Machine Programming

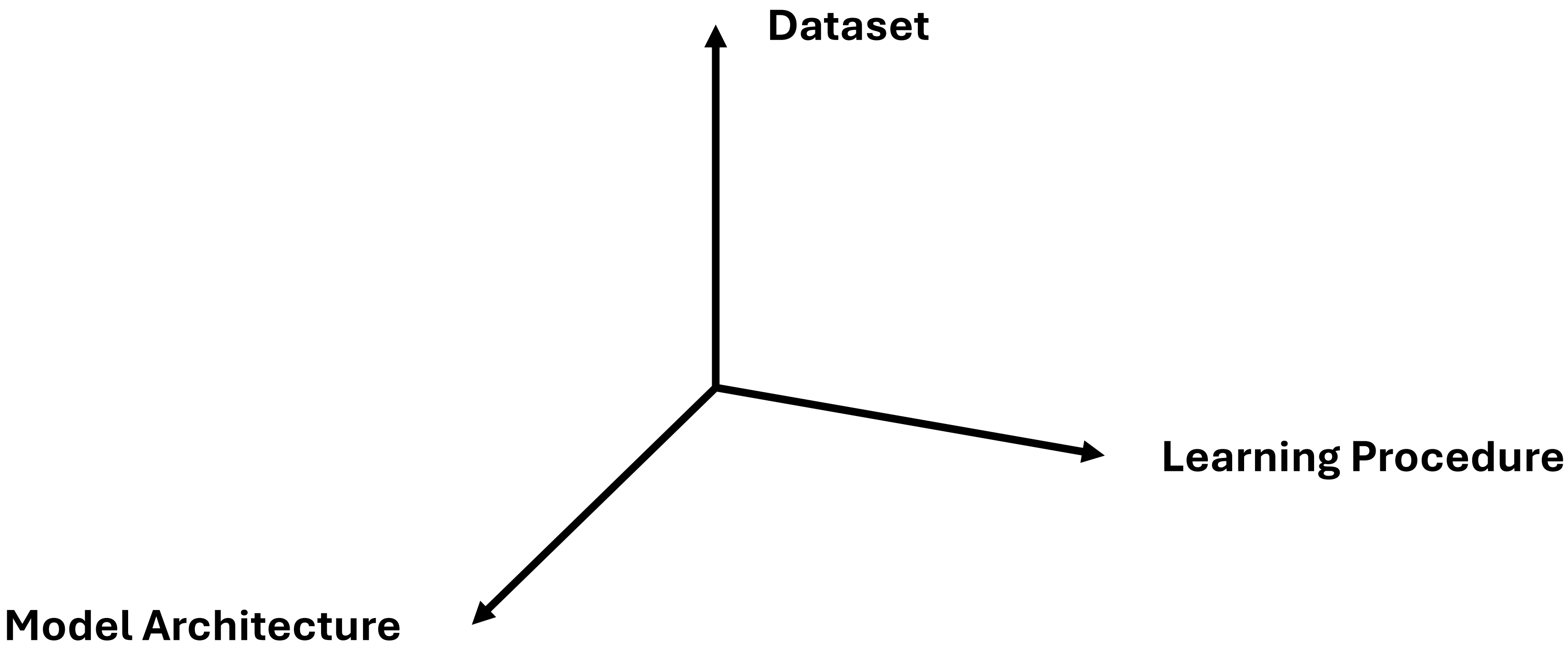
Lecture 14 – Post-training of Coding Language Models: SFT & RL

Ziyang Li

Logistics – Week 8

- Assignment 3: Coding LLM Agents
  - https://github.com/machine-programming/assignment-3
  - Fully functional web-app agent. Due: Oct 23 (Thu)
- Oral presentation sign up sheet
  - Please sign up! (16/19 received)
- Forming groups for your final projects!
  - Form a group of 2-3 before This Sunday (Oct 19)

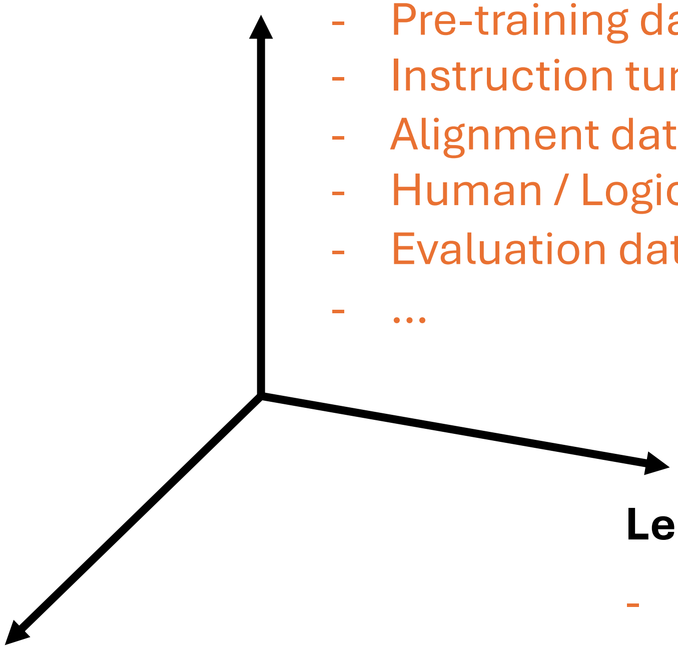
How to obtain a “good enough” LLM



How to obtain a “good enough” LLM

Dataset

- Pre-training dataset / Fine-tuning dataset
- Instruction tuning dataset
- Alignment dataset
- Human / Logical feedback dataset
- Evaluation dataset
- …



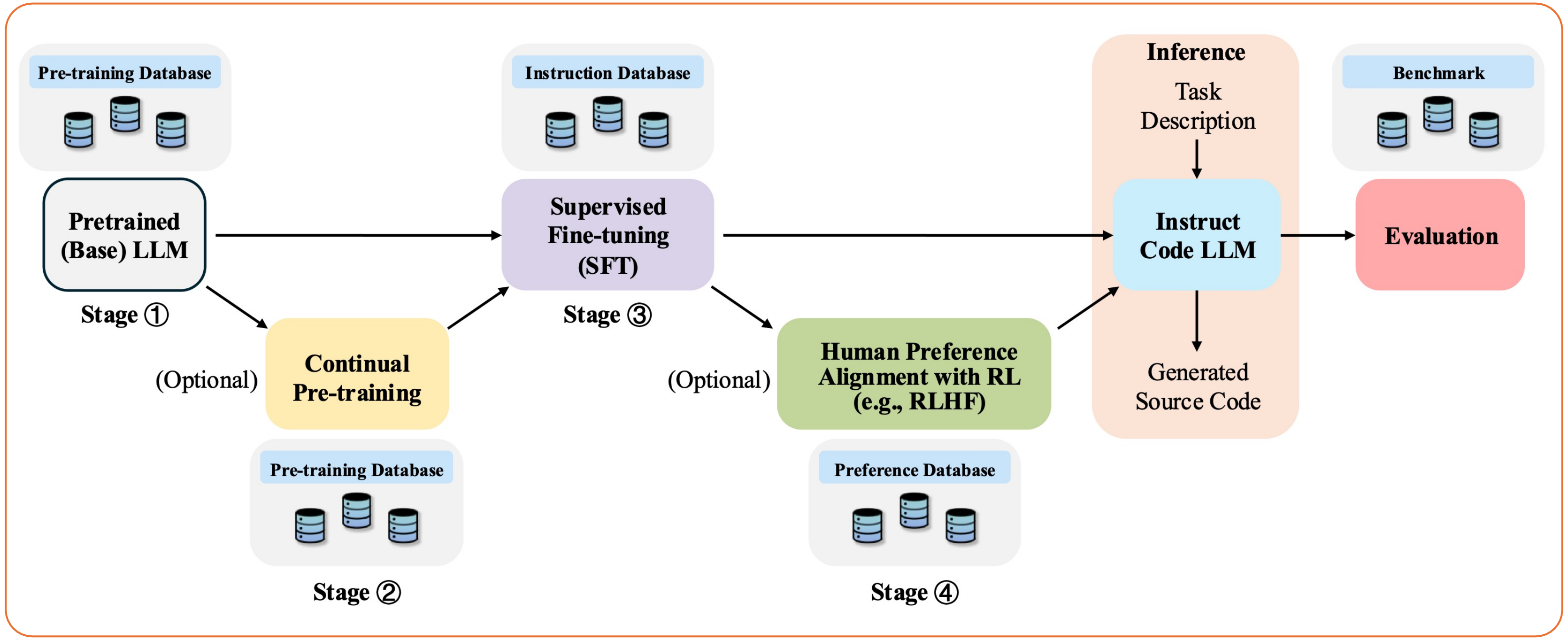
Learning Procedure

- Optimization objectives
- Learning algorithm (SFT, RL, etc.)
- Continual learning, Curriculum learning
- Staged learning
- …

Model Architecture

- Encoder-decoder models
- Decoder-only models
- Hyper-parameter tuning
- …

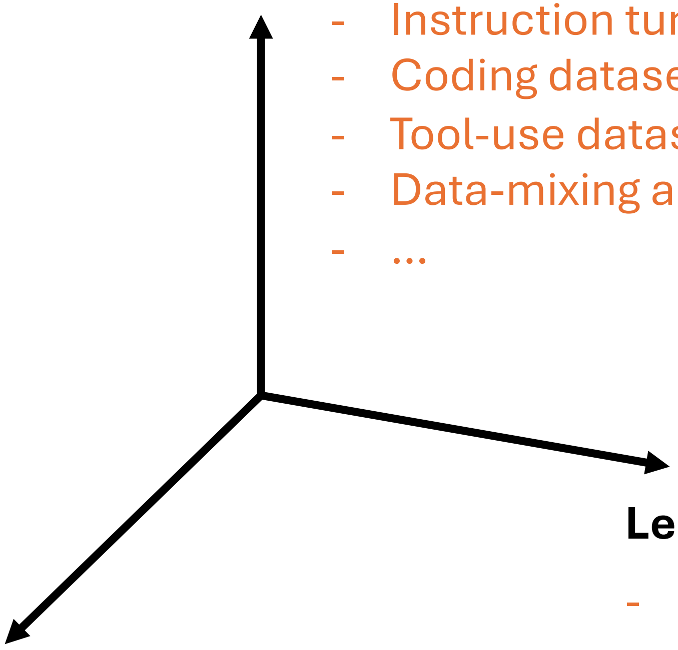
How to obtain a “good enough” LLM



Post-Training: Supervised Fine-tuning (SFT)

Dataset

- Instruction tuning dataset
- Coding dataset
- Tool-use dataset
- Data-mixing and Curriculum for SFT
- …



Learning Procedure

- **(FIXED)** Next token prediction objective
- Full-parameter optimization
- Parameter-efficient fine-tuning (PEFT)
- Component freezing
- Hyper-parameter

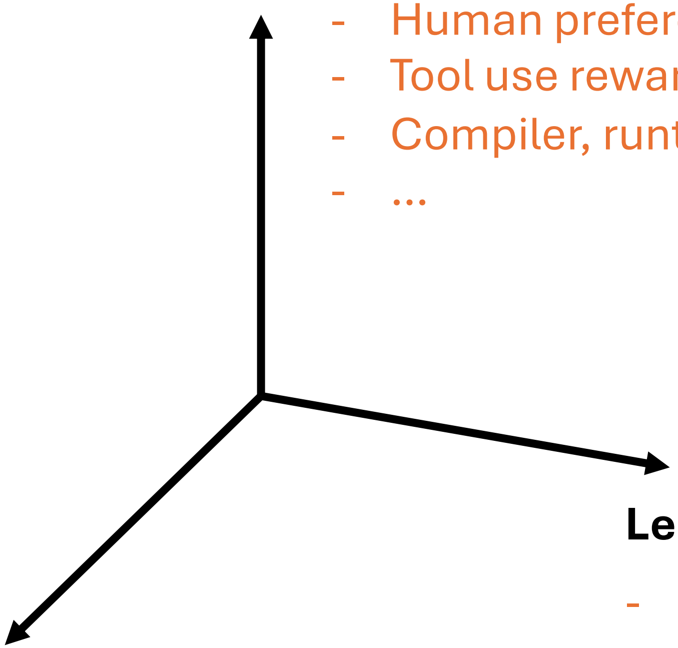
Model Architecture

- **(FIXED)** Language Model
- Adaptation layers (during PEFT)

Post-Training: Reinforcement Learning (RL)

Dataset

- Human preference reward
- Tool use reward
- Compiler, runtime, testing feedback
- ...



Learning Procedure

- PPO, GRPO, DPO
- Reward provider: model, tool, logical spec
- Gradient-based RL?
- ...

Model Architecture

- **(FIXED)** Language Model

Post-Training: Reinforcement Learning (RL)

Dataset

- Instruction tuning dataset
- Coding dataset
- Tool-use dataset
- Data-mixing and Curriculum for SFT
- …

Learning Procedure

- **(FIXED)** Next token prediction objective
- Full-parameter optimization
- Parameter-efficient fine-tuning (PEFT)
- Component freezing
- Hyper-parameter

Model Architecture

- **(FIXED)** Language Model

Dataset

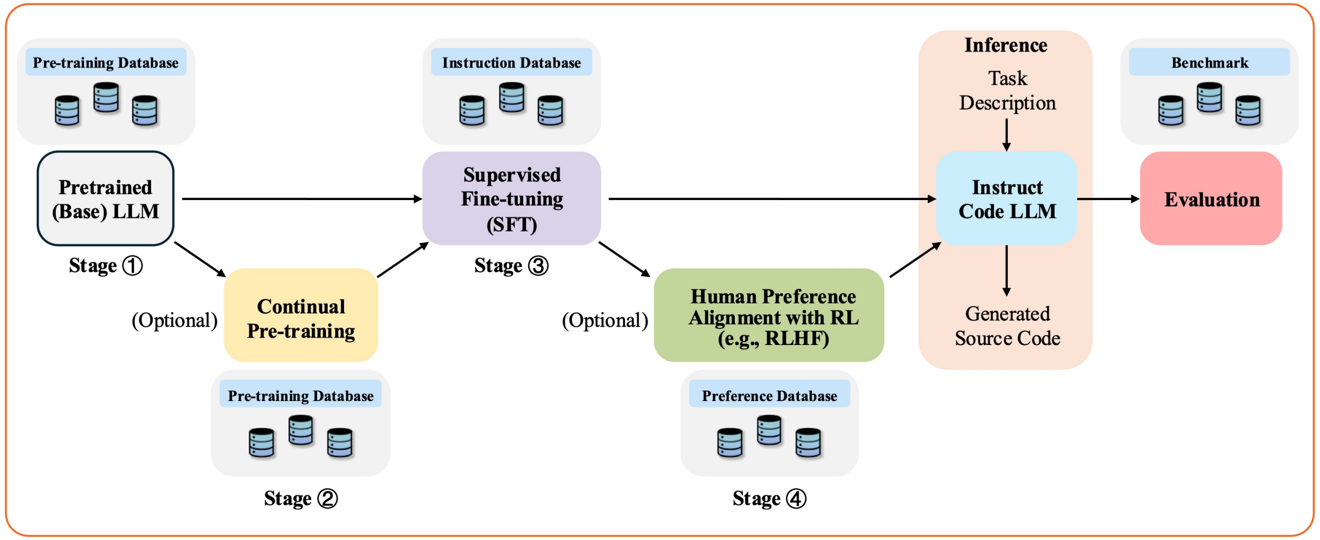
- Human preference reward
- Tool use reward
- Compiler, runtime, testing feedback
- …

Learning Procedure

- PPO, GRPO, DPO
- Reward provider: model, tool, logical spec
- Gradient-based RL?
- …

Today's Agenda

- Supervised Fine-tuning
  - Learning objective
  - Dataset
- Reinforcement Learning
  - Learning objective
  - Optimization
  - Dataset



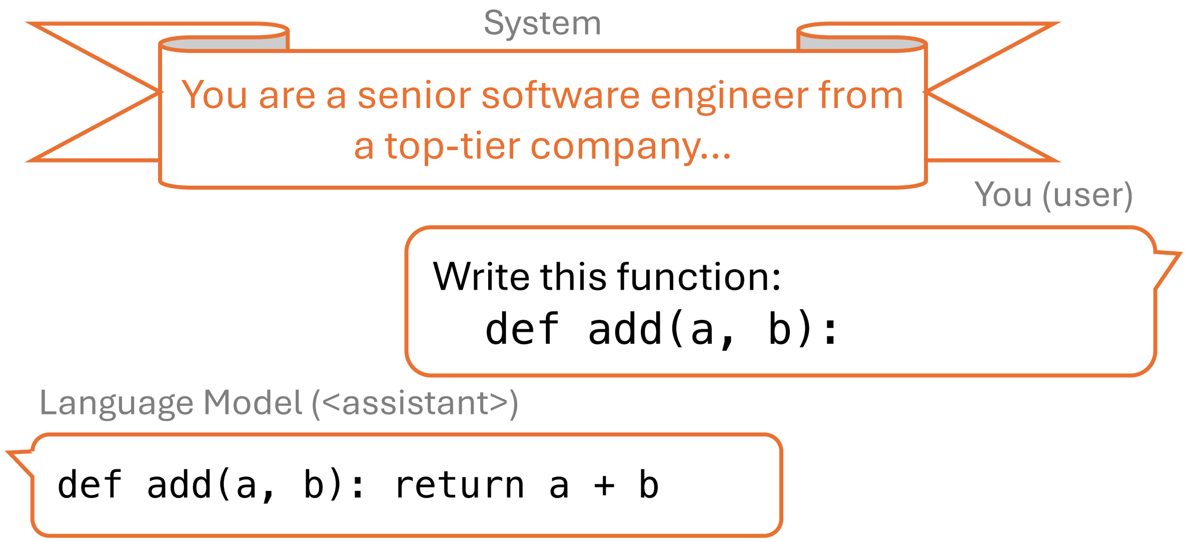
SFT: Learning Objectives

- Pre-training: General understanding of language
- SFT: Aligns with human intent
- High-level Objective:
  - Instruction following / dialog
  - Reasoning and chain-of-thought
  - Tool use and agentic protocol-following
  - Code generation (completion + Infilling/FIM)
- Low-level Objective: Next-token prediction

$$\mathcal{L}(\mathbf{x};\theta) = \sum_{i=1}^{n} -\log P_\theta(x_i \mid \mathbf{x}_{<i})$$

SFT: Learning Objectives

- Instruction following
  - Utilizing special tokens such as <system>, <user>, and <assistant>
  - Instruction: system prompts and user instructions
  - From completion style to multi-step turns from <assistant> token



Training language models to follow instructions with human feedback

Long Ouyang[*] Jeff Wu[*] Xu Jiang[*] Diogo Almeida[*] Carroll L. Wainwright[*]

Pamela Mishkin[*] Chong Zhang Sandhini Agarwal Katarina Slama Alex Ray

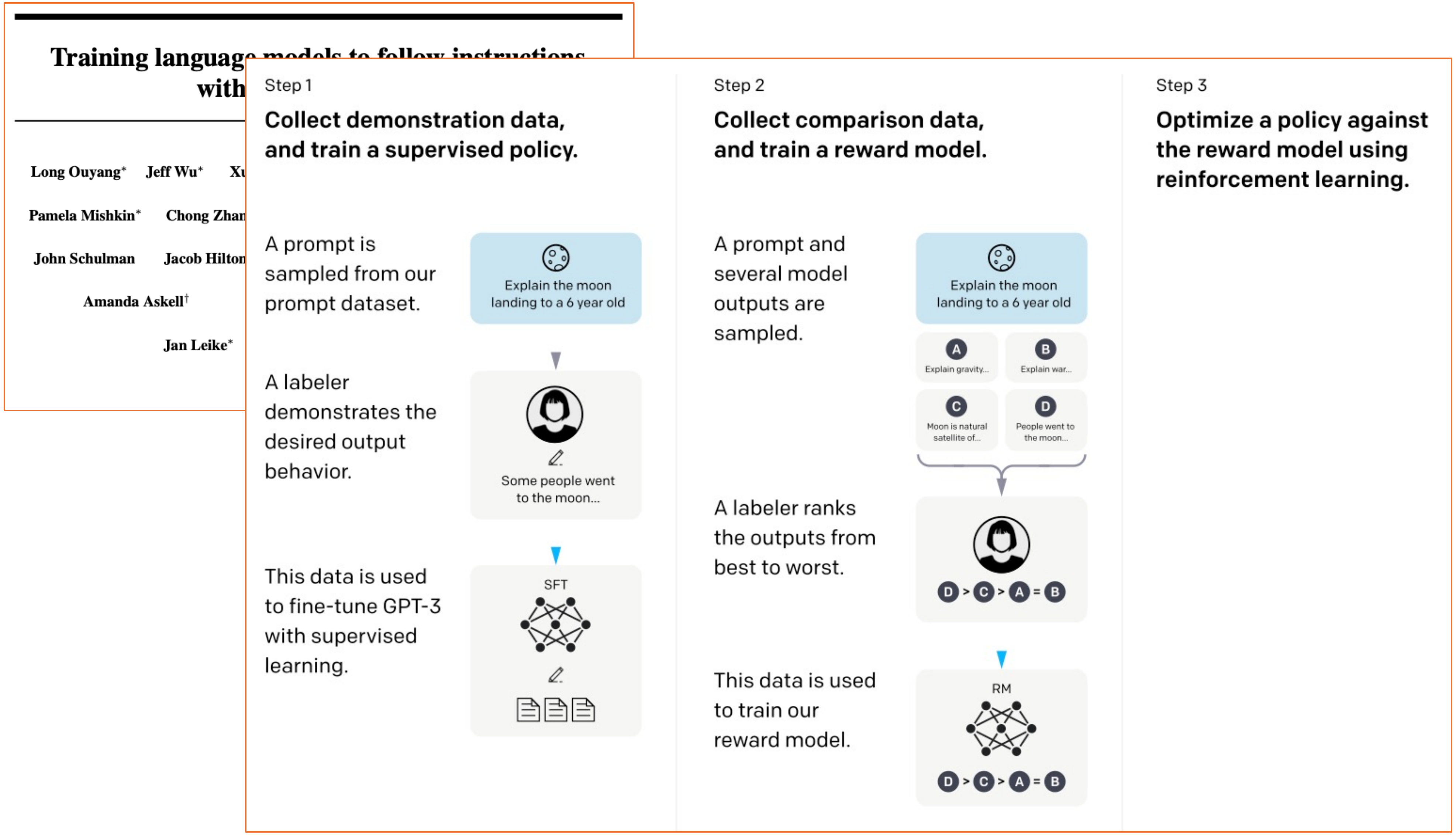
John Schulman Jacob Hilton Fraser Kelton Luke Miller Maddie Simens

Amanda Askell[†] Peter Welinder Paul Christiano[*†]

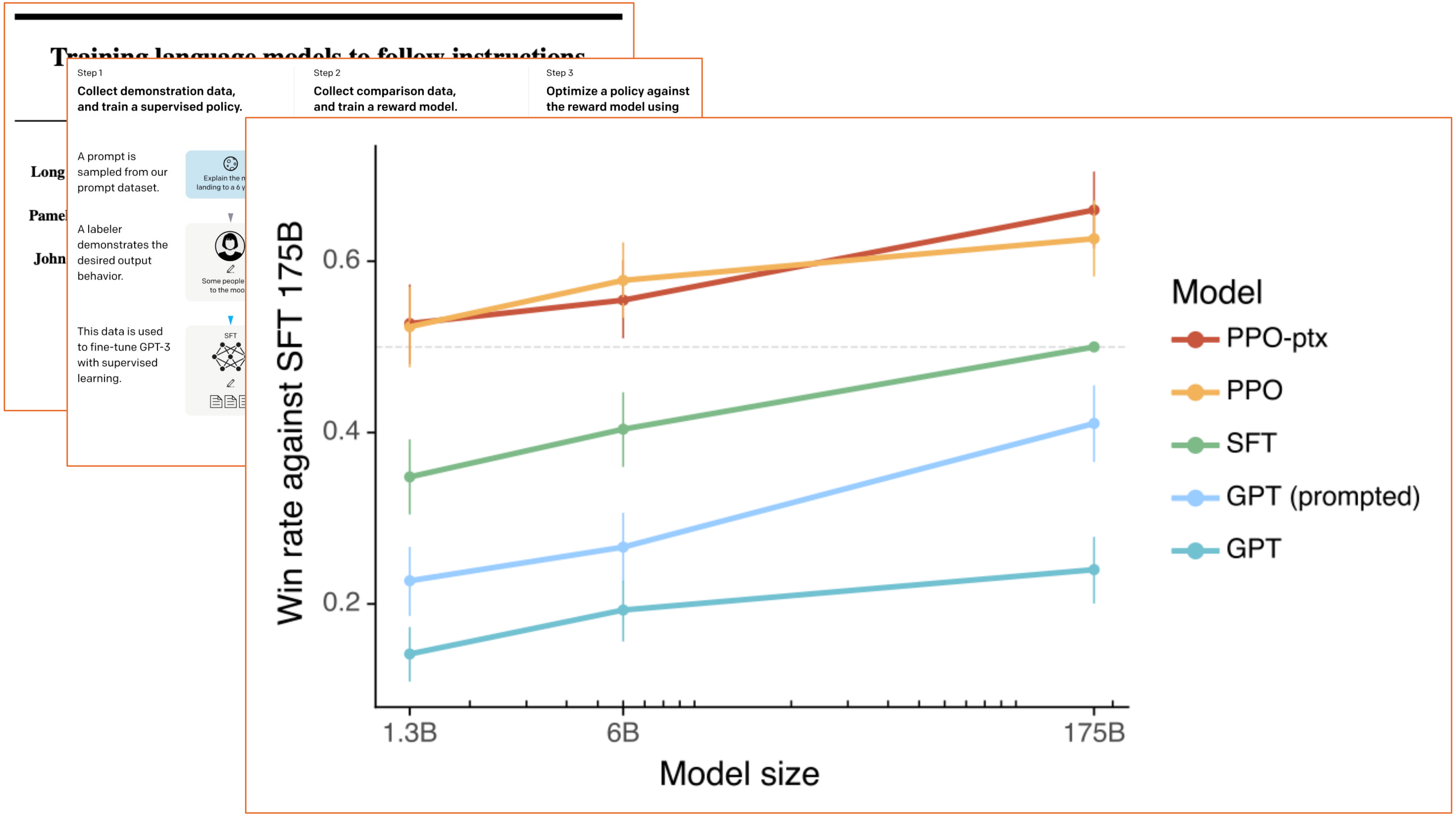
Jan Leike[*] Ryan Lowe[*]

OpenAI

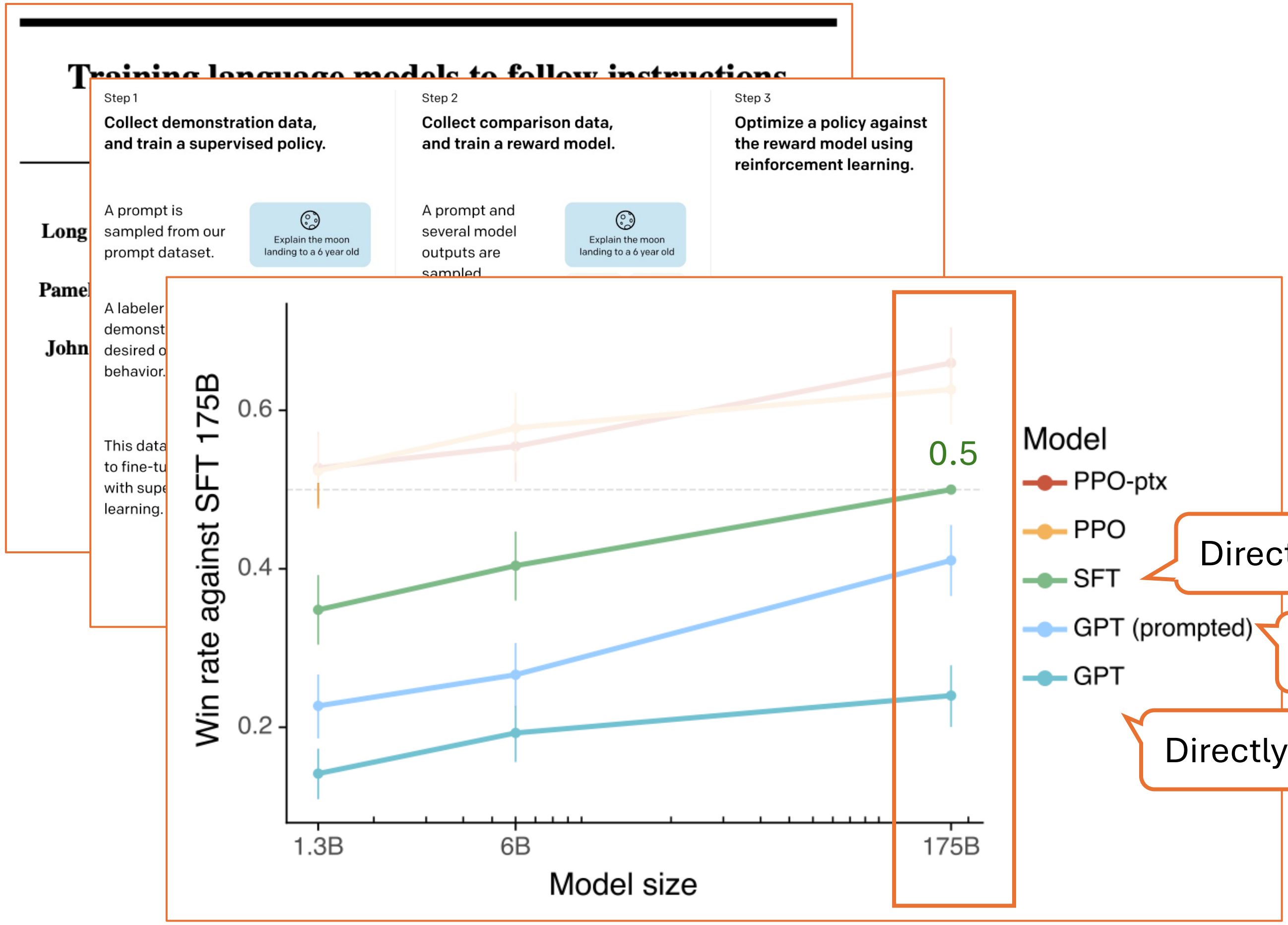
Traning language models to follow instructions with human feedback, Ouyang et. al., OpenAI, 2022



Traning language models to follow instructions with human feedback, Ouyang et. al., OpenAI, 2022



Traning language models to follow instructions with human feedback, Ouyang et. al., OpenAI, 2022



Directly giving human instruction, but model is SFT-ed

Giving human instruction wrapped by a prompt

Directly giving human instruction

Win rate: human evaluation of which answer is more preferrable, target or SFT 175B;
The score for SFT 175B would be 0.5 by construction

Traning language models to follow instructions with human feedback, Ouyang et. al., OpenAI, 2022

SFT: Learning Objectives

- Reasoning and Chain-of-thought
  - Learn from existing reasoning chain in a fully-supervised manner
- Intuition: Imitate “reasoning” from the dataset
- Problem: RL might be better fit
  - Single-path bias, Teacher-forcing mismatch, outcome misalignment, …

Orca: Progressive Learning from Complex Explanation Traces of GPT-4

Subhabrata Mukherjee[*†], Arindam Mitra[*]

Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, Ahmed Awadallah

Microsoft Research

Orca: Progressive Learning from Complex Explanation Traces of GPT-4, Mukherjee et. al., Microsoft Research, 2023

Orca: Progressive Learning from Complex Explanation Traces of GPT-4

Subhabrata Mukherjee*†, Arindam Mitra*

Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, Ahmed Awadallah

Model	Tuning Method	Data Size	Teacher
Alpaca	Simple Instructions / Self-instruct	52K	text-da-vinci-003
Vicuna	User Instructions / Natural	70K	ChatGPT
Dolly	User Instructions / Natural	15K	Human
WizardLM	Complex Instructions / Evol-instruct	250K	ChatGPT
Orca	Complex Instructions / Explanations	5M	ChatGPT (5M) $\cap$ GPT-4 (1M)

Table 1: Overview of popular models instruction tuned with OpenAI large foundation models (LFMs). Orca leverages complex instructions and explanations for progressive learning.

Orca: Progressive Learning from Complex Explanation Traces of GPT-4, Mukherjee et. al., Microsoft Research, 2023

Orca: Progressive Learning from Complex Explanation Traces of GPT-4

Ganesh

Model	Tuning Method	Data Size	Teacher
Alpaca	Simple Instructions / Self-instruct	52K	text-da-vinci-003
Vicu			
Dol			
Wizard			
Orc			

Table 1: Ove
(LFMs). Or

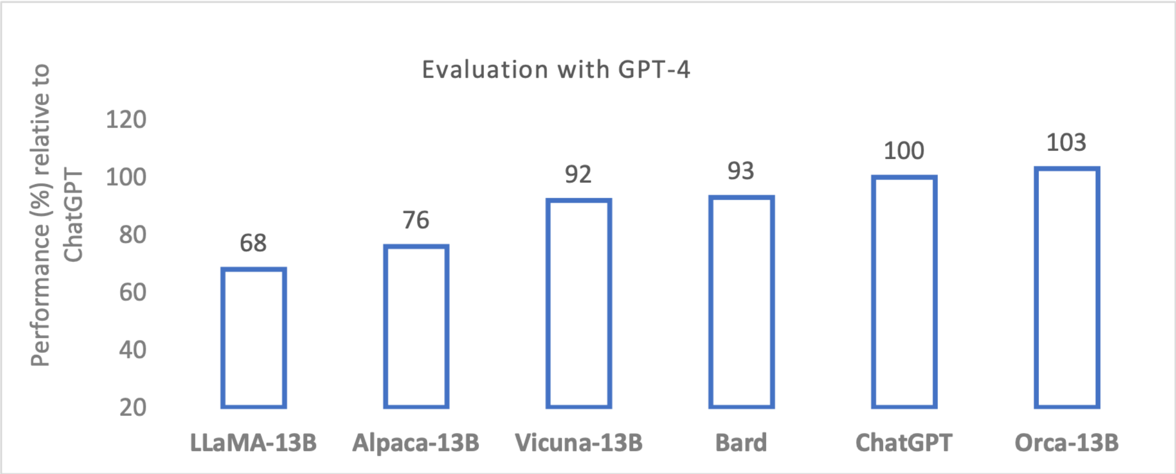


Figure 1: Orca (13B params) outperforms a wide range of foundation models including OpenAI ChatGPT as evaluated by GPT-4 in the Vicuna evaluation set. We further demonstrate similar results against a wide range of evaluation sets from other works in experiments.

Orca: Progressive Learning from Complex Explanation Traces of GPT-4, Mukherjee et. al., Microsoft Research, 2023

Meta

The Llama 3 Herd of Models

Llama Team, AI @ Meta[1]

[1]A detailed contributor list can be found in the appendix of this paper.

Modern artificial intelligence (AI) systems are powered by foundation models. This paper presents a new set of foundation models, called Llama 3. It is a herd of language models that natively support multilinguality, coding, reasoning, and tool usage. Our largest model is a dense Transformer with 405B parameters and a context window of up to 128K tokens. This paper presents an extensive empirical evaluation of Llama 3. We find that Llama 3 delivers comparable quality to leading language models such as GPT-4 on a plethora of tasks. We publicly release Llama 3, including pre-trained and post-trained versions of the 405B parameter language model and our Llama Guard 3 model for input and output safety. The paper also presents the results of experiments in which we integrate image, video, and speech capabilities into Llama 3 via a compositional approach. We observe this approach performs competitively with the state-of-the-art on image, video, and speech recognition tasks. The resulting models are not yet being broadly released as they are still under development.

Date: July 23, 2024
Website: https://llama.meta.com/

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024

Meta

The Llama 3 Herd of Models

Llama Team, AI @ Meta[1]
[1]A detailed contributor list can be found in the appendix of this paper.

Modern artificial intelligence (AI) systems are powered by foundation models. This paper presents a new set of foundation models, called Llama 3. It is a herd of language models that natively support multilinguality, coding, reasoning, and tool usage. Our largest model is a dense Transformer with 405B parameters and a context window of up to 128K tokens. This paper presents an extensive empirical evaluation of Llama 3. We find that Llama 3 delivers comparable quality to leading language models such as GPT-4 on a plethora of tasks. We publicly release Llama 3, including pre-trained and post-trained versions of the 405B parameter language model and our Llama Guard 3 model for input and output safety. The paper also presents the results of experiments in which we integrate image, video, and speech capabilities into Llama 3 via a compositional approach. We observe this approach performs competitively with the state-of-the-art on image, video, and speech recognition tasks. The resulting models are

Date: July 23, 2024
Website: https://llama.

4.3 Capabilities

We highlight special efforts to improve performance for specific capabilities such as code (Section 4.3.1), multilinguality (Section 4.3.2), math and reasoning (Section 4.3.3), long context (Section 4.3.4), tool use (Section 4.3.5), factuality (Section 4.3.6), and steerability (Section 4.3.7).

Meta

Date: July 23, 2024
Website: https://llama.meta.com/

4.3 Capabilities

4.3.3 Math and Reasoning

We define reasoning as the ability to perform multi-step computations and arrive at the correct final answer. Several challenges guide our approach to training models that excel in mathematical reasoning:

- **Lack of prompts**: As the complexity of questions increases, the number of valid prompts or questions for Supervised Fine-Tuning (SFT) decreases. This scarcity makes it difficult to create diverse and representative training datasets for teaching models various mathematical skills (Yu et al., 2023; Yue et al., 2023; Luo et al., 2023; Mitra et al., 2024; Shao et al., 2024; Yue et al., 2024b).
- **Lack of ground truth chain of thought**: Effective reasoning requires a step-by-step solution to facilitate the reasoning process (Wei et al., 2022c). However, there is often a shortage of ground truth chains of thought, which are essential for guiding the model how to break down the problem step-by-step and reach the final answer (Zelikman et al., 2022).
- **Incorrect intermediate steps**: When using model-generated chains of thought, the intermediate steps may not always be correct (Cobbe et al., 2021; Uesato et al., 2022; Lightman et al., 2023; Wang et al., 2023a). This inaccuracy can lead to incorrect final answers and needs to be addressed.
- **Teaching models to use external tools**: Enhancing models to utilize external tools, such as code interpreters, allows them to reason by interleaving code and text (Gao et al., 2023; Chen et al., 2022; Gou et al., 2023). This capability can significantly improve their problem-solving abilities.
- **Discrepancy between training and inference**: There is often a discrepancy between how the model is finetuned during training and how it is used during inference. During inference, the finetuned model may interact with humans or other models, requiring it to improve its reasoning using feedback. Ensuring consistency between training and real-world usage is crucial for maintaining reasoning performance.

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024

SFT: Learning Objectives

- Tool use
  - Understanding tokens such as <tool>
  - Being able to understand tools when tools are given in the context
  - Conform to protocol of tool call (e.g., Model Context Protocol, MCP)
- Intuition: Usefulness within agentic frameworks

```
{ ”tool": "filesystem.writeFile",
  ”args": {
    "path": ”src/parity.py",
    "content": "def parity(x):\nreturn x % 2 == 0"
  }
}
```

Meta

The Llama 3 Herd of Models

Llama Team, AI @ Meta[1]

[1] A detailed contributor list can be found in the appendix of this paper.

Modern artificial intelligence (AI) systems are powered by foundation models. This paper presents a new set of foundation models, called Llama 3. It is a herd of language models that natively support multilinguality, coding, reasoning, and tool usage. Our largest model is a dense Transformer with 405B parameters and a context window of up to 128K tokens. This paper presents an extensive empirical evaluation of Llama 3. We find that Llama 3 delivers comparable quality to leading language models such as GPT-4 on a plethora of tasks. We publicly release Llama 3, including pre-trained and post-trained versions of the 405B parameter language model and our Llama Guard 3 model for input and output safety. The paper also presents the results of experiments in which we integrate image, video, and speech capabilities into Llama 3 via a compositional approach. We observe this approach performs competitively with the state-of-the-art on image, video, and speech recognition tasks. The resulting models are not yet being broadly released as they are still under development.

Date: July 23, 2024
Website: https://llama.meta.com/

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024

Meta

The Llama 3 Herd of Models

Llama Team, AI @ Meta[1]
[1]A detailed contributor list can

Modern artificial intelligence (A
new set of foundation models, c
multilinguality, coding, reasoni
405B parameters and a context
empirical evaluation of Llama 3.
models such as GPT-4 on a plet
post-trained versions of the 405B
and output safety. The paper a
video, and speech capabilities in
performs competitively with the
resulting models are not yet bei

Date: July 23, 2024
Website: https://llama.meta.com/

4.3.5 Tool Use

Teaching LLMs to use tools such as search engines or code interpreters hugely expands the range of tasks they can solve, transforming them from pure chat models into more general assistants (Nakano et al., 2021; Thoppilan et al., 2022; Parisi et al., 2022; Gao et al., 2023; Mialon et al., 2023a; Schick et al., 2024). We train Llama 3 to interact with the following tools:

- **Search engine.** Llama 3 is trained to use Brave Search[7] to answer questions about recent events that go beyond its knowledge cutoff or that require retrieving a particular piece of information from the web.
- **Python interpreter.** Llama 3 can generate and execute code to perform complex computations, read files uploaded by the user and solve tasks based on them such as question answering, summarization, data analysis or visualization.
- **Mathematical computational engine.** Llama 3 can use the Wolfram Alpha API[8] to more accurately solve math, science problems, or retrieve accurate information from Wolfram's database.

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024

SFT: Learning Objectives

- Code generation
  - Being better at generating high-quality code
  - Multi-lingual: generating code in different programming languages
  - Explanation: writing code with comments and documentations

```python
def gushti_cdi():
    n = int(input())
    arr = list(map(int, input().split()))
    points = 0
    for i in range(n):
        if arr[i] == 0 and i != 0:
            idx = arr.index(max(arr[:i]))
            points += arr[idx]
            arr[idx] = 0
    return points

for _ in range(int(input())):
    print(gushti_cdi())
```

```php
<?php
function gushti_cdi() {
    $n = (int)fgets(STDIN);
    $arr = explode(" ", fgets(STDIN));
    $points = 0;
    for ($i = 0; $i < $n; $i++) {
        if ($arr[$i] == 0 && $i != 0) {
            $maxVal = max(array_slice($arr, 0, $i));
            $idx = array_search($maxVal, $arr);
            $points += $arr[$idx];
            $arr[$idx] = 0;
        }
    }
    return $points;
}

$t = (int)fgets(STDIN);
for ($i = 0; $i < $t; $i++) {
    echo gushti_cdi() . "\n";
}
?>
```

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024

Meta

The Llama 3 Herd of Models

Llama Team, AI @ Meta[1]

[1]A detailed contributor list can be found in the appendix of this paper.

Modern artificial intelligence (AI) systems are powered by foundation models. This paper presents a new set of foundation models, called Llama 3. It is a herd of language models that natively support multilinguality, coding, reasoning, and tool usage. Our largest model is a dense Transformer with 405B parameters and a context window of up to 128K tokens. This paper presents an extensive empirical evaluation of Llama 3. We find that Llama 3 delivers comparable quality to leading language models such as GPT-4 on a plethora of tasks. We publicly release Llama 3, including pre-trained and post-trained versions of the 405B parameter language model and our Llama Guard 3 model for input and output safety. The paper also presents the results of experiments in which we integrate image, video, and speech capabilities into Llama 3 via a compositional approach. We observe this approach performs competitively with the state-of-the-art on image, video, and speech recognition tasks. The resulting models are not yet being broadly released as they are still under development.

Date: July 23, 2024
Website: https://llama.meta.com/

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024

Meta

The Llama 3 Herd of Models

Llama Team, AI @ Meta[1]
[1]A detailed contributor list can be found in the appendix of this paper.

Modern artificial intelligence (AI) systems are powered by foundation models. This paper presents a new set of foundation models, called Llama 3. It is a herd of language models that natively support multilinguality, coding, reasoning, and tool usage. Our largest model is a dense Transformer with 405B parameters and a context window of up to 128K tokens. This paper presents an extensive empirical evaluation of Llama 3. We find that Llama 3 delivers comparable quality to leading language models such as GPT-4 on a plethora of tasks. We publicly release Llama 3, including pre-trained and post-trained versions of the 405B parameter language model and our Llama Guard 3 model for input and output safety. The
video, and speech capab
performs competitively
resulting models are not

Date: July 23, 2024
Website: https://llama.me

4.3.1 Code

LLMs for code have received significant attention since the release of Copilot and Codex (Chen et al., 2021). Developers are now widely using these models to generate code snippets, debug, automate tasks, and improve code quality. For Llama 3, we target improving and evaluating code generation, documentation, debugging, and review capabilities for the following high priority programming languages: Python, Java, Javascript, C/C++, Typescript, Rust, PHP, HTML/CSS, SQL, bash/shell. Here, we present our work on improving these coding capabilities via training a code expert, generating synthetic data for SFT, improving formatting with system prompt steering, and creating quality filters to remove bad samples from our training data.

Meta

The l

Llama Tea

[1]A detail

Modern a
new set o
multiling
405B par
empirical
models su
post-trai
and output safety. The paper also p
video, and speech capabilities into L
performs competitively with the sta
resulting models are not yet being b

Date: July 23, 2024
Website: https://llama.meta.com/

4.3.1 Code

LLMs for code have received significant attention since the release of Copilot and Codex (Chen et al., 2021). Developers are now widely using these models to generate code snippets, debug, automate tasks, and improve code quality. For L
and review capabil
C/C++, Typescrip
these coding capabi
with system promp

Expert training. We train a **code expert** which we use to collect high quality human annotations for code throughout subsequent rounds of post-training. This is accomplished by branching the main pre-training run and continuing pre-training on a 1T token mix of mostly (>85%) code data. Continued pre-training on domain-specific data has been shown to be effective for improving performance in a specific domain (Gururangan et al., 2020). We follow a recipe similar to that of CodeLlama (Rozière et al., 2023). For the last several thousand steps of training we perform long-context finetuning (LCFT) to extend the expert's context length to 16K tokens on a high quality mix of repo-level code data. Finally, we follow the similar post-training modeling recipes described in Section 4.1 to align this model, except with SFT and DPO data mixes primarily targeting code. This model is also used for rejection sampling (Section 4.2.2) for coding prompts.

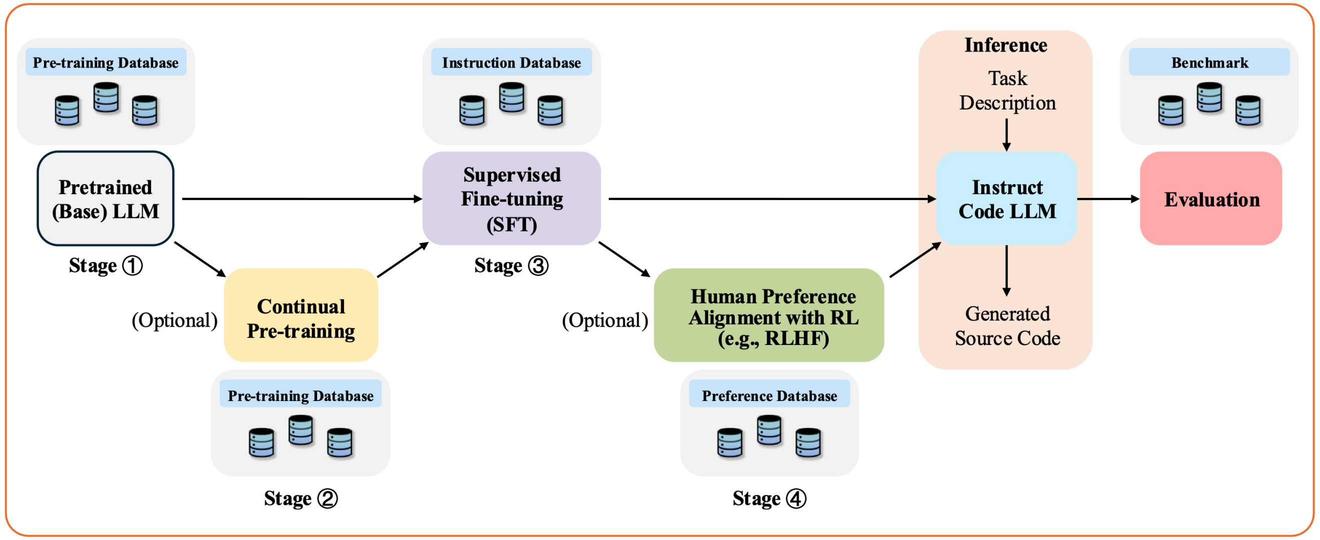
Synthetic data generation. During development, we identified key issues in code generation, including difficulty in following instructions, code syntax errors, incorrect code generation, and difficulty in fixing bugs. While intensive human annotation could theoretically resolve these issues, synthetic data generation offers a complementary approach at a lower cost and higher scale, unconstrained by the expertise level of annotators. As such, we use Llama 3 and the code expert to generate a large quantity of synthetic SFT dialogs.

We describe three high-level approaches for generating synthetic code data. In total, we generate over 2.7M synthetic examples which were used during SFT.

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024

Today's Agenda

- Supervised Fine-tuning
  - ~~Learning objective~~
  - Dataset
- Reinforcement Learning
  - Learning objective
  - Optimization
  - Dataset



SFT: Dataset & Data-mix

- Intuition:
  - Pre-training gives LLM general understanding of language
  - SFT during Post-training adds capabilities related to expected usage contexts: dialogue, coding, agentic tool use, reasoning, etc.
  - Note: multiple capabilities → general-purpose language model
- Challenges:
  - How do we obtain data for these purposes?
  - How do we mix them into SFT dataset?

SFT: Dataset & Data-mix

- Instruction following
- Reasoning
- Tool use / agentic behavior
- Coding
- …

SFT: Dataset & Data-mix

- Instruction following
- Reasoning
- Tool use / agentic behavior
- Coding
- …

SUPER-NATURALINSTRUCTIONS:
Generalization via Declarative Instructions on 1600+ NLP Tasks

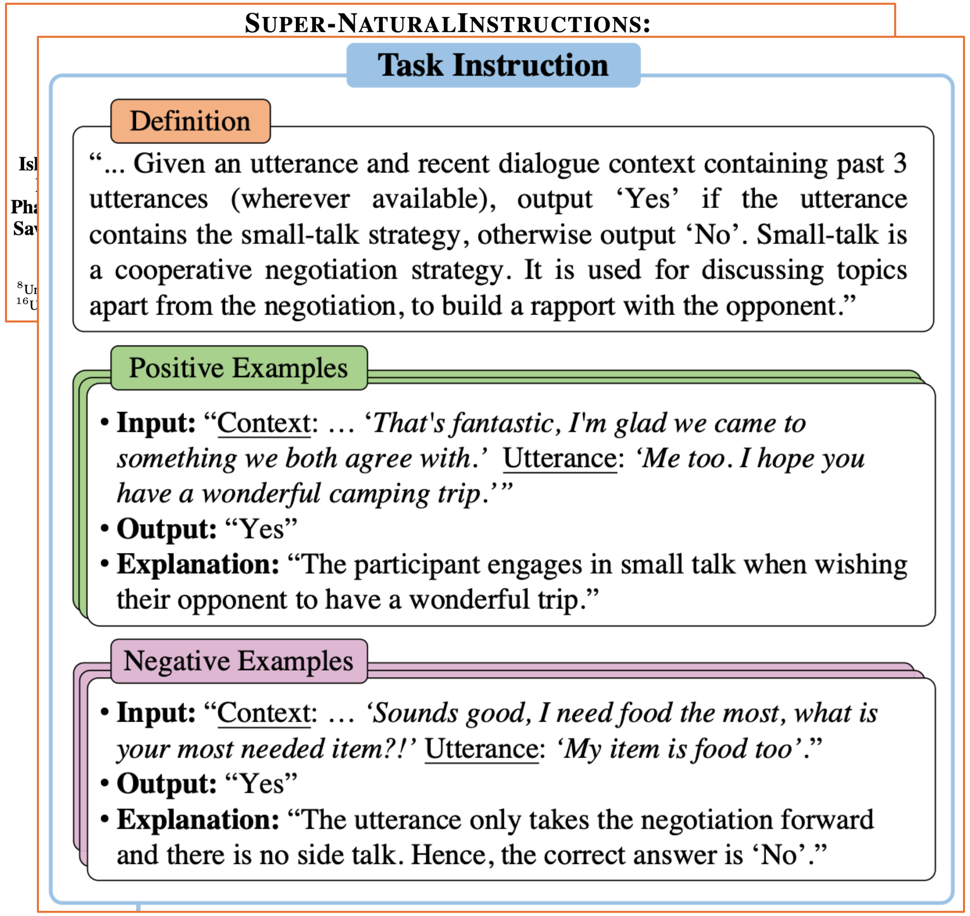
◇Yizhong Wang[2] ◇Swaroop Mishra[3] ♣Pegah Alipoormolabashi[4] ♣Yeganeh Kordi[5] Amirreza Mirzaei[4] Anjana Arunkumar[3] Arjun Ashok[6] Arut Selvan Dhanasekaran[3] Atharva Naik[7] David Stap[8] Eshaan Pathak[9] Giannis Karamanolakis[10] Haizhi Gary Lai[11] Ishan Purohit[12] Ishani Mondal[13] Jacob Anderson[3] Kirby Kuznia[3] Krima Doshi[3] Maitreya Patel[3] Kuntal Kumar Pal[3] Mehrad Moradshahi[14] Mihir Parmar[3] Mirali Purohit[15] Neeraj Varshney[3] Phani Rohitha Kaza[3] Pulkit Verma[3] Ravsehaj Singh Puri[3] Rushang Karia[3] Shailaja Keyur Sampat[3] Savan Doshi[3] Siddhartha Mishra[16] Sujan Reddy[17] Sumanta Patro[18] Tanay Dixit[19] Xudong Shen[20] Chitta Baral[3] Yejin Choi[1,2] Noah A. Smith[1,2] Hannaneh Hajishirzi[1,2] Daniel Khashabi[21]

[1]Allen Institute for AI [2]Univ. of Washington [3]Arizona State Univ. [4]Sharif Univ. of Tech. [5]Tehran Polytechnic [6]PSG College of Tech. [7]IIT Kharagpur [8]Univ. of Amsterdam [9]UC Berkeley [10]Columbia Univ. [11]Factored AI [12]Govt. Polytechnic Rajkot [13]Microsoft Research [14]Stanford Univ. [15]Zycus Infotech [16]Univ. of Massachusetts Amherst [17]National Inst. of Tech. Karnataka [18]TCS Research [19]IIT Madras [20]National Univ. of Singapore [21]Johns Hopkins Univ.

Super-NaturalInstruction: Generalization via Declarative Instructions on 1600+ NLP Tasks, Wang et. al., 2022

SFT: Dataset & Data-mix

- Instruction following
- Reasoning
- Tool use / agentic behavior
- Coding
- …

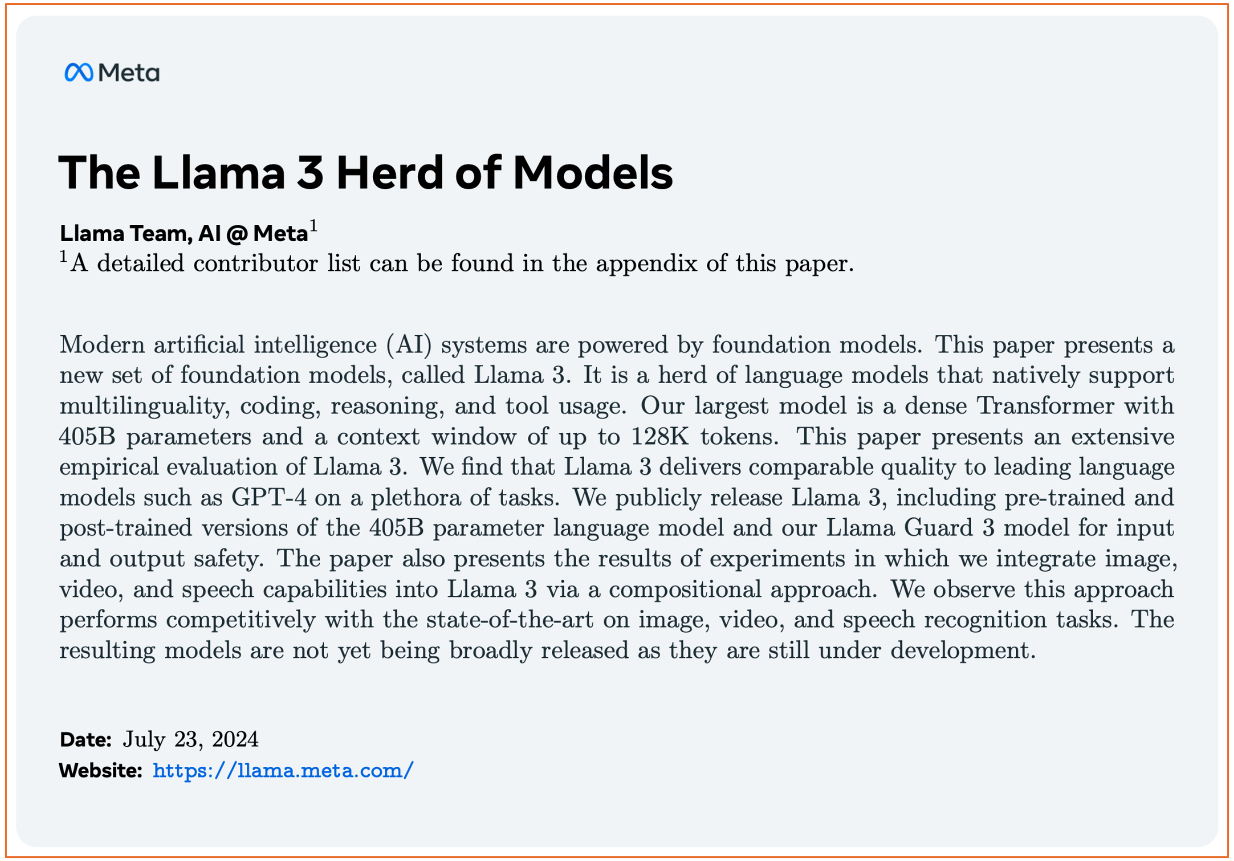


Expert written

Super-NaturalInstruction: Generalization via Declarative Instructions on 1600+ NLP Tasks, Wang et. al., 2022

SFT: Dataset & Data-mix

- Instruction following
- Reasoning
- Tool use / agentic behavior
- Coding
- …



Meta

The Llama 3 Herd of Models

Llama Team, AI @ Meta[1]

[1]A detailed contributor list can be found in the appendix of this paper.

Modern artificial intelligence (AI) systems are powered by foundation models. This paper presents a new set of foundation models, called Llama 3. It is a herd of language models that natively support multilinguality, coding, reasoning, and tool usage. Our largest model is a dense Transformer with 405B parameters and a context window of up to 128K tokens. This paper presents an extensive empirical evaluation of Llama 3. We find that Llama 3 delivers comparable quality to leading language models such as GPT-4 on a plethora of tasks. We publicly release Llama 3, including pre-trained and post-trained versions of the 405B parameter language model and our Llama Guard 3 model for input and output safety. The paper also presents the results of experiments in which we integrate image, video, and speech capabilities into Llama 3 via a compositional approach. We observe this approach performs competitively with the state-of-the-art on image, video, and speech recognition tasks. The resulting models are not yet being broadly released as they are still under development.

Date: July 23, 2024
Website: https://llama.meta.com/

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024

SFT: Dataset & Data-mix

- Instruction following
- Reasoning
- Tool use / agentic behavior
- Coding
- …

To address these challenges, we apply the following methodologies:

- **Addressing the lack of prompts:** We source relevant pre-training data from mathematical contexts and converted it into a question-answer format which can then be used for supervised finetuning. Additionally, we identify mathematical skills where the model under-performs and actively sourced prompts from humans to teach models such skills. To facilitate this process, we create a taxonomy of mathematical skills (Didolkar et al., 2024) and ask humans to provide relevant prompts/questions accordingly.
- **Augmenting training data with step-wise reasoning traces**: We use Llama 3 to generate step-by-step solutions for a set of prompts. For each prompt, the model produces a variable number of generations. These generations are then filtered based on the correct answer (Li et al., 2024a). We also do self-verification where Llama 3 is used to verify whether a particular step-by-step solution is valid for a given question. This process improves the quality of the finetuning data by eliminating instances where the model does not produce valid reasoning traces.
- **Filtering incorrect reasoning traces**: We train outcome and stepwise reward models (Lightman et al., 2023; Wang et al., 2023a) to filter training data where the intermediate reasoning steps were incorrect. These reward models are used to eliminate data with invalid step-by-step reasoning, ensuring high-quality data for finetuning. For more challenging prompts, we use Monte Carlo Tree Search (MCTS) with learned step-wise reward models to generate valid reasoning traces, further enhancing the collection of high-quality reasoning data (Xie et al., 2024).
- **Interleaving code and text reasoning**: We prompt Llama 3 to solve reasoning problems through a combination of textual reasoning and associated Python code (Gou et al., 2023). Code execution is used as a feedback signal to eliminate cases where the reasoning chain was not valid, ensuring the correctness of the reasoning process.

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024

SFT: Dataset & Data-mix

- Instruction following
- Reasoning
- Tool use / agentic behavior
- Coding
- …

To address these challenges, we apply the following methodologies:

- **Addressing the lack of prompts:** We source relevant pre-training data from mathematical contexts and converted it into a question-answer format which can then be used for supervised finetuning. Additionally, we identify mathematical skills where the model under-performs and actively sourced prompts from humans to teach models such skills. To facilitate this process, we create a taxonomy of mathematical skills (Didolkar et al., 2024) and ask humans to provide relevant prompts/questions accordingly.
- **Augmenting training data with step-wise reasoning traces**: We use Llama 3 to generate step-by-step solutions for a set of prompts. For each prompt, the model produces a variable number of generations. These generations are then filtered based on the correct answer (Li et al., 2024a). We also do self-verification where Llama 3 is used to verify whether a particular step-by-step solution is valid for a given question. This process improves the quality of the finetuning data by eliminating instances where the model does not produce valid reasoning traces.
- **Filtering incorrect reasoning traces**: We train outcome and stepwise reward models (Lightman et al., 2023; Wang et al., 2023a) to filter training data where the intermediate reasoning steps were incorrect. These reward models are used to eliminate data with invalid step-by-step reasoning, ensuring high-quality data for finetuning. For more challenging prompts, we use Monte Carlo Tree Search (MCTS) with learned step-wise reward models to generate valid reasoning traces, further enhancing the collection of high-quality reasoning data (Xie et al., 2024).
- **Interleaving code and text reasoning**: We prompt Llama 3 to solve reasoning problems through a combination of textual reasoning and associated Python code (Gou et al., 2023). Code execution is used as a feedback signal to eliminate cases where the reasoning chain was not valid, ensuring the correctness of the reasoning process.

Self-supervision: generating reasoning chains by other language models, where results are post-hoc verified and filtered.

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024

SFT: Dataset & Data-mix

- Instruction following
- Reasoning
- Tool use / agentic behavior
- Coding
- …

Tool datasets. To create data for tool usage applications, we leverage the following procedure:

- **Single-step tool use:** We start by few-shot generation of synthetic user prompts which, by construction, require a call to one of our core tools (for example, questions that exceed our knowledge cutoff date). Then, still relying on few-shot generation, we generate appropriate tool calls for these prompts, execute them, and add the output to the model's context. Finally, we prompt the model again to generate a final answer to the user's query based on the tool output. We end up with trajectories of the following form: system prompt, user prompt, tool call, tool output, final answer. We also filter around 30% this dataset to remove tool calls that cannot be executed or other formatting issues.
- **Multi-step tool use:** We follow a similar protocol and first generate synthetic data to teach the model basic multi-step tool use capabilities. To do this, we first prompt Llama 3 to generate user prompts that require at least two tool calls, that can be the same or different tools from our core set. Then, conditioned on these prompts, we few-shot prompt Llama 3 to generate a solution consisting of interleaved reasoning steps and tool calls, similar to ReAct (Yao et al., 2022). See Figure 10 for an example of Llama 3 performing a task involving multi-step tool usage.
- **File uploads:** We annotate for the following filetypes: .TXT, .DOCX, .PDF, .PPTX, .XLSX, .CSV, .TSV, .PY, .JSON, .JSONL, .HTML, .XML. Our prompts are based on a provided file, and ask to summarize the contents of the file, find and fix bugs, optimize a piece of code, perform data analysis or visualization. See Figure 11 for an example of Llama 3 performing a task involving a file upload.

Synthetic dataset:
- Problem: given the instruction, generate a tool call
- "Inverse problem": given a tool specification, generate sample instructions that require the given tool call

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024

SFT: Dataset & Data-mix

- Instruction following
- Reasoning
- Tool use / agentic behavior
- Coding
- ...

1. **Synthetic data generation: execution feedback.** The 8B and 70B models show significant performance improvements when trained on data generated by a larger, more competent model. However, our initial experiments revealed that training Llama 3 405B on its own generated data is not helpful (and can even degrade performance). To address this limitation, we introduced execution feedback as a source of truth, enabling the model to learn from its mistakes and stay on track. In particular, we generate large dataset of approximately one million synthetic coding dialogues using the following process:

2. **Synthetic data generation: programming language translation.** We observe a performance gap between major programming languages (*e.g.*, Python/C++) and less common ones (*e.g.*, Typescript/PHP). This is not surprising as we have less training data for less common programming languages. To mitigate this, we supplement our existing data by *translating* data from common programming languages to less common languages (similar to Chen et al. (2023) in the context of reasoning). This is achieved by prompting Llama 3 and ensuring quality via syntax parsing, compilation, and execution. Figure 8 demonstrates an example of synthetic PHP code translated from Python. This improves performance significantly for less common languages as measured by the MultiPL-E (Cassano et al., 2023) benchmark.

3. **Synthetic data generation: backtranslation.** To improve certain coding capabilities (e.g., documentation, explanations) where execution feedback is less informative for determining quality, we employ an alternative multi-step approach. Using this procedure, we generated approximately 1.2M synthetic dialogs related to code explanation, generation, documentation, and debugging. Beginning with code snippets from a variety of languages in our pre-training data:

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024

SFT: Dataset & Data-mix

- Intuition:
  - Pre-training gives LLM general understanding of language
  - SFT during Post-training adds capabilities related to expected usage contexts: dialogue, coding, agentic tool use, reasoning, etc.
- Challenges:
  - ~~How do we obtain data for these purposes?~~
  - How do we mix them into SFT dataset?

Dataset	% of examples	Avg. # turns	Avg. # tokens	Avg. # tokens in context	Avg. # tokens in final response
General English	52.66%	6.3	974.0	656.7	317.1
Code	14.89%	2.7	753.3	378.8	374.5
Multilingual	3.01%	2.7	520.5	230.8	289.7
Exam-like	8.14%	2.3	297.8	124.4	173.4
Reasoning and tools	21.19%	3.1	661.6	359.8	301.9
Long context	0.11%	6.7	38,135.6	37,395.2	740.5
Total	100%	4.7	846.1	535.7	310.4

Table 7 Statistics of SFT data. We list internally collected SFT data used for Llama 3 alignment. Each SFT example consists of a context (i.e., all conversation turns except the last one) and a final response.

Keeping the capability of a general language model, avoiding forgetting

Keeping multilingual capability present

Optimize for modern features such as reasoning and tool use

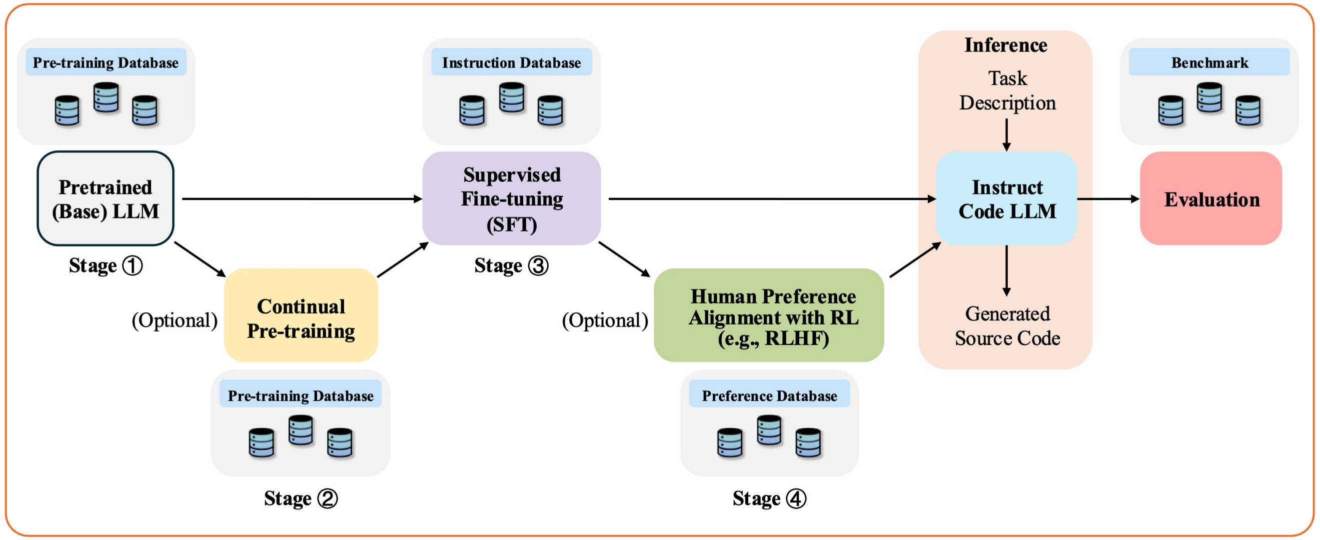
Dataset	% of examples	Avg. # turns	Avg. # tokens	Avg. # tokens in context	Avg. # tokens in final response
General English	52.66%	6.3	974.0	656.7	317.1
Code	14.89%	2.7	753.3	378.8	374.5
Multilingual	3.01%	2.7	520.5	230.8	289.7
Exam-like	8.14%	2.3	297.8	124.4	173.4
Reasoning and tools	21.19%	3.1	661.6	359.8	301.9
Long context	0.11%	6.7	38,135.6	37,395.2	740.5
Total	100%	4.7	846.1	535.7	310.4

Table 7 Statistics of SFT data. We list internally collected SFT data used for Llama 3 alignment. Each SFT example consists of a context (i.e., all conversation turns except the last one) and a final response.

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024

Today's Agenda

- Supervised Fine-tuning
  - ~~Learning objective~~
  - ~~Dataset~~
- Reinforcement Learning
  - Learning objective
  - Optimization
  - Dataset



RL: Learning Objectives

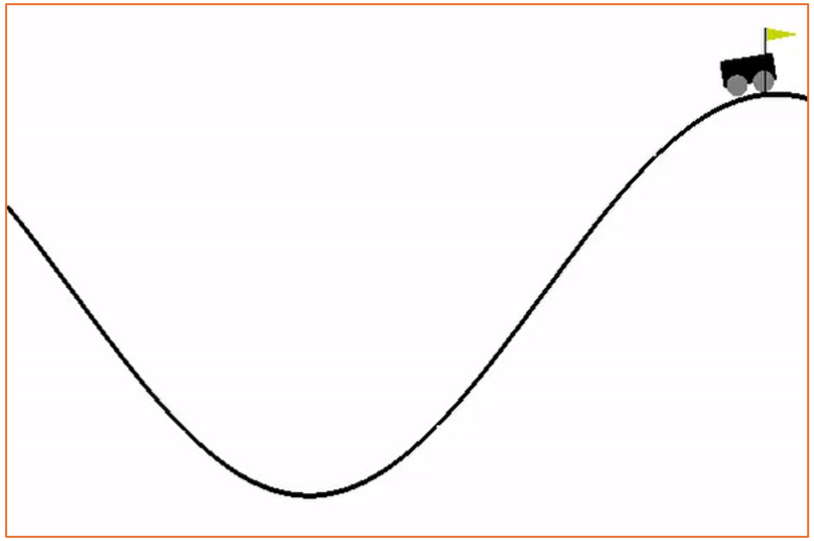
RL: Learning Objectives

- Quick review of reinforcement learning
- Core concepts: learn a policy that can maximize return
    - Environment ($P(s_{t+1} \mid s_t, a_t)$)
    - State / observation (s_t)
    - Action (a_t)
    - Reward (r_t; return R)
    - Policy (π)

RL: Review

- Environment ($P(s_{t+1} \mid s_t, a_t)$)
- State / observation (s_t)
- Action (a_t)
- Reward (r_t; return R)
- Policy (π)

Mountain Car



State: $((p_x, p_y), (v_x, v_y))$

Action: $a \in \{-1, 0, +1\}$

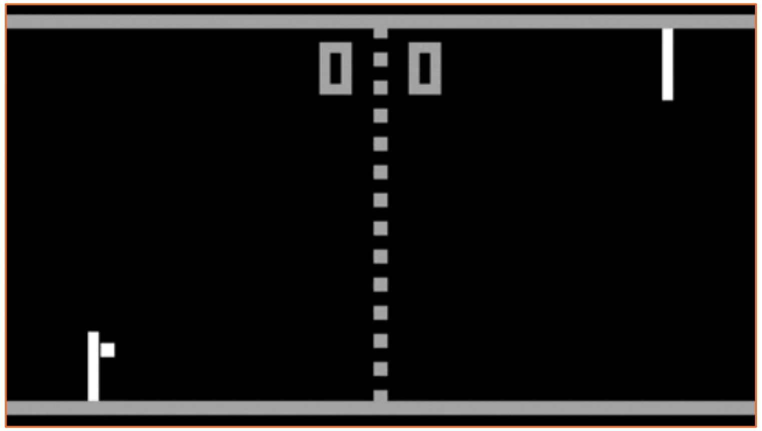
Reward: -1 per time step until goal; 0 at the goal

Policy: π, throttle until speed=0, then reverse direction

RL: Review

- Environment ($P(s_{t+1} \mid s_t, a_t)$)
- State / observation (s_t)
- Action (a_t)
- Reward (r_t; return R)
- Policy (π)

Pong



State: Image $(r, g, b)^{84\times 84}$

Action: $a \in \{\mathrm{up}, \mathrm{down}, \mathrm{no\text{-}op}\}$

Reward: -1 per lost, $+1$ per win

Policy: π, detect position of the ball; always stay synchronized

RL: Learning Objective

- Environment ($P(s_{t+1} \mid s_t, a_t)$)
- State / observation (s_t)
- Action (a_t)
- Reward (r_t; return R)
- Policy (π_θ; θ is parameters)

Language Modeling

Q: "Explain why the sky is blue."

A: "The sky appears blue because of Rayleigh scattering — the way sunlight interacts with Earth's atmosphere. Here's the process step by step: Sunlight is made of many colors White sunlight actually contains all colors of light..."

State: input token sequence $\mathbf{x}$ = [explain, why, the, ...] and the current output sequence y = [The, sky, appears, ...]

Action: $a \in \Sigma$, a single token

Reward: $r_\phi(\mathbf{x}, \mathbf{y})$ a reward model for helpfulness/correctness/...

Policy: π_θ, the language model parametrized by θ

RL: Learning Objective

Environment ($P(s_{t+1} \mid s_t, a_t)$)
State / observation (s_t)
Action (a_t)
Reward (r_t; return R)
Policy (π)

- Intuition:
  - In open settings, you only know whether the answer is "good/bad" after you have generated the entire sentence or have interacted with the user with multiple turns
  - "goodness" of an answer can be evaluated with many metrics: helpfulness, alignment with human values, reasoning clarity, correctness of the answer, syntax and semantic of a program

Training language models to follow instructions with human feedback

Long Ouyang* Jeff Wu* Xu Jiang* Diogo Almeida* Carroll L. Wainwright*

Pamela Mishkin* Chong Zhang Sandhini Agarwal Katarina Slama Alex Ray

John Schulman Jacob Hilton Fraser Kelton Luke Miller Maddie Simens

Amanda Askell† Peter Welinder Paul Christiano*†

Jan Leike* Ryan Lowe*

OpenAI

Training language models to follow instructions with human feedback, Ouyang et. al., OpenAI, 2022

Training language models to follow instructions with human feedback

Long Ouyang* Jeff Wu* X

Pamela Mishkin* Chong Zha

John Schulman Jacob Hilto

Amanda Askell†

Jan Leike*

3.5 Models

We start with the GPT-3 pretrained language models from Brown et al. (2020). These models are trained on a broad distribution of Internet data and are adaptable to a wide range of downstream tasks, but have poorly characterized behavior. Starting from these models, we then train models with three different techniques:

Supervised fine-tuning (SFT). We fine-tune GPT-3 on our labeler demonstrations using supervised learning. We trained for 16 epochs, using a cosine learning rate decay, and residual dropout of 0.2. We do our final SFT model selection based on the RM score on the validation set. Similarly to Wu et al. (2021), we find that our SFT models overfit on validation loss after 1 epoch; however, we find that training for more epochs helps both the RM score and human preference ratings, despite this overfitting.

Reward modeling (RM). Starting from the SFT model with the final unembedding layer removed, we trained a model to take in a prompt and response, and output a scalar reward. In this paper we only use 6B RMs, as this saves a lot of compute, and we found that 175B RM training could be unstable and thus was less suitable to be used as the value function during RL (see Appendix C for more details).

Training language models to follow instructions with human feedback, Ouyang et. al., OpenAI, 2022

Training language models to follow instructions

Long

Pame

John

3.5 Models

We start with the GPT-3 pretrained language models from Brown et al. (2020). These models are trained on a broad distribution of Internet data and are adaptable to a wide range of downstream tasks, but have poorly chara
different techniques:

Supervised fine-tuni
learning. We trained
We do our final SFT
et al. (2021), we find
that training for more
overfitting.

Reward modeling (R
we trained a model to
only use 6B RMs, as
unstable and thus was
more details).

In order to speed up comparison collection, we present labelers with anywhere between $K = 4$ and $K = 9$ responses to rank. This produces $\binom{K}{2}$ comparisons for each prompt shown to a labeler. Since comparisons are very correlated within each labeling task, we found that if we simply shuffle the comparisons into one dataset, a single pass over the dataset caused the reward model to overfit.[5] Instead, we train on all $\binom{K}{2}$ comparisons from each prompt as a single batch element. This is much more computationally efficient because it only requires a single forward pass of the RM for each completion (rather than $\binom{K}{2}$ forward passes for K completions) and, because it no longer overfits, it achieves much improved validation accuracy and log loss.

Specifically, the loss function for the reward model is:

$$\operatorname{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta\left(x,y_w\right) - r_\theta\left(x,y_l\right)\right)\right)\right] \qquad (1)$$

where $r_\theta(x, y)$ is the scalar output of the reward model for prompt x and completion y with parameters θ, y_w is the preferred completion out of the pair of y_w and y_l, and D is the dataset of human comparisons.

Training language models to follow instructions with human feedback, Ouyang et. al., OpenAI, 2022

RL: Reward Modeling

Environment ($P(s_{t+1} \mid s_t, a_t)$)
State / observation (s_t)
Action (a_t)
Reward (r_t; return R)
Policy (π)

- Before:
  - Rewards for LLMs are generated by the environment
  - Usually hardcoded (e.g., mountain car, cartpole, pong)
- Now:
  - Rewards for LLMs are provided by reward models (RM)
  - → Reward models need to be separately trained with human preference
  - → Human preference needs to come from separate datasets

RL: Reward Modeling

- Training a reward model

GPT-3 → InstructGPT → GPT-3.5/ChatGPT

In order to speed up comparison collection, we present labelers with anywhere between $K = 4$ and $K = 9$ responses to rank. This produces $\binom{K}{2}$ comparisons for each prompt shown to a labeler. Since comparisons are very correlated within each labeling task, we found that if we simply shuffle the comparisons into one dataset, a single pass over the dataset caused the reward model to overfit.[5] Instead, we train on all $\binom{K}{2}$ comparisons from each prompt as a single batch element. This is much more computationally efficient because it only requires a single forward pass of the RM for each completion (rather than $\binom{K}{2}$ forward passes for K completions) and, because it no longer overfits, it achieves much improved validation accuracy and log loss.

Specifically, the loss function for the reward model is:

$$\operatorname{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta\left(x, y_w\right) - r_\theta\left(x, y_l\right)\right)\right)\right] \tag{1}$$

where $r_\theta(x, y)$ is the scalar output of the reward model for prompt x and completion y with parameters θ, y_w is the preferred completion out of the pair of y_w and y_l, and D is the dataset of human comparisons.

Training language models to follow instructions with human feedback, Ouyang et. al., OpenAI, 2022

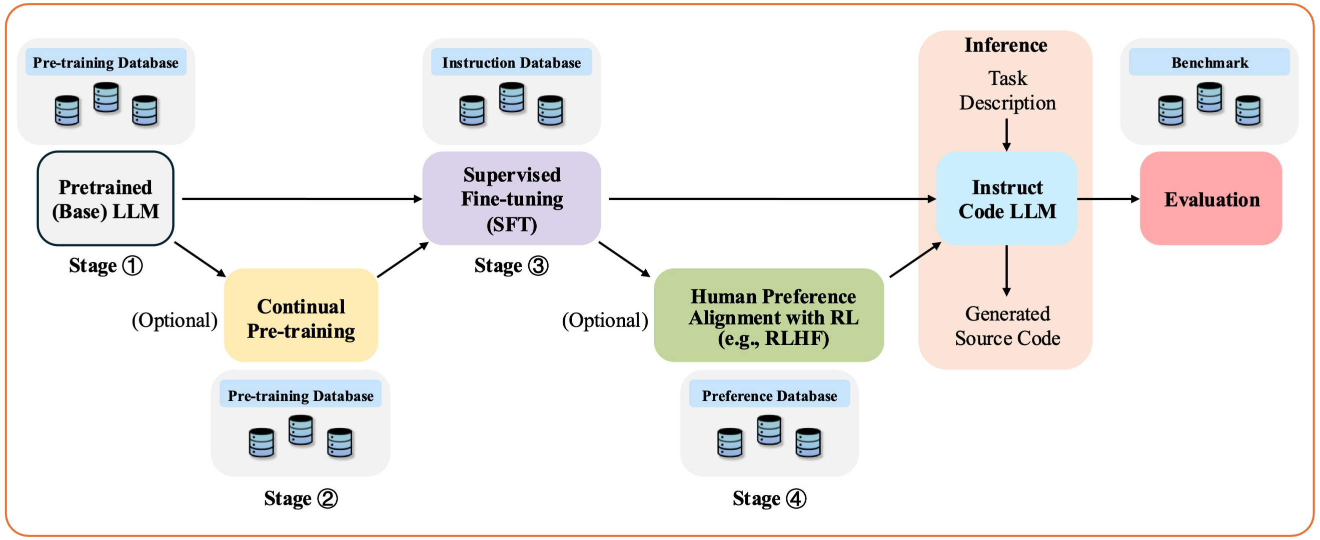
RL: Learning Objective

Environment ($P(s_{t+1} \mid s_t, a_t)$)
State / observation (s_t)
Action (a_t)
Reward (r_t; return R)
Policy (π)

- Summary
  - Use Reinforcement Learning (RL) to fine-tune LLMs to maximize return
  - → returns are meant to represent human preference
  - → mechanically, returns/rewards are generated by reward models
  - → reward models are trained from human preference datasets

Today's Agenda

- Supervised Fine-tuning
  - ~~Learning objective~~
  - ~~Dataset~~
- Reinforcement Learning
  - ~~Learning objective~~
  - Optimization
  - Dataset



RL: Optimization

- DQN (Deep Q-Network)
  - Directly model Q-value: the value function of an action $Q_\theta(s, a)$
- Policy Gradient (REINFORCE)

  $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) R]$
  - Directly optimize the policy $\pi_\theta(a|s)$
- Actor-Critic Model (AC)

  $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) A(s, a)]$
  - Adding a critic $V_\phi(s)$ and “advantage estimate” $A(s, a) = R - V_\phi(s)$
- Proximal Policy Optimization (PPO)

  $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(y|x)\, (r_\phi(x, y) - \beta(\log \pi_\theta(y|x) - \log \pi_{\mathrm{SFT}}(y|x) + 1))]$
  - To stabilize, we clip the loss based on how far the policy deviates
- Group Relative Policy Optimization (GRPO)
  - (Specifically for LLM) grouped normalized advantage $A_i = (r_i - \bar{r})/\sigma_r$

RL: Optimization

- (DQN)
  - Directly model Q-value: the value function of an action $Q_\theta(s, a)$
- (REINFORCE)

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) R]$$

  - Directly optimize the policy $\pi_\theta(a|s)$
- (AC)

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) A(s, a)]$$

  - Adding a critic $V_\phi(s)$ and "advantage estimate" $A(s, a) = R - V_\phi(s)$
- (PPO)

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(y|x)\ (r_\phi(x, y) - \beta(\log \pi_\theta(y|x) - \log \pi_{\mathrm{SFT}}(y|x) + 1))]$$

  - To stabilize, we clip the loss based on how far the policy deviates
- (GRPO)
  - (Specifically for LLM) grouped normalized advantage $\tilde{A}_i = (r_i - \bar{r})/\sigma_r$

$$\boxed{\nabla_\phi J(\phi) = \mathbb{E}_{x,\, y_{1:k} \sim \pi_\phi}\left[\boxed{\frac{1}{k}\sum_{i=1}^{k}} \nabla_\phi \log \pi_\phi(y_i \mid x)\ \underbrace{\left(\tilde{A}_i\ -\ \beta\left(\log \frac{\pi_\phi(y_i|x)}{\pi_{\mathrm{ref}}(y_i|x)} + 1\right)\right)}_{\text{effective (shaped) advantage}}\right]}$$

Training language models to follow instructions

Long Ouyang* Jeff W

Pamela Mishkin* Cl

John Schulman Ja

Amanda Askel

Ja

Reinforcement learning (RL). Once again following Stiennon et al. (2020), we fine-tuned the SFT model on our environment using PPO (Schulman et al., 2017). The environment is a bandit environment which presents a random customer prompt and expects a response to the prompt. Given the prompt and response, it produces a reward determined by the reward model and ends the episode. In addition, we add a per-token KL penalty from the SFT model at each token to mitigate over-optimization of the reward model. The value function is initialized from the RM. We call these models "PPO."

We also experiment with mixing the pretraining gradients into the PPO gradients, in order to fix the performance regressions on public NLP datasets. We call these models "PPO-ptx." We maximize the following combined objective function in RL training:

$$\begin{aligned} \text{objective}(\phi) = & E_{(x,y)\sim D_{\pi_\phi^{\mathrm{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\pi_\phi^{\mathrm{RL}}(y \mid x)/\pi^{\mathrm{SFT}}(y \mid x)\right)\right] + \\ & \gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{\mathrm{RL}}(x))\right] \end{aligned} \tag{2}$$

where π_ϕ^{RL} is the learned RL policy, π^{SFT} is the supervised trained model, and D_{pretrain} is the pretraining distribution. The KL reward coefficient, β, and the pretraining loss coefficient, γ, control the strength of the KL penalty and pretraining gradients respectively. For "PPO" models, γ is set to 0. Unless otherwise specified, in this paper InstructGPT refers to the PPO-ptx models.

Training language models to follow instructions with human feedback, Ouyang et. al., OpenAI, 2022

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

DeepSeek-AI

research@deepseek.com

DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via

2. Approach

2.1. Overview

Previous work has heavily relied on large amounts of supervised data to enhance model performance. In this study, we demonstrate that reasoning capabilities can be significantly improved through large-scale reinforcement learning (RL), even without using supervised fine-tuning (SFT) as a cold start. Furthermore, performance can be further enhanced with the inclusion of a small amount of cold-start data. In the following sections, we present: (1) DeepSeek-R1-Zero, which applies RL directly to the base model without any SFT data, and (2) DeepSeek-R1, which applies RL starting from a checkpoint fine-tuned with thousands of long Chain-of-Thought (CoT) examples. 3) Distill the reasoning capability from DeepSeek-R1 to small dense models.

deepseek

Deep

2. Approach

2.1. Overview

Previous work
performance.
improved thro
fine-tuning (S
the inclusion o
DeepSeek-R1-
(2) DeepSeek-
long Chain-of-
small dense m

2.2.1. *Reinforcement Learning Algorithm*

Group Relative Policy Optimization In order to save the training costs of RL, we adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which foregoes the critic model that is typically the same size as the policy model, and estimates the baseline from group scores instead. Specifically, for each question q, GRPO samples a group of outputs $\{o_1, o_2, \cdots, o_G\}$ from the old policy $\pi_{\theta_{old}}$ and then optimizes the policy model π_θ by maximizing the following objective:

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

$$\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1-\varepsilon, 1+\varepsilon\right)A_i\right) - \beta\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right)\right), \tag{1}$$

$$\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right) = \frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - \log\frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - 1, \tag{2}$$

where ε and β are hyper-parameters, and A_i is the advantage, computed using a group of rewards $\{r_1, r_2, \ldots, r_G\}$ corresponding to the outputs within each group:

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \cdots, r_G\})}{\text{std}(\{r_1, r_2, \cdots, r_G\})}. \tag{3}$$

2. Approach

2.2.2. *Reward Modeling*

The reward is the source of the training signal, which decides the optimization direction of RL. To train DeepSeek-R1-Zero, we adopt a rule-based reward system that mainly consists of two types of rewards:

- **Accuracy rewards**: The accuracy reward model evaluates whether the response is correct. For example, in the case of math problems with deterministic results, the model is required to provide the final answer in a specified format (e.g., within a box), enabling reliable rule-based verification of correctness. Similarly, for LeetCode problems, a compiler can be used to generate feedback based on predefined test cases.
- **Format rewards**: In addition to the accuracy reward model, we employ a format reward model that enforces the model to put its thinking process between '<think>' and '</think>' tags.

We do not apply the outcome or process neural reward model in developing DeepSeek-R1-Zero, because we find that the neural reward model may suffer from reward hacking in the large-scale reinforcement learning process, and retraining the reward model needs additional training resources and it complicates the whole training pipeline.

2. Approach

2.2.2. *Reward Modeling*

The reward is the source of the training signal, which decides the optimization direction of RL. To train DeepSeek-R1-Zero, we adopt a rule-based reward system that mainly consists of two types of rewards:

- **Accuracy rewards**: The accuracy reward model evaluates whether the response is correct. For example, in the case of math problems with deterministic results, the model is required to provide the final answer in a specified format (e.g., within a box), enabling reliable rule-based verification of correctness. Similarly, for LeetCode problems, a compiler can be used to generate feedback based on predefined test cases.
- **Format rewards**: In addition to the accuracy reward model, we employ a format reward model that enforces the model to put its thinking process between tags.

Reward coming from compilers for coding

We do not apply the outcome or process neural reward model in developing DeepSeek-R1-Zero, because we find that the neural reward model may suffer from reward hacking in the large-scale reinforcement learning process, and retraining the reward model needs additional training resources and it complicates the whole training pipeline.

2.2.2. *Reward Modeling*

The reward is the source of the training signal, which decides the optimization direction of RL. To train DeepSeek-R1-Zero, we adopt a rule-based reward system that mainly consists of two types of rewards:

- **Accuracy rewards**: The accuracy reward model evaluates whether the response is correct. For example, in the case of math problems with deterministic results, the model is required to provide the final answer in a specified format (e.g., within a box), enabling reliable rule-based verification of correctness. Similarly, for LeetCode problems, a compiler can be used to generate feedback based on predefined test cases.
- **Format rewards**: In addition to the accuracy reward model, we employ a format reward model that enforces the model to put its thinking process betwee[n] ... tags.

No reward model for DeepSeek-R1-Zero

We do not apply the outcome or process neural reward model in developing DeepSeek-R1-Zero, because we find that the neural reward model may suffer from reward hacking in the large-scale reinforcement learning process, and retraining the reward model needs additional training resources and it complicates the whole training pipeline.

2. Approach

Model	AIME 2024		MATH-500	GPQA Diamond	LiveCode Bench	CodeForces
	pass@1	cons@64	pass@1	pass@1	pass@1	rating
OpenAI-o1-mini	63.6	80.0	90.0	60.0	53.8	1820
OpenAI-o1-0912	74.4	83.3	94.8	77.3	63.4	1843
DeepSeek-R1-Zero	71.0	86.7	95.9	73.3	50.0	1444

Table 2 | Comparison of DeepSeek-R1-Zero and OpenAI o1 models on reasoning-related benchmarks.

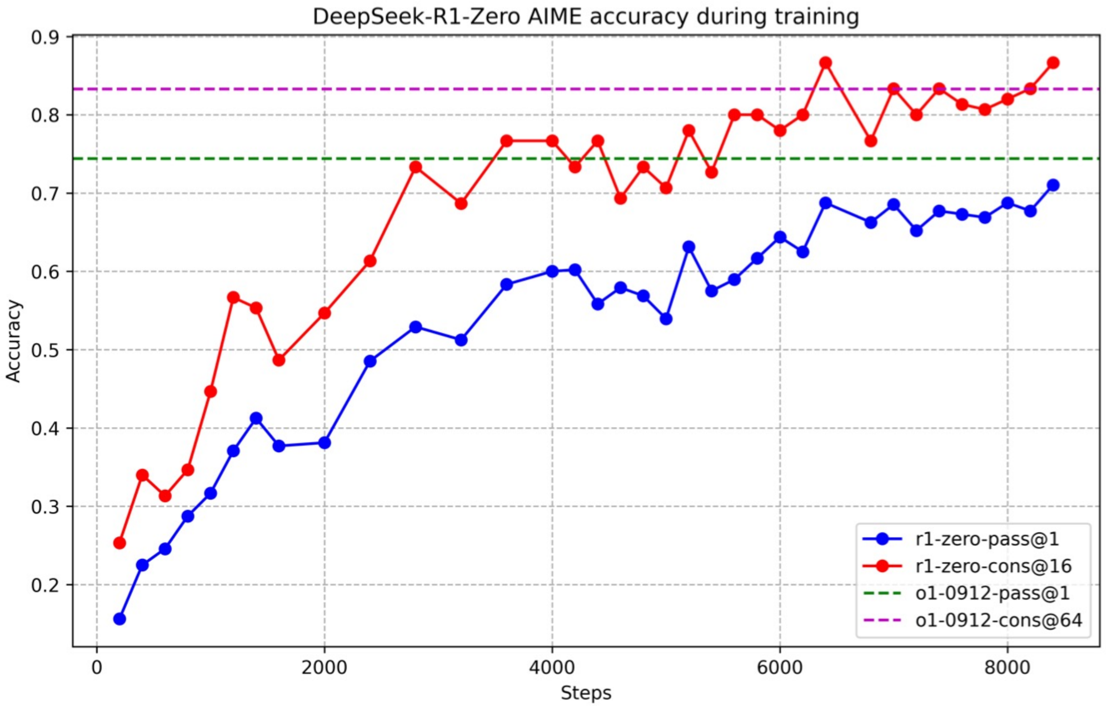


Figure 2 | AIME accuracy of DeepSeek-R1-Zero during training. For each question, we sample 16 responses and calculate the overall average accuracy to ensure a stable evaluation.

deepseek

2. Approach

2.1. Overview

Previous work has heavily relied on large amounts of supervised data to enhance model performance. In this study, we demonstrate that reasoning capabilities can be significantly improved through large-scale reinforcement learning (RL), even without using supervised fine-

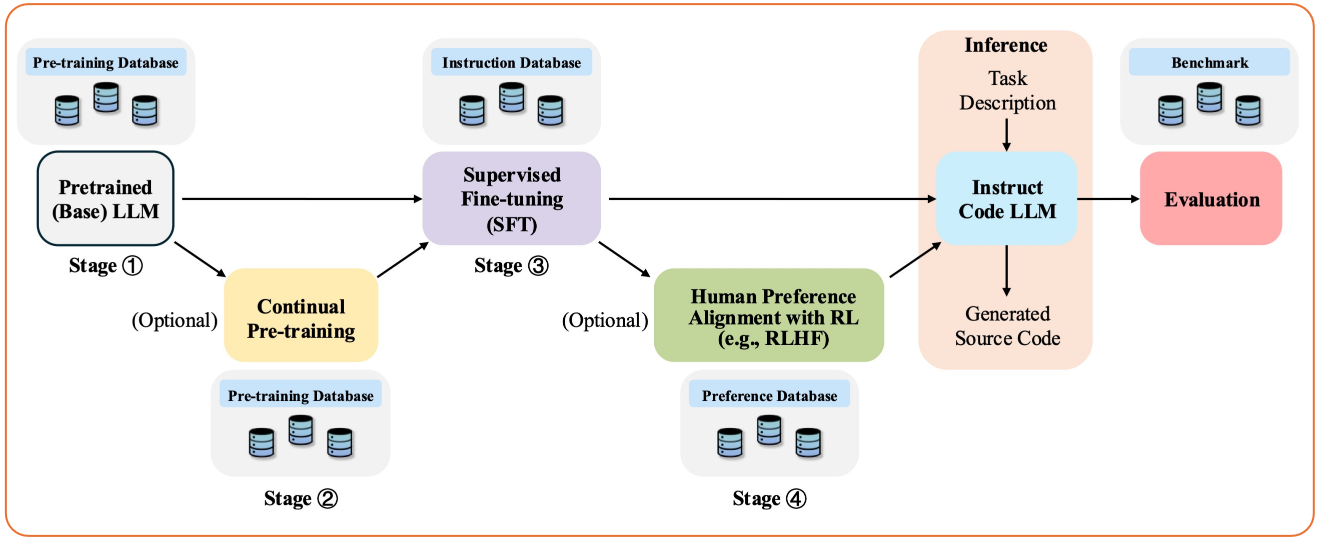
2.3. DeepSeek-R1: Reinforcement Learning with Cold Start

Inspired by the promising results of DeepSeek-R1-Zero, two natural questions arise: 1) Can reasoning performance be further improved or convergence accelerated by incorporating a small amount of high-quality data as a cold start? 2) How can we train a user-friendly model that not only produces clear and coherent Chains of Thought (CoT) but also demonstrates strong general capabilities? To address these questions, we design a pipeline to train DeepSeek-R1. The pipeline consists of four stages, outlined as follows.

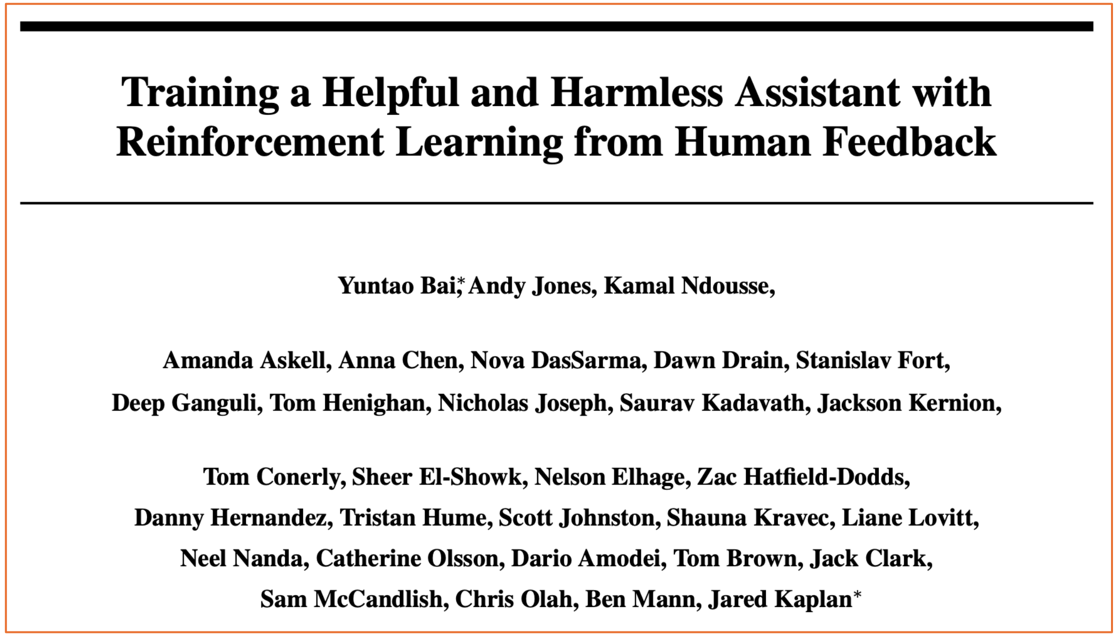
SFT (cold-start) → RL (reasoning-oriented) → SFT (rejection-sampling) → RL (all-scenario)

Today's Agenda

- Supervised Fine-tuning
  - ~~Learning objective~~
  - ~~Dataset~~
- Reinforcement Learning
  - ~~Learning objective~~
  - ~~Optimization~~
  - Dataset



RL: Human Preferences

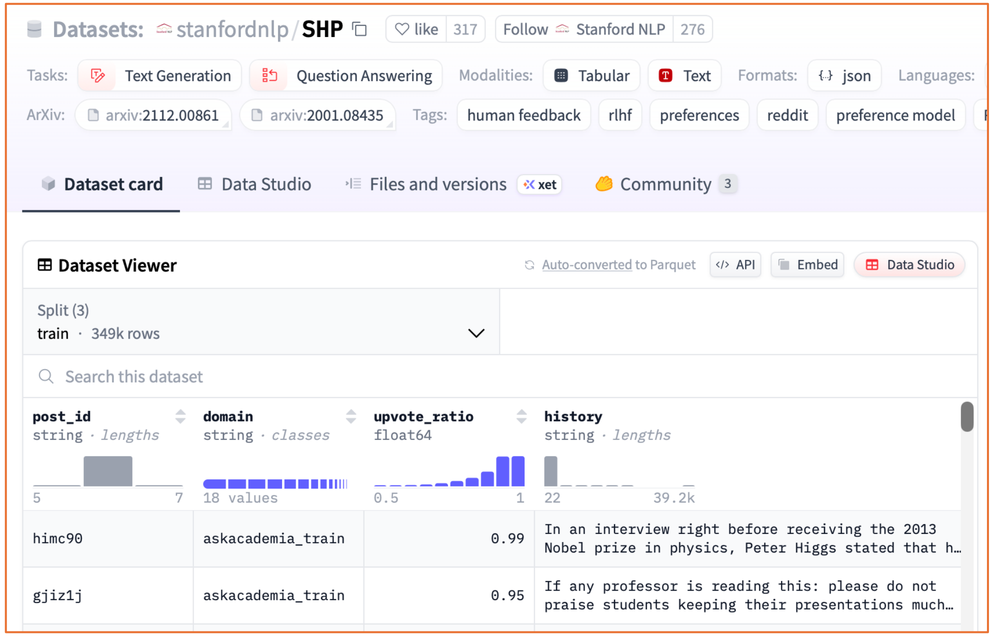

Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback

Yuntao Bai*, Andy Jones, Kamal Ndousse,

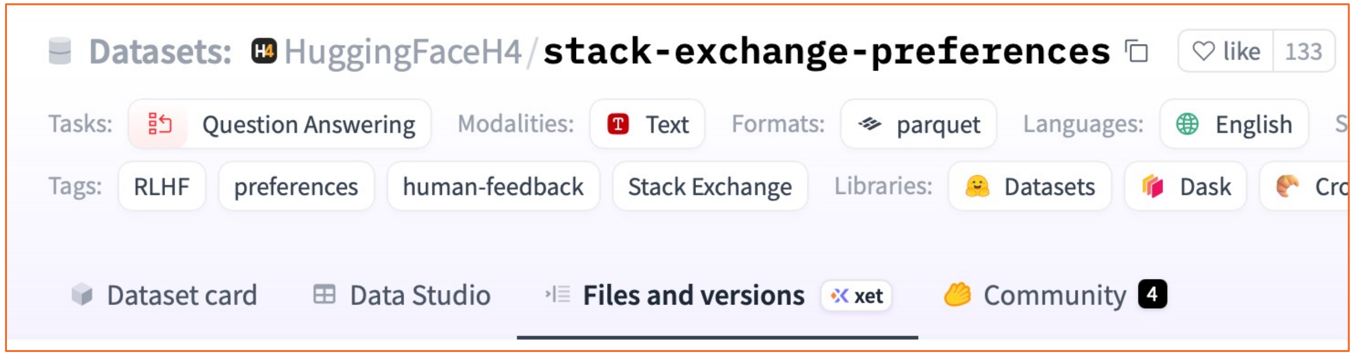
Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort,
Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion,

Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds,
Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt,
Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark,
Sam McCandlish, Chris Olah, Ben Mann, Jared Kaplan*

HH-RLHF, Anthropic, Red Teaming Dataset



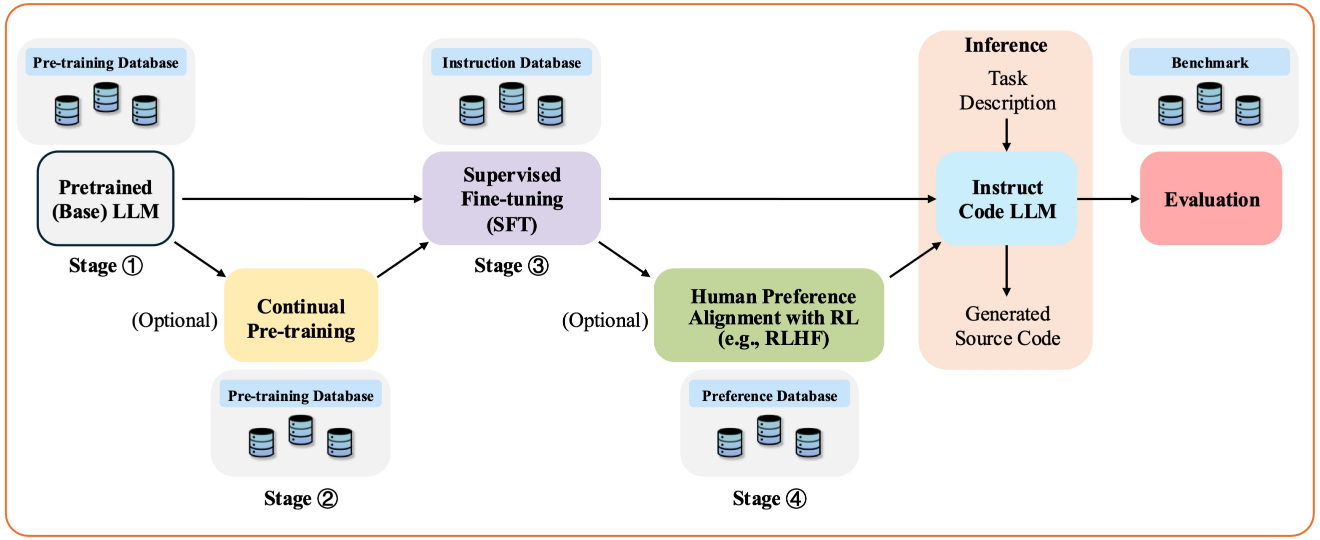
SHP, Stanford Human Preference Dataset



Stack Exchange Preferences

Today's Agenda

- Supervised Fine-tuning
  - ~~Learning objective~~
  - ~~Dataset~~
- Reinforcement Learning
  - ~~Learning objective~~
  - ~~Optimization~~
  - ~~Dataset~~



Logistics – Week 8

- Assignment 3: Coding LLM Agents
  - https://github.com/machine-programming/assignment-3
  - Fully functional web-app agent. Due: Oct 23 (Thu)
- Oral presentation sign up sheet
  - Please sign up! (16/19 received)
- Forming groups for your final projects!
  - Form a group of 2-3 before This Sunday (Oct 19)