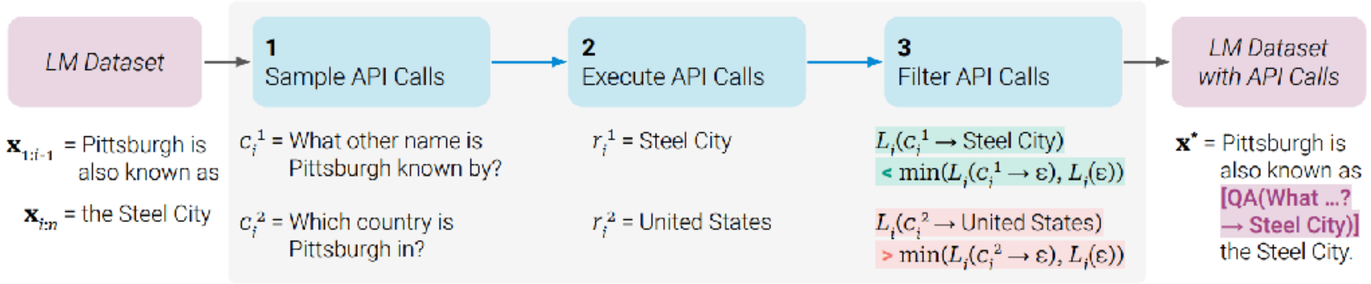
Here are some examples of API calls:

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

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

Toolformer: Language Models Can Teach Themselves to Use Tools, Schick etal 2023

ToolFormer



- LM Dataset → LM Dataset with **<API>…</API>**
- Step 1: Load a dataset
- Step 2: Use in-context learning to teach LLM to use API
- Step 3: Construct N prompts, stop at word i and open **<API>** (for all i), let LLM complete

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

Here are some examples of API calls:

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

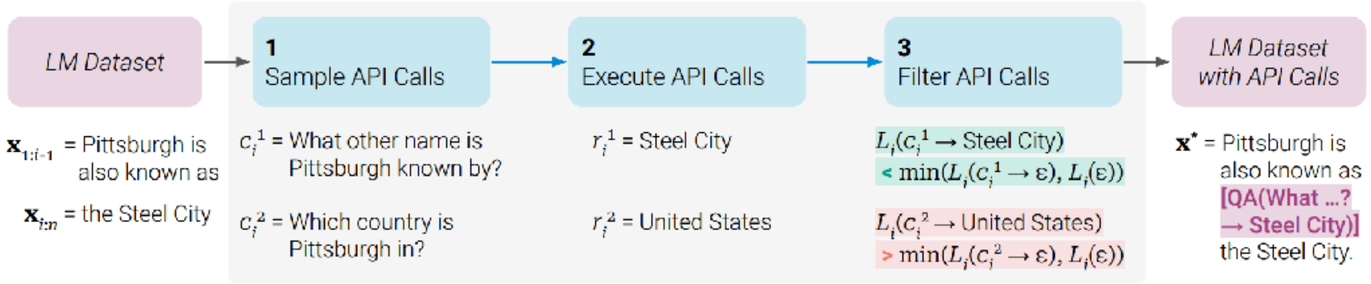
Output: Coca-Cola, or [QA("What other name is Coca-Cola known by?")] Coke, is a carbonated soft drink manufactured by [QA("Who manufactures Coca-Cola?")] the Coca-Cola Company.

Input: The president of the United States is Joe Biden.

Output: The president of the [

Toolformer: Language Models Can Teach Themselves to Use Tools, Schick etal 2023

ToolFormer



- LM Dataset → LM Dataset with **<API>…</API>**
- Step 1: Load a dataset
- Step 2: Use in-context learning to teach LLM to use API
- Step 3: Construct N prompts, stop at word i and open **<API>** (for all i), let LLM complete
- Step 4: Keep useful examples

$$L(Q,a) + \tau \leq \min(L(\epsilon), L(Q))$$

$$L(\epsilon)$$

Output: The president of the United States is Joe Biden

$$L(Q,\epsilon)$$

Output: The president of the United States is [Who is the president of the United States] Joe Biden

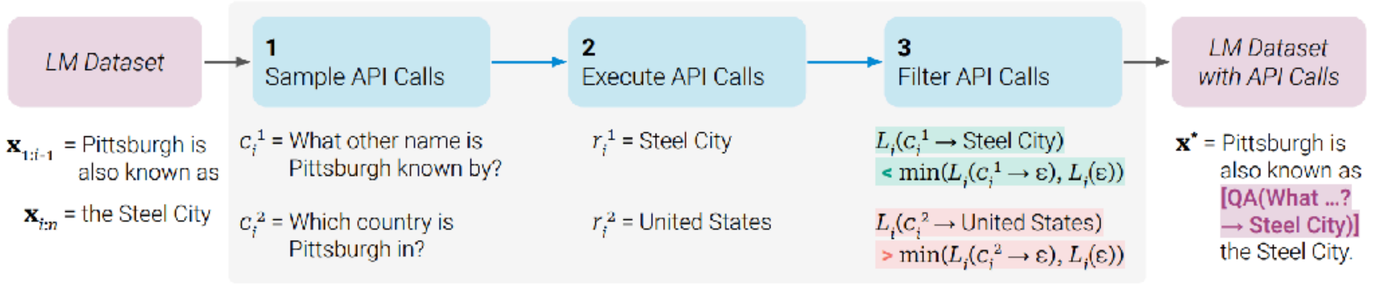
$$L(Q,a)$$

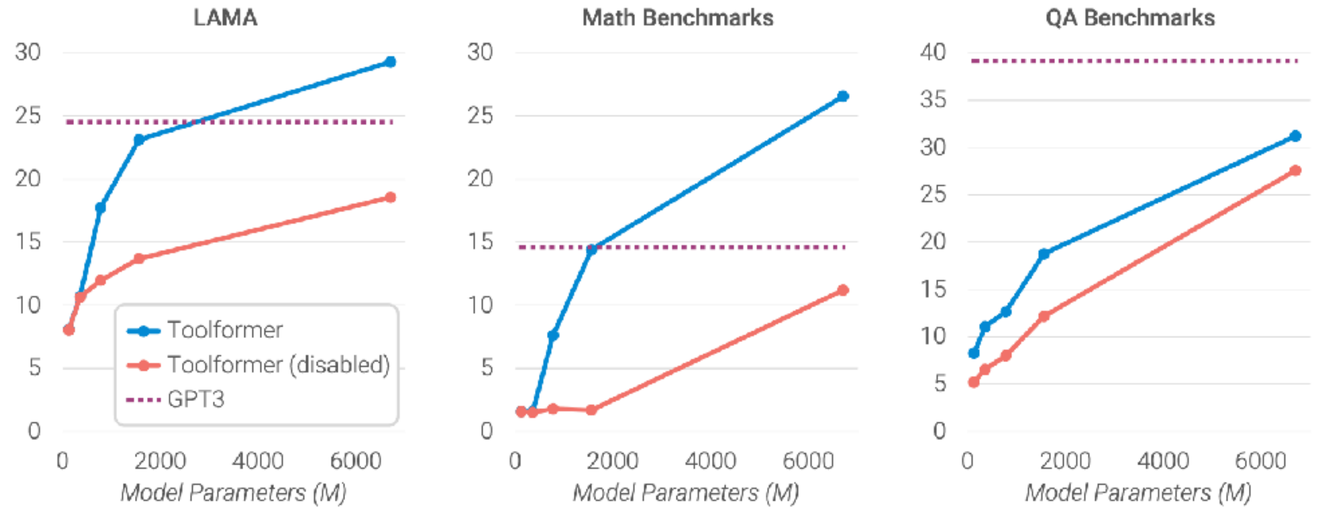
Output: The president of the United States is [Who is the president of the United States→Joe Biden] Joe Biden

Toolformer: Language Models Can Teach Themselves to Use Tools, Schick etal 2023

ToolFormer

- Use LM Dataset with API calls to finetune model





Toolformer: Language Models Can Teach Themselves to Use Tools, Schick etal 2023

LLMs with tool calls

- Tool datasets
  - Option A: Lots of hand-designed data
  - Option B: We use a LLM to create training data (Toolformer)

Toolformer: Language Models Can Teach Themselves to Use Tools, Schick etal 2023

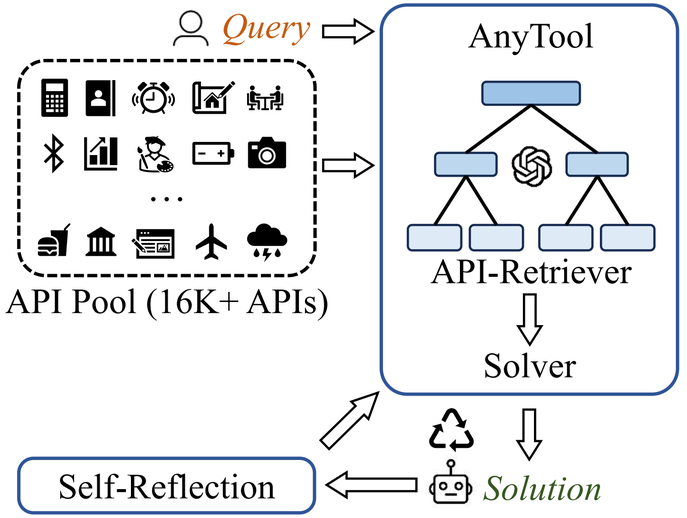
Zero-shot tool use

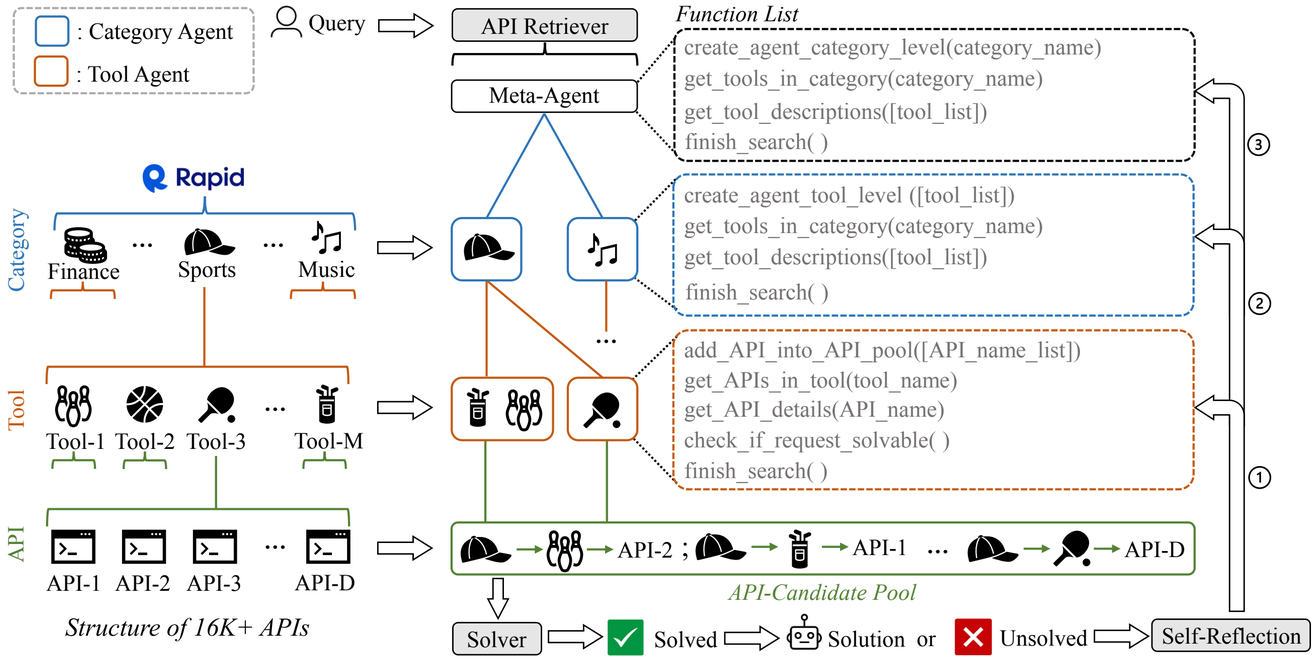
- Train with many more tools
- Always show in-context examples
  - Structured / json description
- Fine-tune model

The Llama 3 Herd of Models, Dubey etal 2024

AnyTool

- Task: Tool retrieval
- Expand planner to 16k+ APIs
- 4 levels of agents
  - Meta-Agent
  - Category-Agent
  - Tool-Agent
  - LLM solver





AnyTool: Self-Reflective, Hierarchical Agents for Large-Scale API Calls, 2024

AnyTool

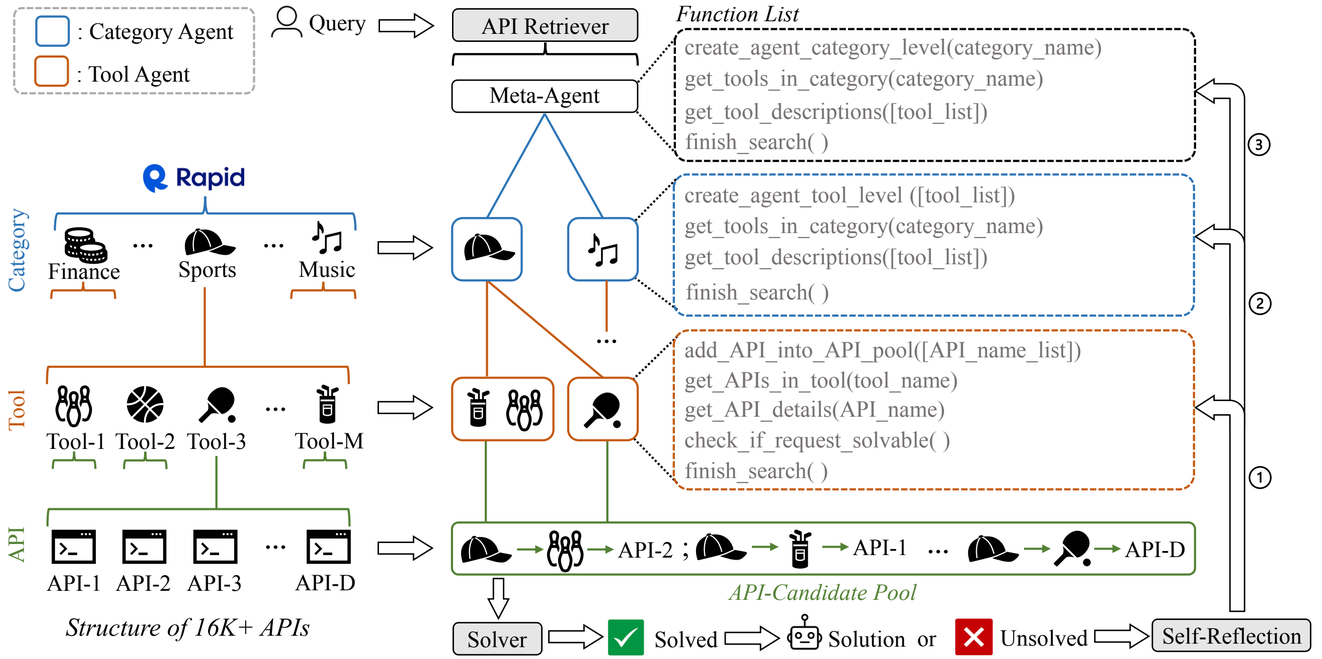
- First 3 levels build candidate tool set
  - Meta-Agent
  - Category-Agent
  - Tool-Agent
- Called recursively until tool set is defined (each search terminates with **`finish_…`**)

Type	Function Name	Functionality	Input	Output
Meta Agent	`create_agent_category_level`	Create a category agent.	Category name	Category agent
	`get_tools_in_category`	Get tool names under a category.	Category name	[Tool names]
	`get_tool_descriptions`	Get description of each tool.	[Tools]	[Tool descriptions]
	`finish_search`	Send out finish signal.	None	None
Category Agent	`create_agent_tool_level`	Create a tool agent.	[Tools]	Tool agent
	`get_tools_in_category`	Get tool names under a category.	Category name	[Tool names]
	`get_tool_descriptions`	Get description of each tool.	[Tools]	[Tool descriptions]
	`finish_search`	Send out finish signal.	None	None
Tool Agent	`add_API_into_API_pool`	Add APIs into candidate pool.	[APIs]	None
	`get_APIs_in_tool`	Get API names under a tool.	Tool name	[API names]
	`get_API_detail`	Get detail* of each API.	[API names]	[API details]
	`check_if_request_solvable`	Check whether the query is solvable using the current candidate pool.	None	True\False
	`finish_search`	Send out finish signal.	None	None

AnyTool: Self-Reflective, Hierarchical Agents for Large-Scale API Calls, 2024

AnyTool

- Final agent (LLM solver)
  - Solves task using toolset
- Self-Reflection (Reflextion) to correct output
- All models are prompted GPT-4 variants



	AnyTool
Turn	Single
Step	Multi
Learned	No
Objective	Goal State
Toolset	Open
Feedback	Yes

AnyTool: Self-Reflective, Hierarchical Agents for Large-Scale API Calls, 2024

LLMs with tools

- Allow LLM to output tool calls
  - Special tags
  - Special chat-template

References

- [1] Toolformer: Language Models Can Teach Themselves to Use Tools, Schick etal 2023
- [2] The Llama 3 Herd of Models, Dubey etal 2024
- [3] AnyTool: Self-Reflective, Hierarchical Agents for Large-Scale API Calls, 2024

Structured outputs

Philipp Krähenbühl, UT Austin

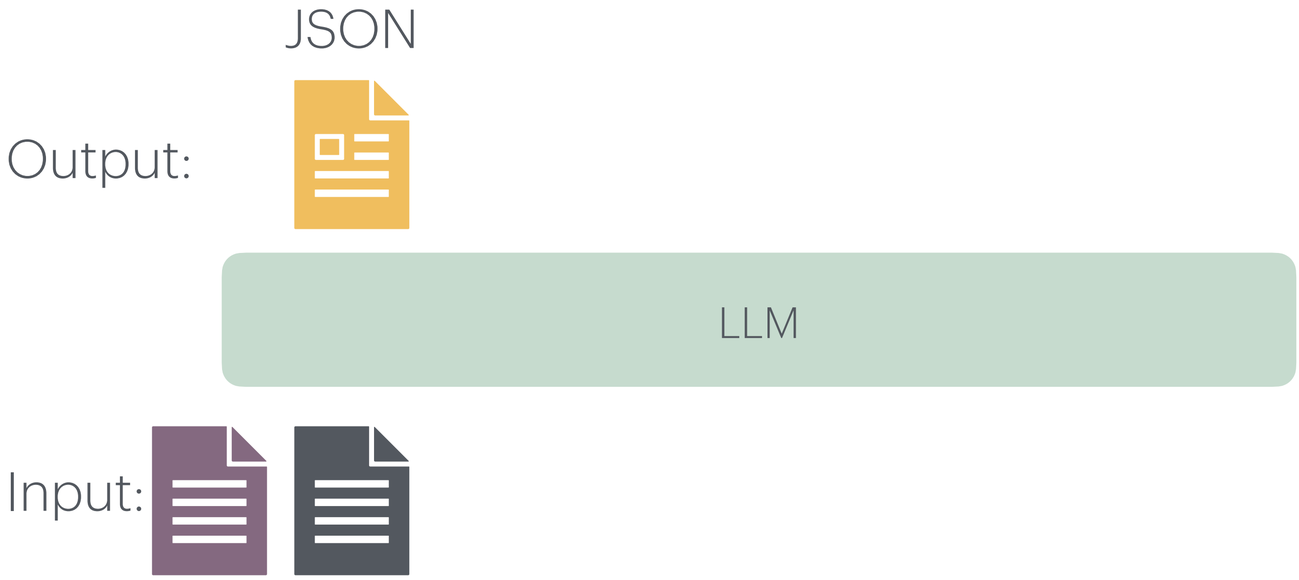
LLMs with tools

- Allow LLM to output tool calls
  - Special tags
  - Special chat-template

Structured output

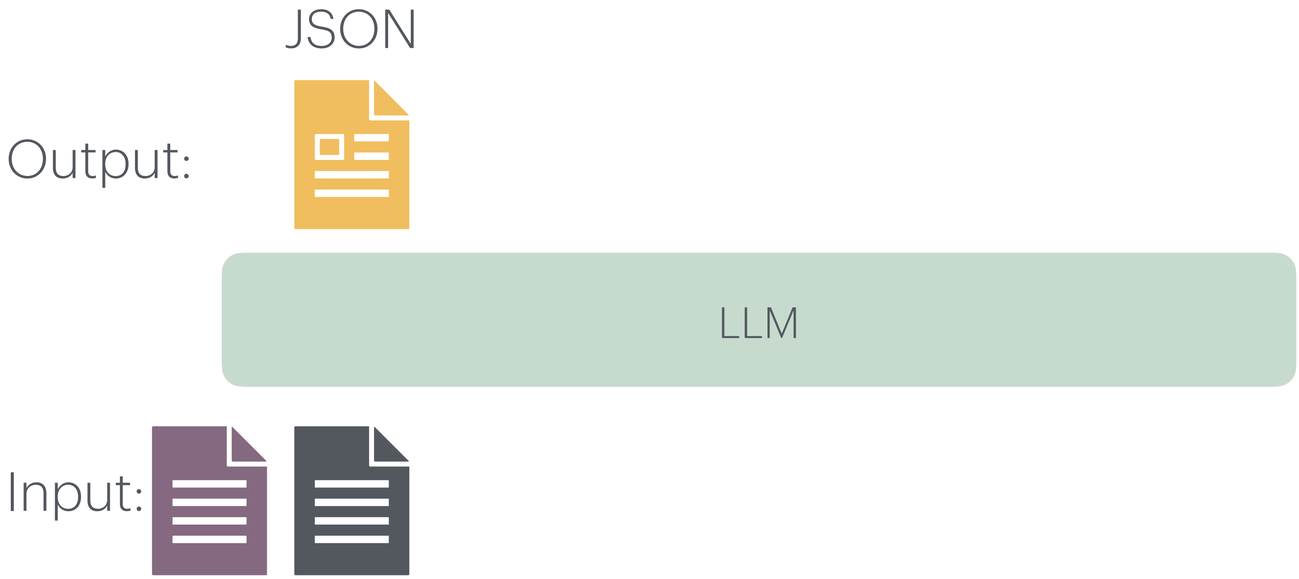
- What if we only want to **parse** output of LLM?
  - Option 1: In context example

    DEMO



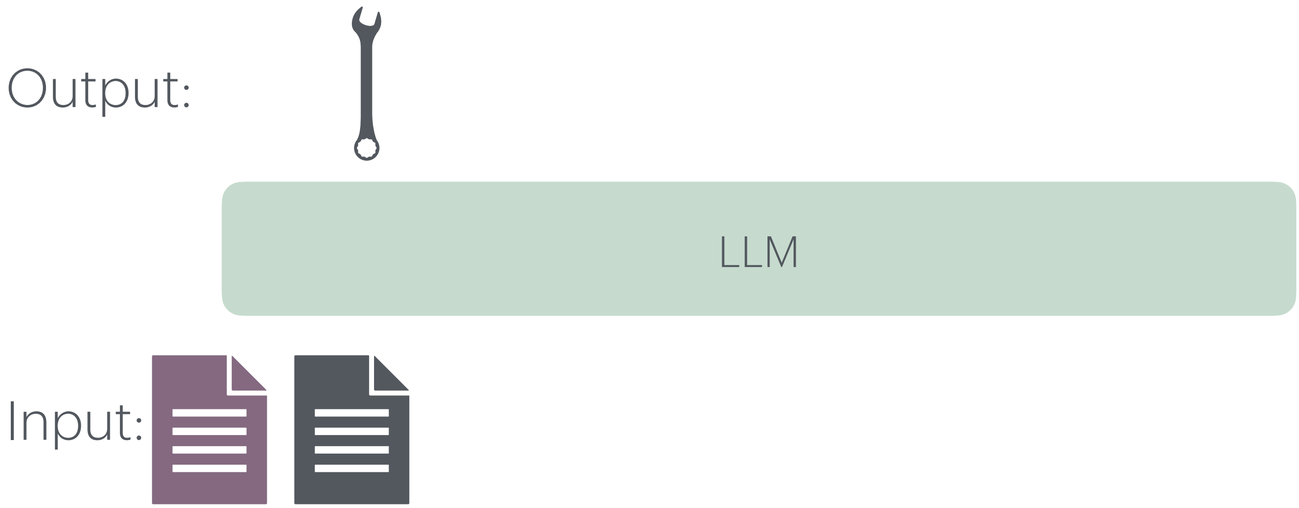
Structured output

- What if we only want to **parse** output of LLM?
  - Option 1: In context example
    - Parsing can easily fail (more later)



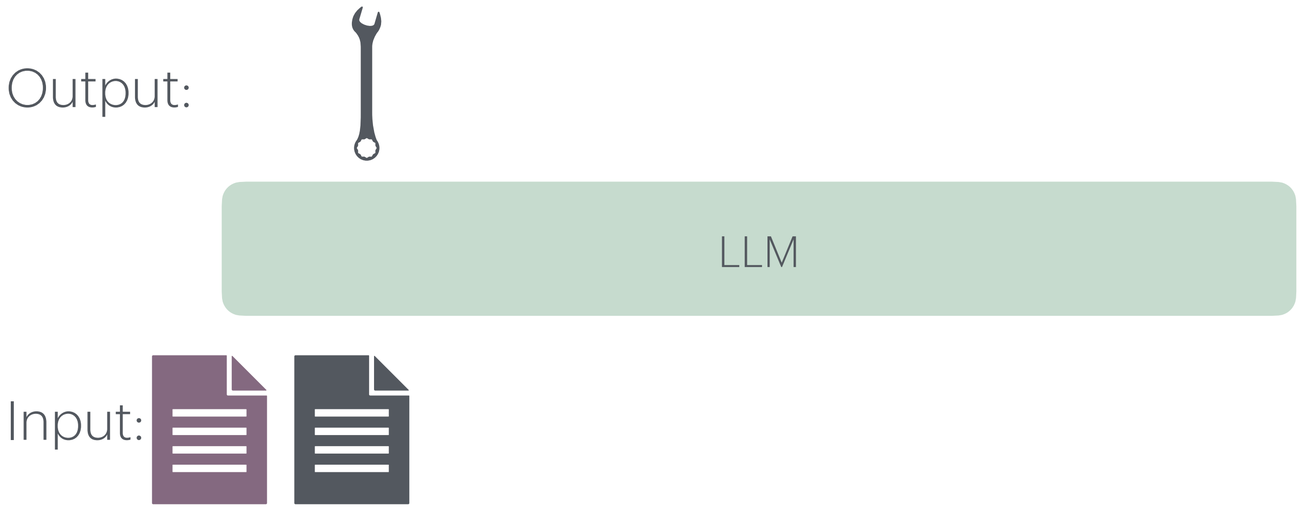
Structured output

- What if we only want to **parse** output of LLM?
  - Option 1: In context example
    - Parsing can easily fail (more later)
  - Option 2: Use a tool, arguments = json fields



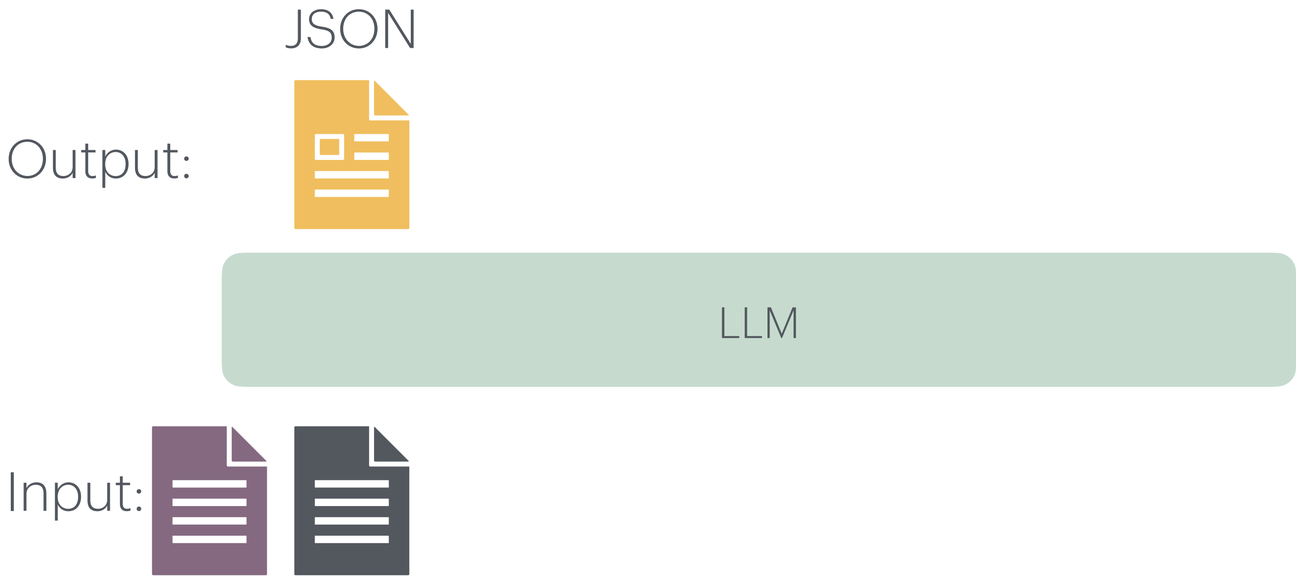
Structured output

- What if we only want to **parse** output of LLM?
  - Option 1: In context example
    - Parsing can easily fail (more later)
  - Option 2: Use a tool, arguments = json fields
    - More training data
    - Parsing might still fail



Structured output

- What if we only want to **parse** output of LLM?
  - Option 1.1
    - Write a robust parser (in python)
    - Let LLM know that you failed to parse
    - Hope for the best
  - Option 1.2: Constrain Decoding

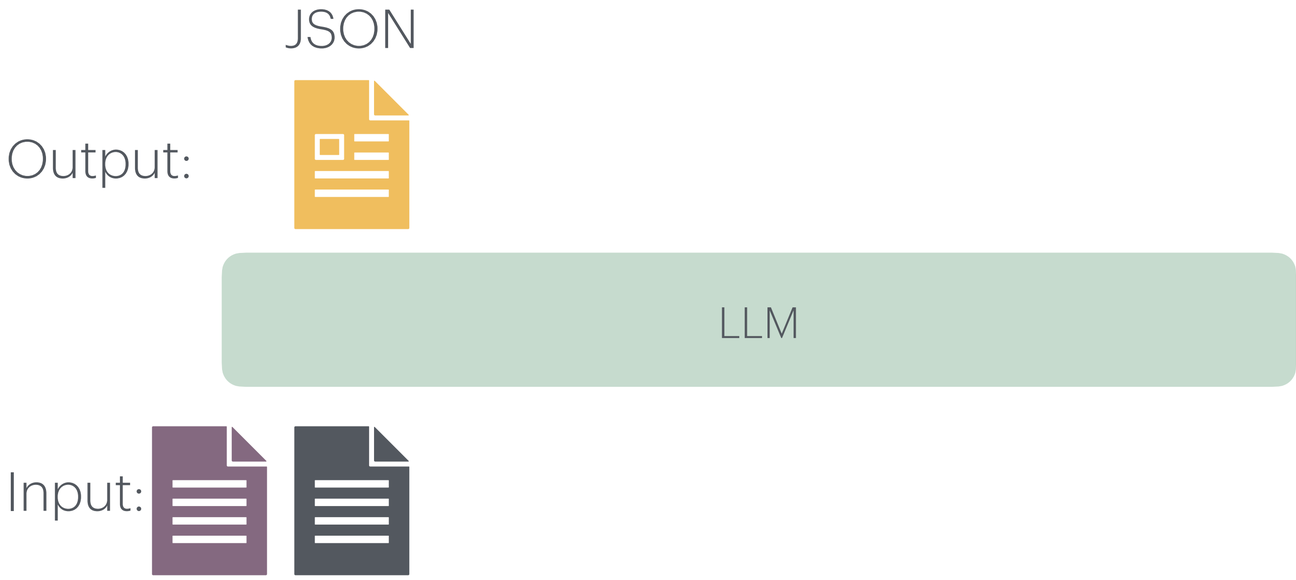


Constrained Decoding

Philipp Krähenbühl, UT Austin

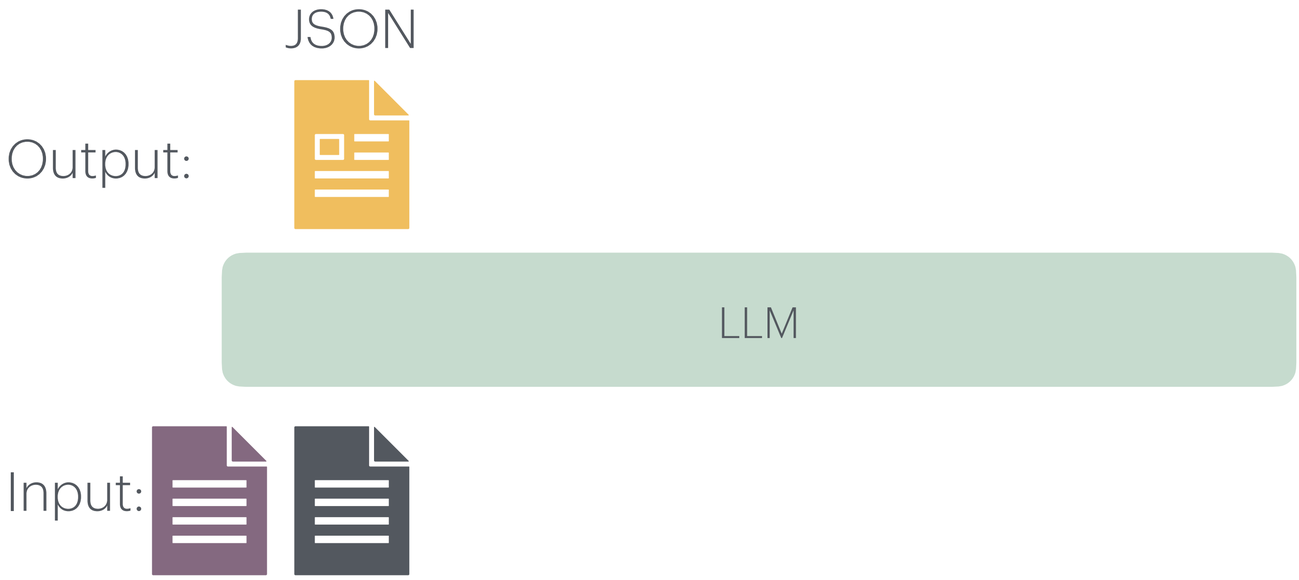
Structured output

- What if we only want to **parse** output of LLM?
  - Option 1.1
    - Write a robust parser (in python)
    - Let LLM know that you failed to parse
    - Hope for the best
  - **Option 1.2: Constrain output**



Constrained Decoding

- Change Sampling of LLM
  - $x_{i+1} \sim P(x_1 \ldots x_i)$
  - Only sample valid next tokens x_{i+1}
- How to define valid?



Constrained Decoding

- Change Sampling of LLM
  - $x_{i+1} \sim P(x_1 \ldots x_i)$
  - Only sample valid next tokens x_{i+1}
- Use a Context Free Grammar
  - Regex++

```
root   ::= object
value  ::= object | array | string | number | ("true" | "false"
| "null") ws

object ::=
  "{" ws (
            string ":" ws value
    ("," ws string ":" ws value)*
  )? "}" ws

array  ::=
  "[" ws (
            value
    ("," ws value)*
  )? "]" ws

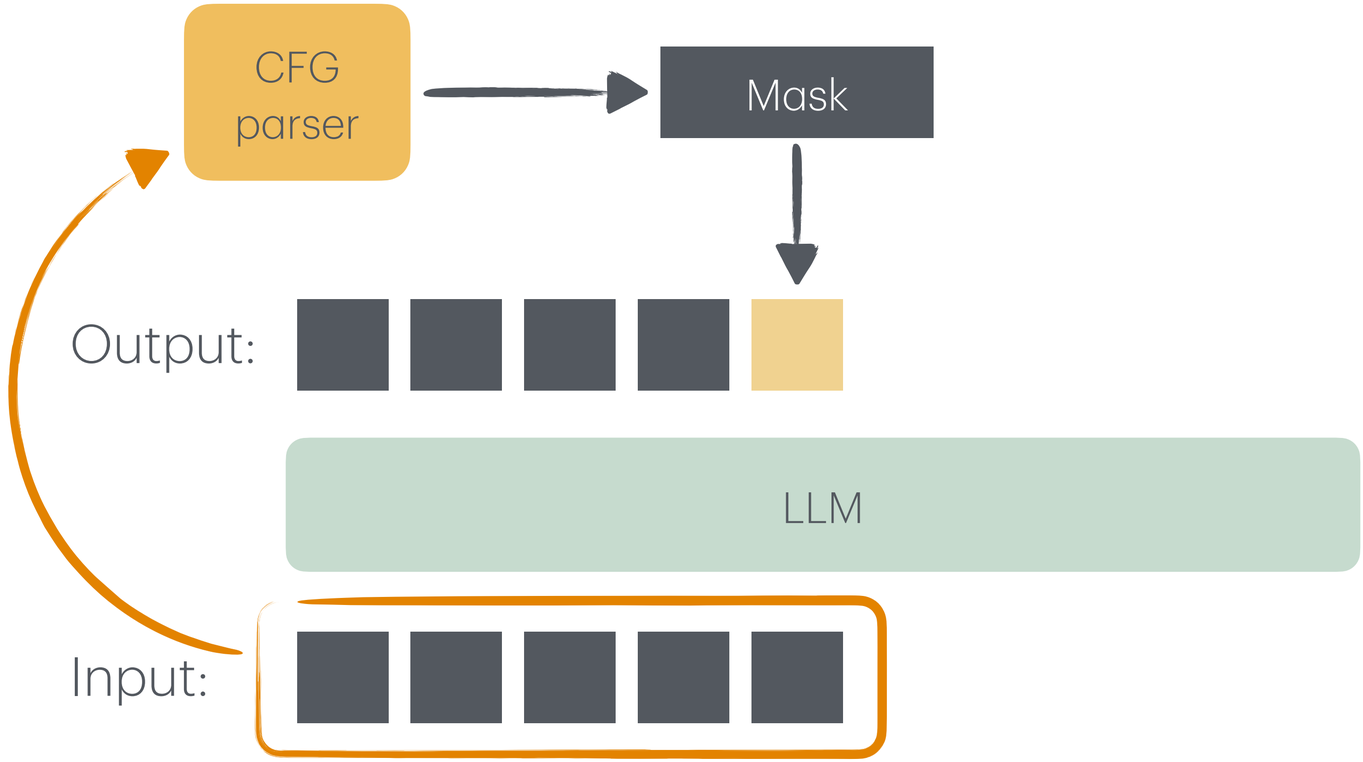
string ::=
  "\"" (
    [^"\\\x7F\x00-\x1F] |
    "\\" (["\\bfnrt] | "u" [0-9a-fA-F]{4}) # escapes
  )* "\"" ws

number ::= ("-"? ([0-9] | [1-9] [0-9]{0,15})) ("." [0-9]+)?
([eE] [-+]? [0-9] [1-9]{0,15})? ws

# Optional space: by convention, applied in this grammar after
literal chars when allowed
ws ::= | " " | "\n" [ \t]{0,20}
```

https://github.com/ggerganov/llama.cpp

Constrained Decoding



- CFG parser
  - Computes valid completions $v \in V_{i+1}$
- Only sample valid next tokens $P \rightarrow \hat{P}$
  - $\hat{P}(v) = 0 \quad \forall v \notin V_{i+1}$
  - $x_{i+1} \sim \hat{P}(x_1 \ldots x_i)$

Algorithm 1 Constrained Decoding

Input: Checker C, LLM f, Tokenized Prompt x
Output: Completion o adhering to C

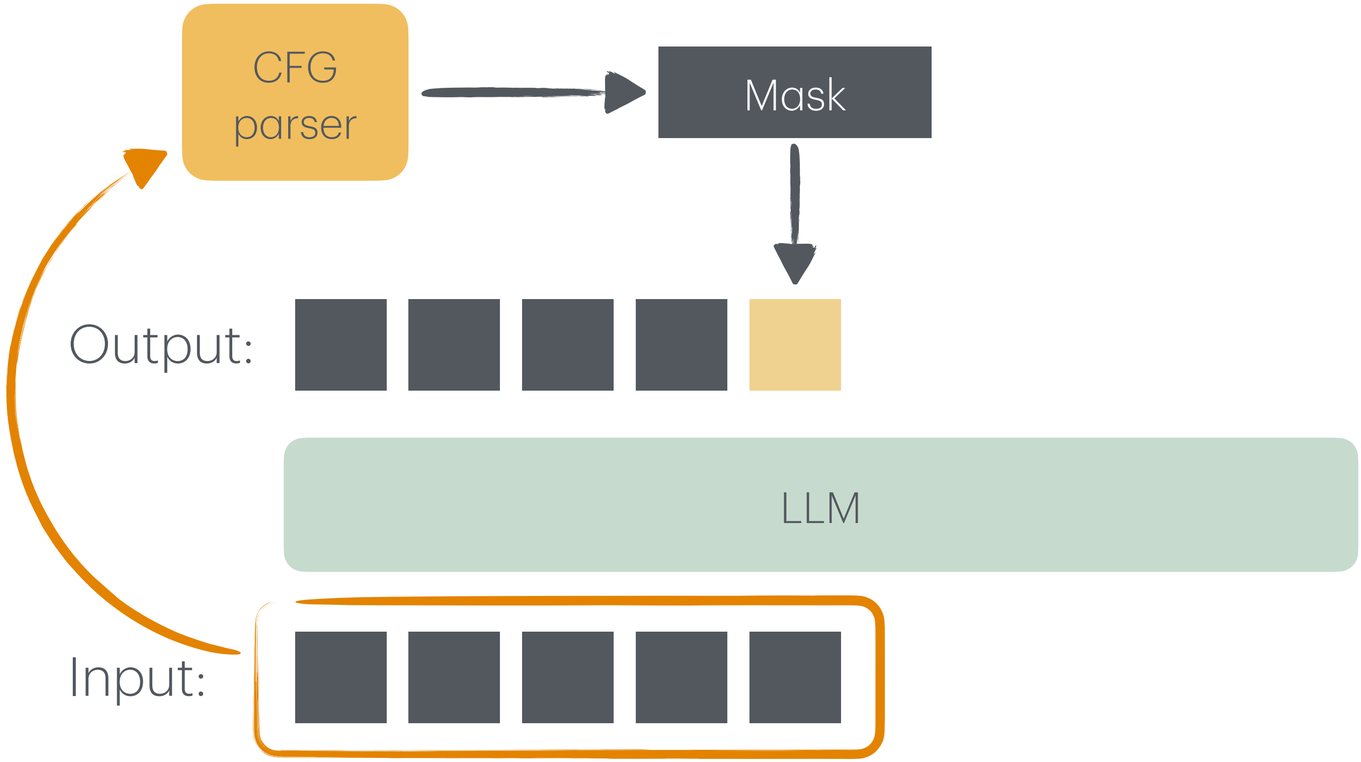
```
o ← []
C.init()
loop
    C.update(o)                    // advance state of C
    m ← C.mask()                   // compute mask
    v ← f(x + o)                   // compute logits
    v' ← m ⊙ v'
    t ← decode(α')                 // e.g., argmax or sample
    if t = EOS then break
    o.append(t)
end loop
return o                           // optionally detokenize
```

Synchromesh: Reliable code generation from pre-trained language models, Poesia etal 2022

Constrained Decoding

Issue

- Constraints: On text / output
- Decoding: on tokens
  - Many tokens streams = one text
  - Model only likes one



Guiding LLMs The Right Way: Fast, Non-Invasive Constrained Generation, Beurer-Kellner etal 2024

Constrained Decoding

Solution

- Lift CFG to token level

Algorithm 2 Construct Terminal Tree

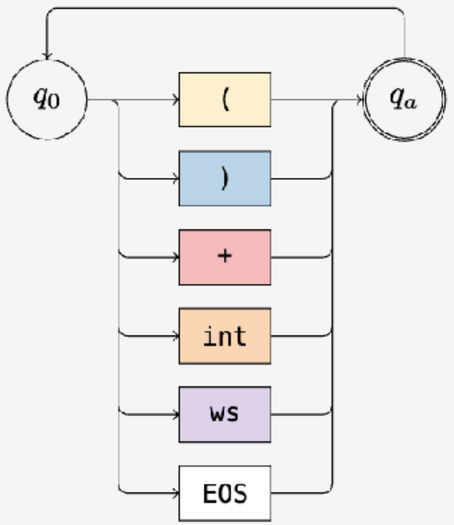
Input: CFG G, Alphabet Σ, Vocabulary $\mathcal{V}$
Output: Scanner S

$\mathcal{T} = \{\}$
for all $q \in S.\texttt{states}()$ **do**
  $\alpha \leftarrow q.\texttt{subterminal}()$ // get current (sub)terminal
  for all $l \in \mathcal{V}$ **do**
    $\{\alpha_1^j, \ldots, \alpha_{m_j}^j\}_j \leftarrow q.\texttt{traverse}(l)$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{(\alpha_1^j, \ldots, \alpha_{m_j}^j), l\}_j$
  end for
  $T_q \leftarrow \texttt{PrefixTree}(\mathcal{T})$
end for

(a) Example Grammar

```
E = E | E + E | (E) | int
int = ([1-9][0-9]*)|(0+)
```

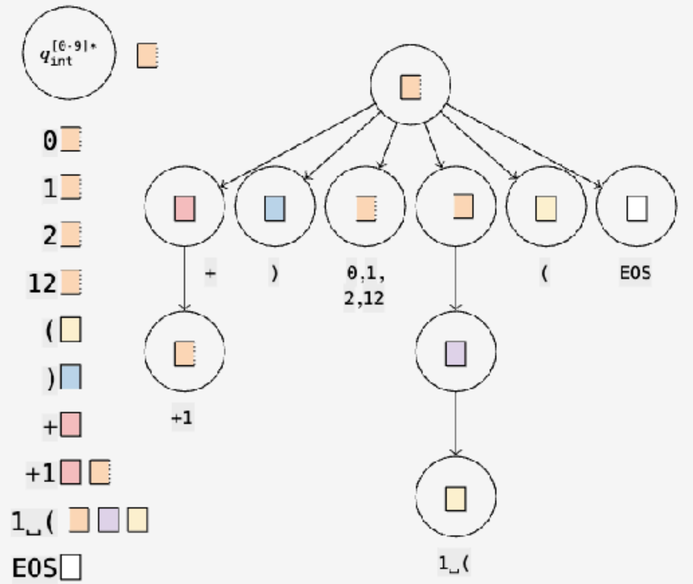
(b) Character Scanner



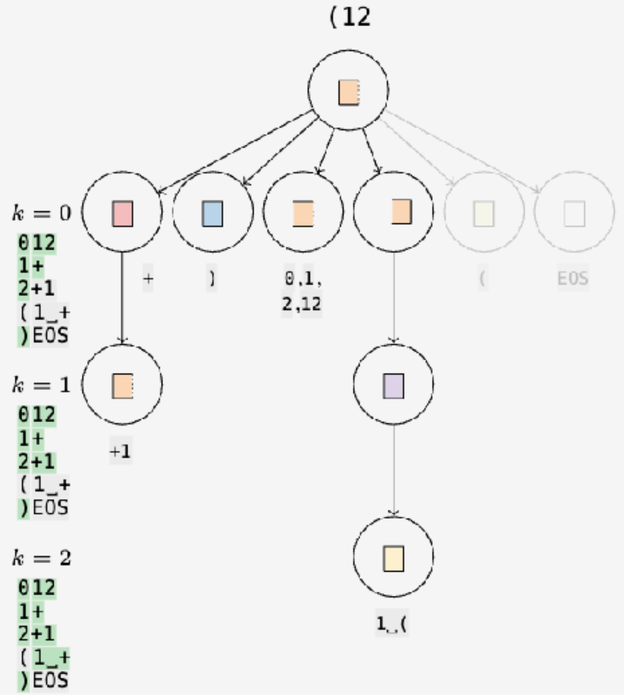
(c) Vocabulary

$$\mathcal{V} = \{0, 1, 2, 12,), (, +, +1\,1\_(, \texttt{EOS}\}$$

(d) Vocabulary-aligned Subterminal Tree (offline, per node)



(e) Parser (online)



Guiding LLMs The Right Way: Fast, Non-Invasive Constrained Generation, Beurer-Kellner etal 2024

Constrained Decoding

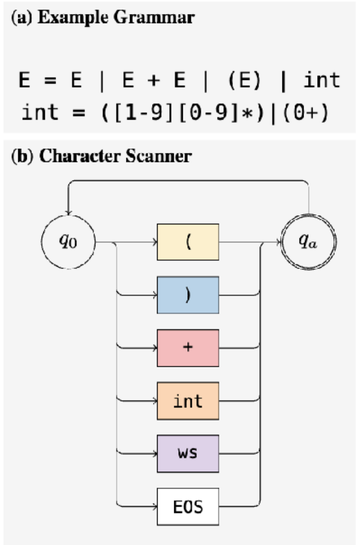
Solution

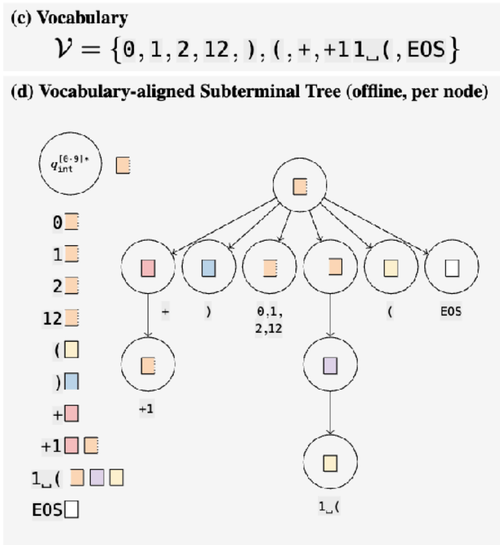
- Very complex to implement

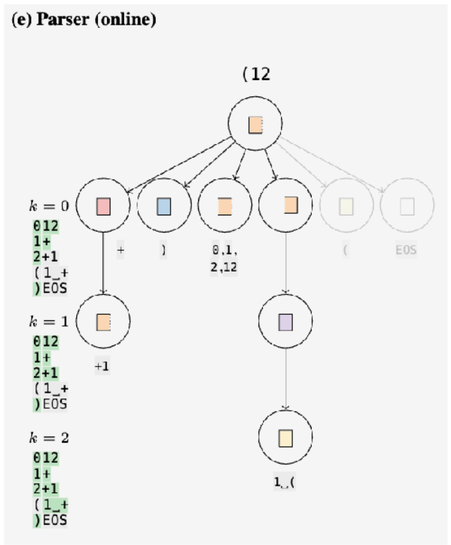
Algorithm 2 Construct Terminal Tree

Input: CFG G, Alphabet Σ, Vocabulary $\mathcal{V}$
Output: Scanner S

$\mathcal{T} = \{\}$
for all $q \in S.\texttt{states}()$ **do**
  $\alpha \leftarrow q.\texttt{subterminal}()$ // get current (sub)terminal
  for all $l \in \mathcal{V}$ **do**
    $\{\alpha_1^j, \ldots, \alpha_{m_j}^j\}_j \leftarrow q.\texttt{traverse}(l)$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{(\alpha_1^j, \ldots, \alpha_{m_j}^j), l\}_j$
  end for
  $T_q \leftarrow \texttt{PrefixTree}(\mathcal{T})$
end for





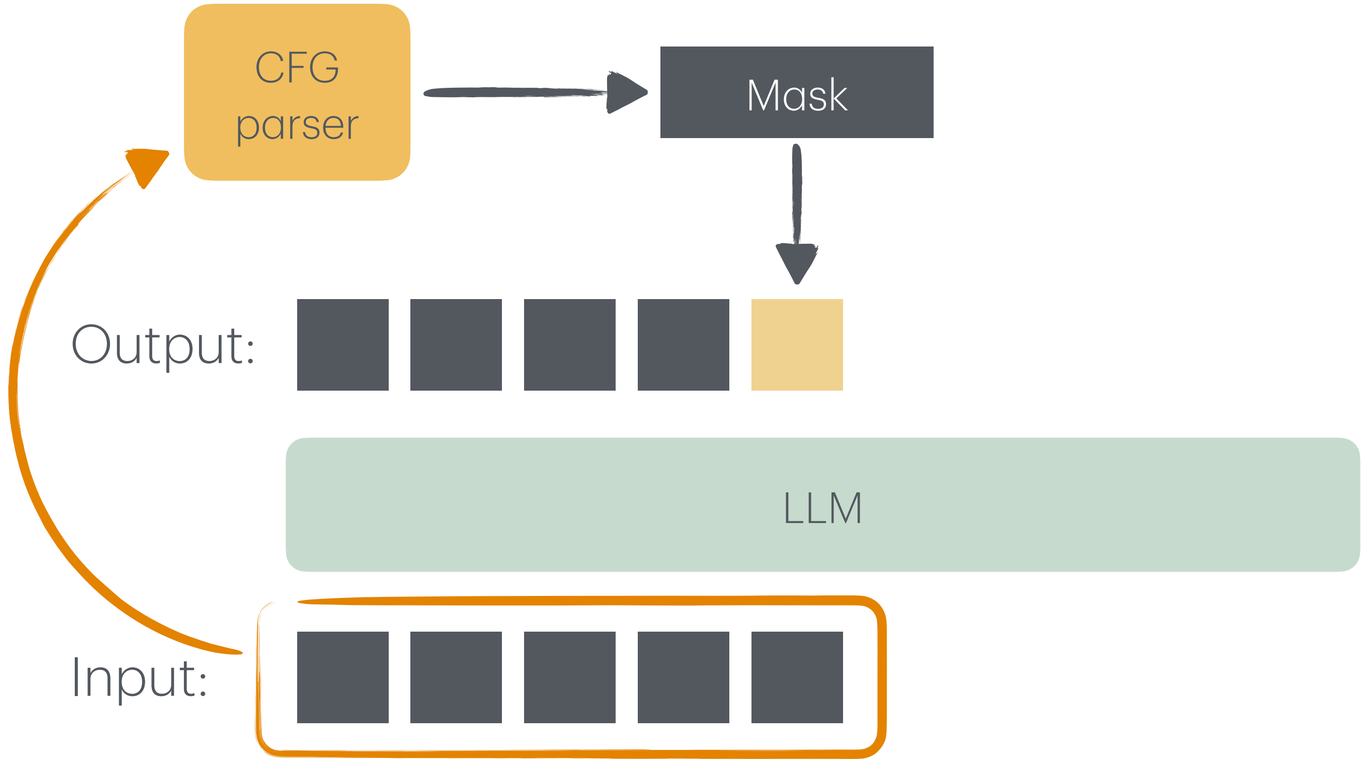


Guiding LLMs The Right Way: Fast, Non-Invasive Constrained Generation, Beurer-Kellner etal 2024

Constrained Decoding

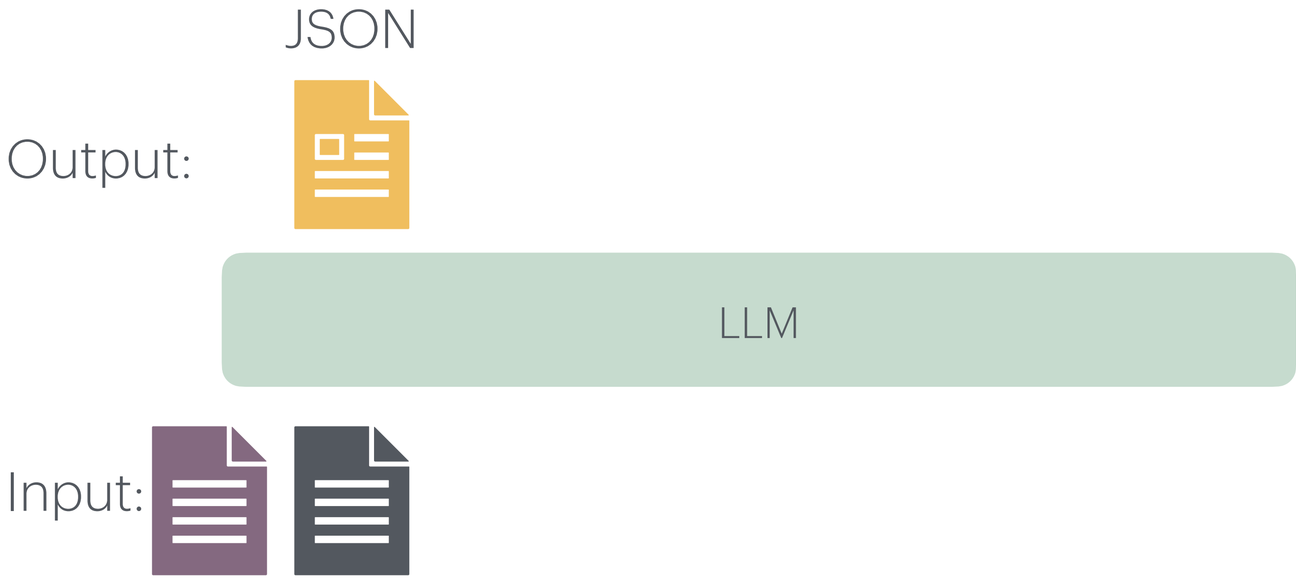
Issue

- Outputs are Biased
- Example CFG
  - S -> aSb | Sc | ε
  - $$P(x_{i+1} \mid x_1 \ldots x_i) = \begin{cases} 0.9 & \text{if } x_{i+1} = a \\ 0 & \text{if } x_{i+1} = b \\ 0.1 & \text{if } x_{i+1} = c \end{cases}$$
  - Sampling will never stop: aaaaaaaa



Structured output

- What if we only want to **parse** output of LLM?
  - Option 1.1
    - Write a robust parser (in python)
    - Let LLM know that you failed to parse
    - Hope for the best
  - Option 1.2: Constrain output



References

- [1] Synchromesh: Reliable code generation from pre-trained language models, Poesia etal 2022

- [2] Guiding LLMs The Right Way: Fast, Non-Invasive Constrained Generation, Beurer-Kellner etal 2024