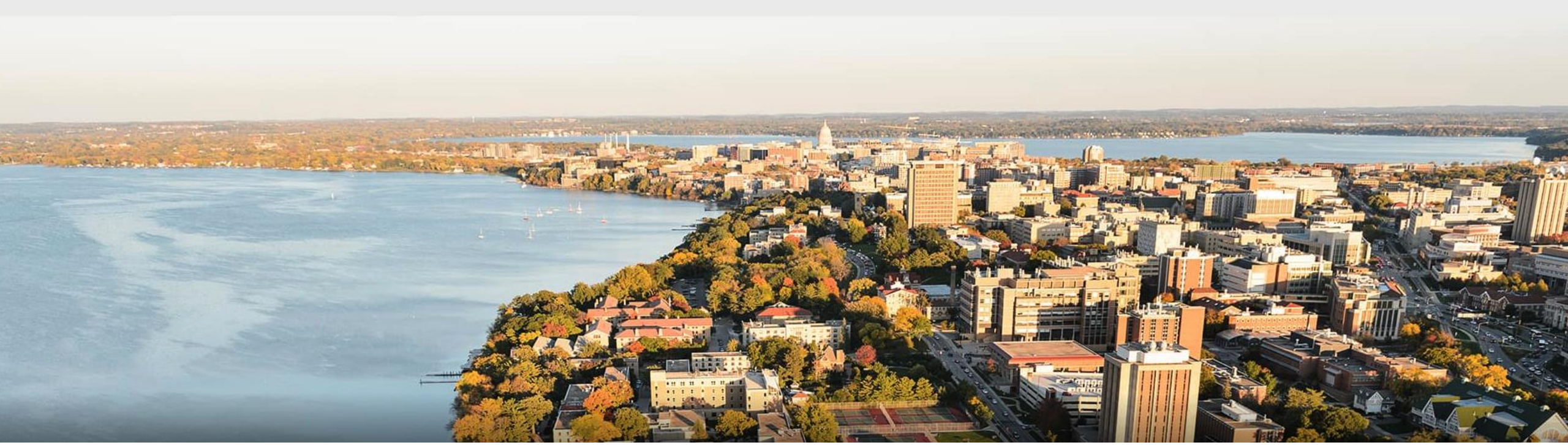


CS 839: Foundation Models
Training

Fred Sala

University of Wisconsin-Madison

Oct. 10, 2024

Announcements

- **Logistics:**
  - Homework 2 actually out!
  - Continue to form teams and sign up for presentations!
- Class roadmap:

Thursday Oct. 10	Efficient Training
Tuesday Oct. 15	Efficient Inference
Thursday Oct. 17	Hybrid Models (**Guest Lecture**)
Tuesday Oct. 22	Evaluation
Thursday Oct. 24	Multimodal Models

Outline

- **Alignment: Review and Why Does It Work?**
  - Alignment Review, Failures of supervised learning, knowledge-seeking interactions, abstains
- **Variations + Open Questions**
  - Direct Preference Optimization (DPO), RLAIF, other techniques
- **Efficient Training**
  - Scale, memory optimization (FlashAttention), parallelism, heterogenous training

Outline

- **Alignment: Review and Why Does It Work?**
  - Alignment Review, Failures of supervised learning, knowledge-seeking interactions, abstains
- **Variations + Open Questions**
  - Direct Preference Optimization (DPO), RLAIF, other techniques
- **Efficient Training**
  - Scale, memory optimization (FlashAttention), parallelism, heterogenous training

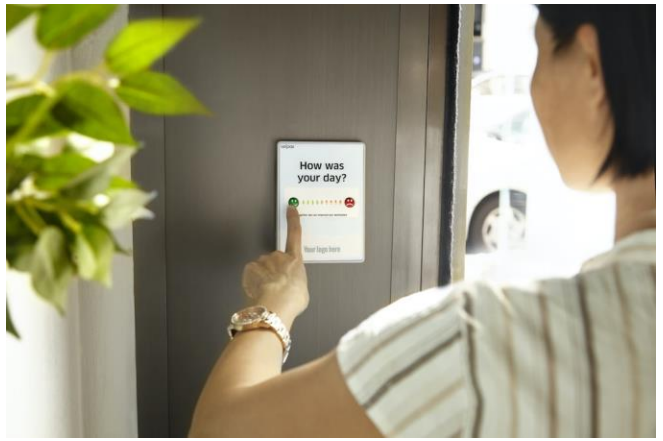
Alignment: **Basic Motivation**

Goal: produce language model outputs that users like better…

- **Hard** to specify exactly what this means,
- **Easy** to query users

Collect human feedback and use it to change the model

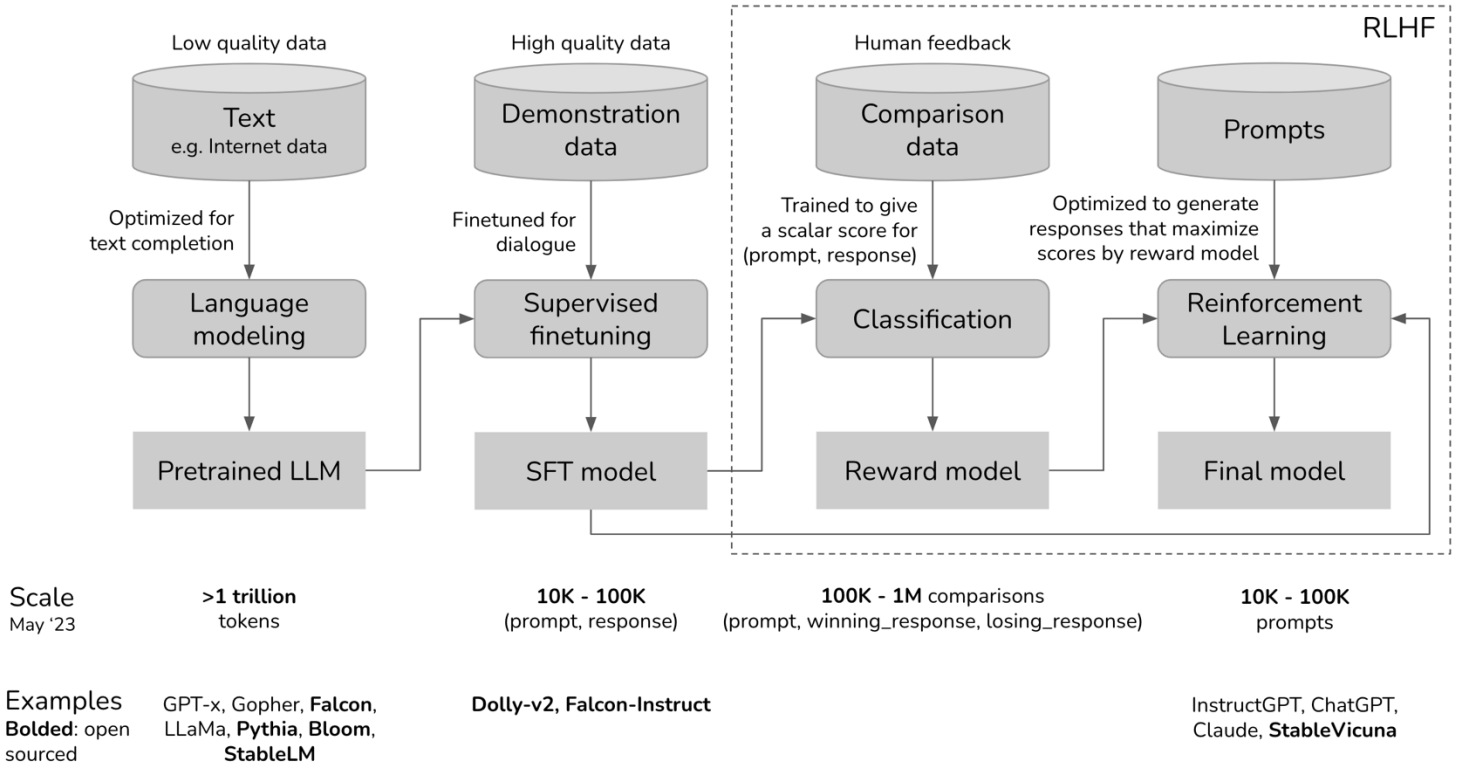
- Can do this by fine-tuning, especially with instructions
- Doesn’t quite capture what users want
- We’ll use other approaches, like RLHF



RLHF: **Setup**

Goal: produce language model outputs that users like better…

- Via RL with trained reward model (Ouyang et al ‘222)

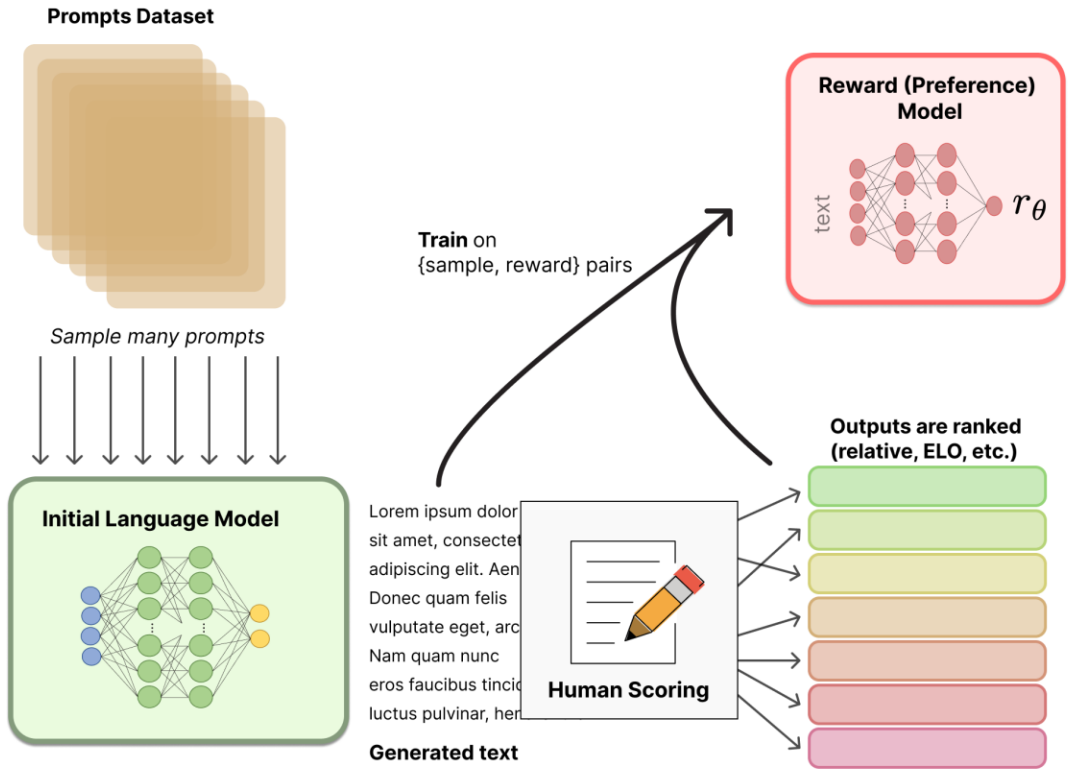


Chip Huyen

RLHF: **Reward/Preference Model**

Second stage: train reward model

- Use the human feedback to train/fine-tune another model to reproduce the metric
- **Preference model**



https://huggingface.co/blog/rlhf

RLHF: **Reward/Preference Model**

Second stage: train reward model

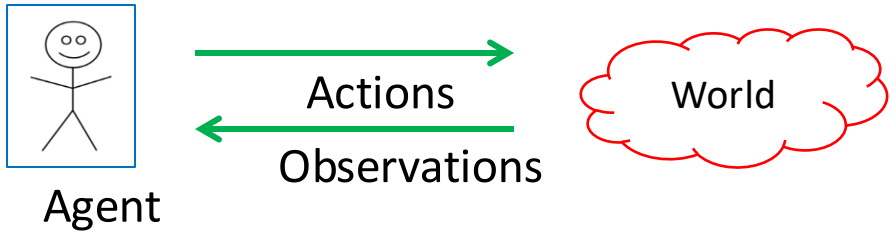
- Use the human feedback to train/fine-tune another model to reproduce the metric
- **Loss?** Based on preference models,
  - Example: Bradley-Terry model

$$p^*(y_1 \succ y_2 \mid x) = \frac{\exp\left(r^*(x, y_1)\right)}{\exp\left(r^*(x, y_1)\right) + \exp\left(r^*(x, y_2)\right)}.$$

  - Then, our reward model loss is based on the log likelihood,

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

RLHF: **Fine-Tuning with RL**



Third stage: RL

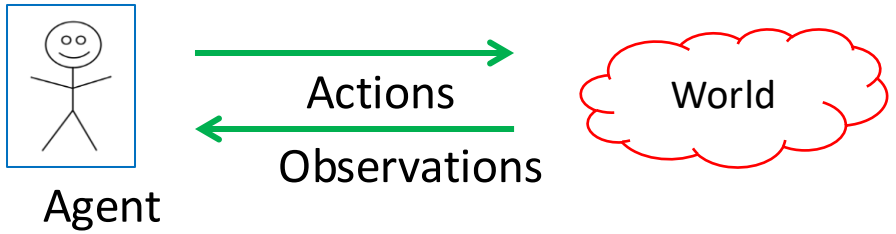
- Use an RL algorithm
- **Goal:** produce outputs that have high reward

RL formulation:

- **Action space**: all the tokens possible to output
- **State space**: all the sequences of tokens
- **Reward function**: the trained reward model
- **Policy**: the new version of the LM, taking in state and returning tokens

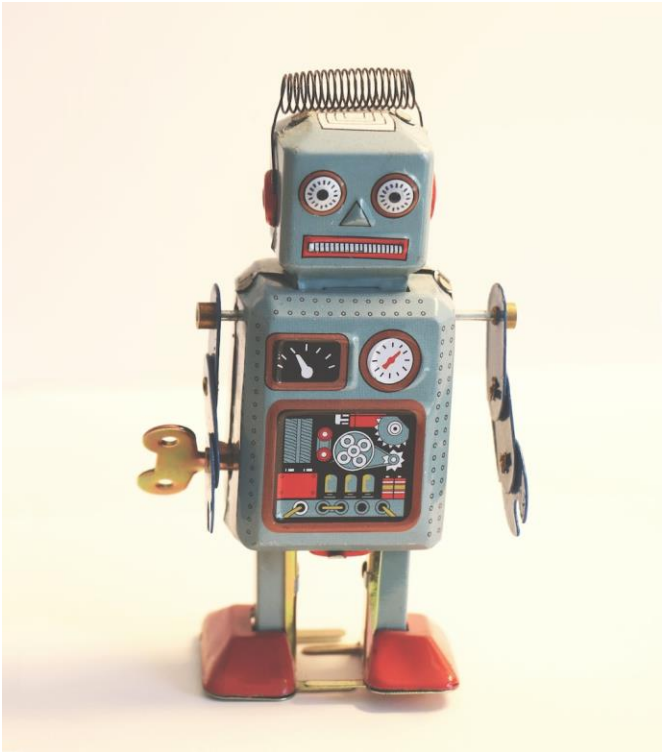
RLHF: **RL Approach**

What approach for RL stage?



- Many deep RL methods available
- Policy gradient methods
- Popular: PPO (Proximal Policy Optimization)
  - Main difference from vanilla policy gradient, you constrain change to policy at each step (Schulman et al)

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi_\theta(y|x)}\left[r_\phi(x,y)\right] - \beta\mathbb{D}_{\mathrm{KL}}\left[\pi_\theta(y \mid x) \,||\, \pi_{\mathrm{ref}}(y \mid x)\right]$$

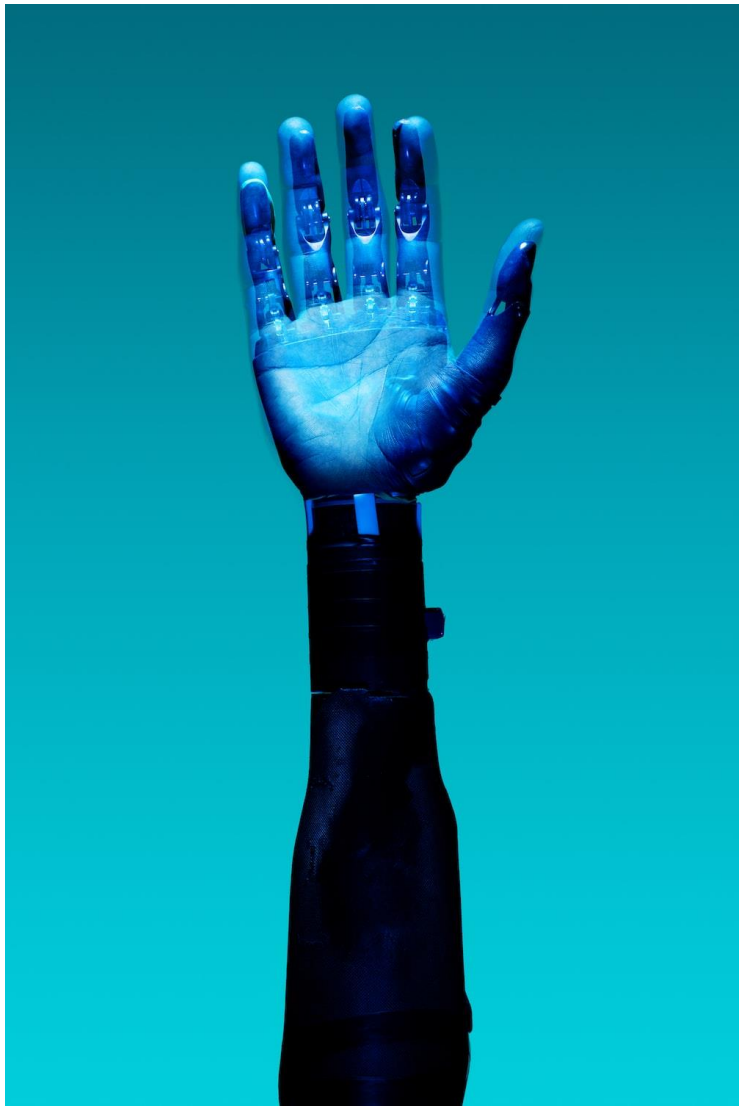


Why RLHF?

Why should we do this?

- Why does supervised fine-tuning by itself not give our goal results?

- Many hypotheses; this section inspired by Yoav Goldberg's blog:
  - https://gist.github.com/yoavg/6bff0fecd65950898eba1bb321cfbd81
  - Itself based on Schulman's talk
  - https://www.youtube.com/watch?v=hhiLw5Q_UFg



Why RLHF? **Ways To Interact**

Three “modes of interaction”:

- **text-grounded**: provide the model with text, instruction ("what are the chemical names mentioned in this text“),
- **knowledge-seeking**: provide the model with question or instruction, and expect a (truthful) answer based on the model's internal knowledge
- **creative**: provide the model with question or instruction, expect some creative output. ("Write a story about...")

Why RLHF? **Knowledge-seeking**

Three “modes of interaction”:

- **knowledge-seeking**: provide the model with question or instruction, and expect a (truthful) answer based on the model's internal knowledge

- This is hypothesized to require RL. Why does **SL fail?**
  - Case 1: know the answer: fine.
  - Case 2: don’t know the answer. Supervised learning forces memorization, cannot produce “don’t know”.
  - Worse, SL on case 2 encourages **model to lie…**

Why RLHF? **Knowledge-seeking with RL**

Three “modes of interaction”:

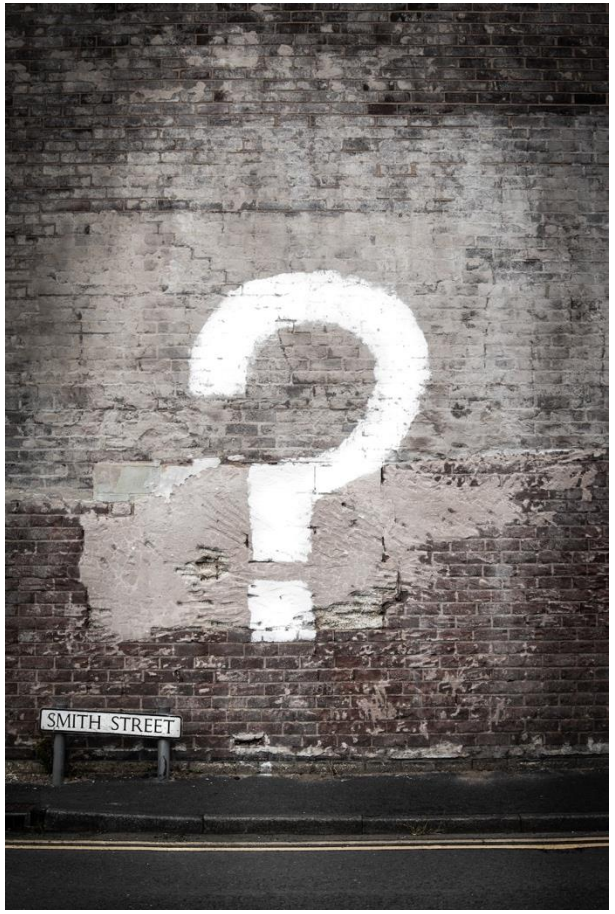
- **knowledge-seeking**: provide the model with question or instruction, and expect a (truthful) answer based on the model's internal knowledge

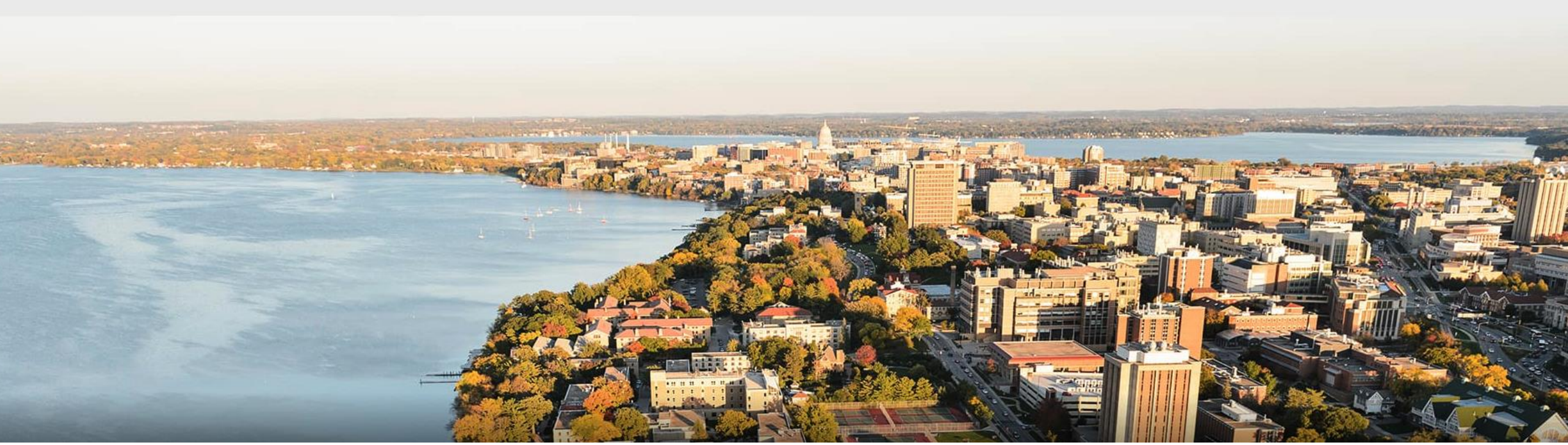
- Why does RL succeed?
  - Case 1: know the answer: fine. Get a reward
  - Case 2: don’t know the answer. Sometimes make it up and get a reward if lucky, most of the time low reward
  - **Encourages truth telling.**

Why RLHF? **Abstains**

Additionally, **we'd like our model to abstain**

- SL will really struggle with this
  - Usually no abstains in datasets
  - Even if there were, "generalization" here means abstaining on similar questions? Difficult
- RL still challenging, need to produce high reward for "don't know", but specific to model
- One way to craft a reward function:
  - High reward: correct answers
  - Medium reward: abstain
  - Negative reward: incorrect





Break & Questions

Outline

- **Alignment: Review and Why Does It Work?**
  - Alignment Review, Failures of supervised learning, knowledge-seeking interactions, abstains
- **Variations + Open Questions**
  - Direct Preference Optimization (DPO), RLAIF, other techniques
- **Efficient Training**
  - Scale, memory optimization (FlashAttention), parallelism, heterogenous training

RLHF Problems

Lots of challenges!

- **Casper et al**, “Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback”

- Challenges everywhere, all three phases:
  - **In human feedback,**
  - **In obtaining reward model,**
  - **In obtaining the policy**

RLHF Problems: **Human Feedback**

- Need to obtain some kind of “representative” collection of feedback providers
- **Simpler:**
  - Some people have biases
  - Mistakes due to lack of care (standard in crowdsourcing)
  - Adversarial data poisoners
- **Harder:**
  - In tough settings, what is “good” output?
  - Possible to manipulate humans

RLHF Problems: **Human Feedback**

- Additionally, **need high-quality data**.
- Expensive to hand-craft good prompts to drive feedback

- Feedback quality:
  - Tradeoffs in feedback levels
  - Ideally, rich
  - But harder to work with to train reward

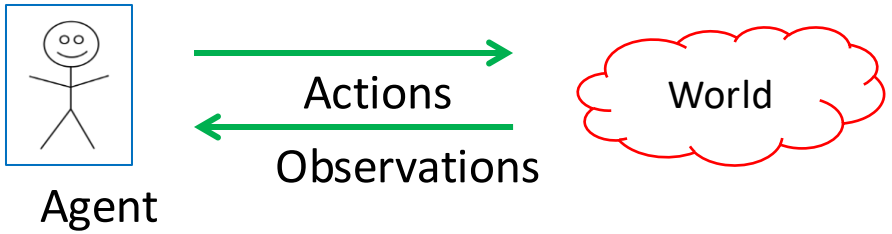
RLHF Problems: **Reward Model**

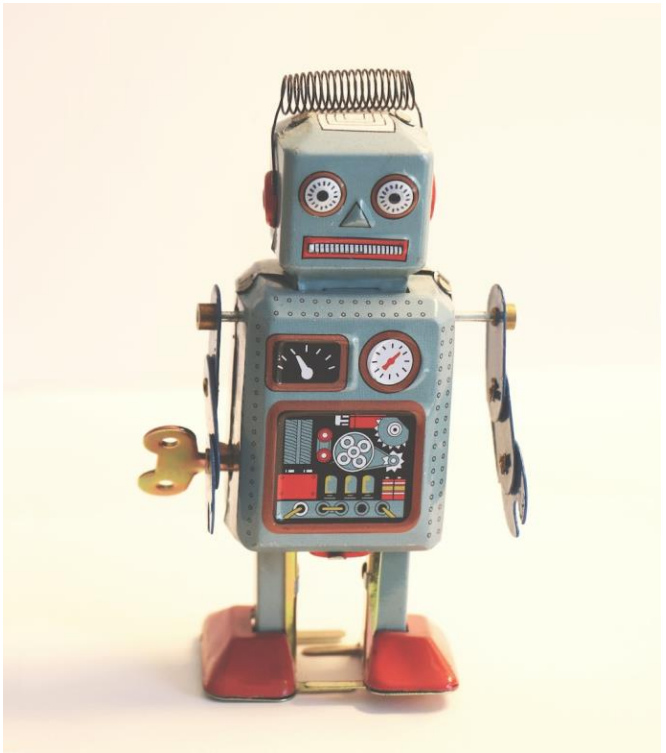
- Values can be difficult to express as a reward function
- May need to combine multiple reward functions:
  - What's a "universal" one? People are different

- Reward Hacking
  - In tough settings, what is "good" output?
  - Possible to manipulate humans

RLHF Problems: **Training**

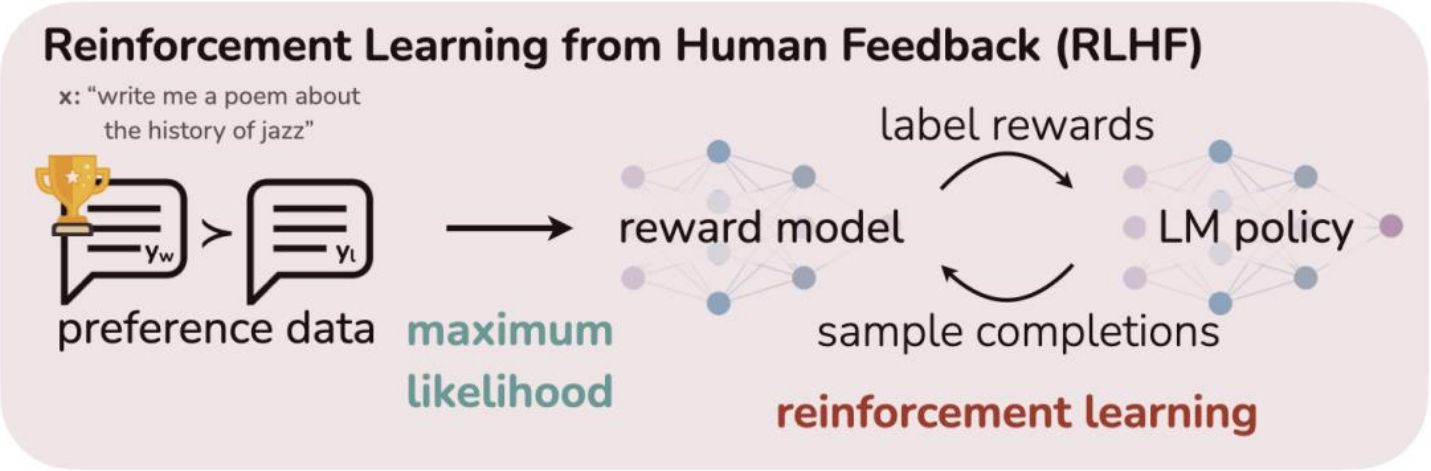
- The RL in RLHF can be difficult
- Also, learned policies **do not necessarily generalize to other environments**

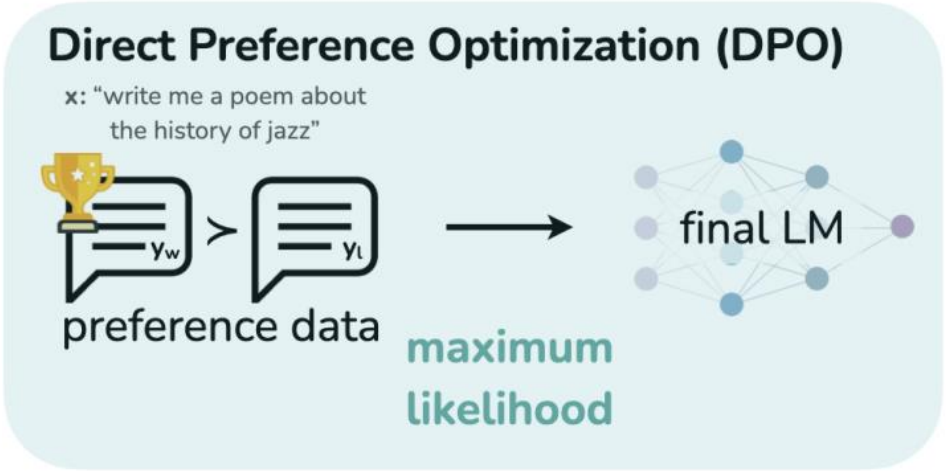




RLHF **Alternatives**

- **Direct preference optimization** (DPO)
  - Bypass separate trained reward model: just use preference information **directly** (Rafailov et al,'23)
  - **How?** Model a preference distribution from samples, integrate into a single loss (one-stage approach)





RLHF **Alternatives**

- **Direct preference optimization** (DPO)
  - Bypass separate trained reward model: just use preference information **directly** (Rafailov et al,'23)
  - **How?** Model a preference distribution from samples, integrate into a single loss (one-stage approach)

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta\log\frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right].$$

  - **Gradient step:**

$$\nabla_\theta\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) =$$

$$-\beta\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\Big[\underbrace{\sigma(\hat{r}_\theta(x,y_l) - \hat{r}_\theta(x,y_w))}_{\text{higher weight when reward estimate is wrong}}\Big[\underbrace{\nabla_\theta\log\pi(y_w \mid x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_\theta\log\pi(y_l \mid x)}_{\text{decrease likelihood of } y_l}\Big]\Big]$$

RLHF **Alternatives**

- **Many new approaches:**
  - A good survey: Ji et al ’24
- New approaches to rewards, new forms of feedback (including AI feedback), etc
- Popular research area!

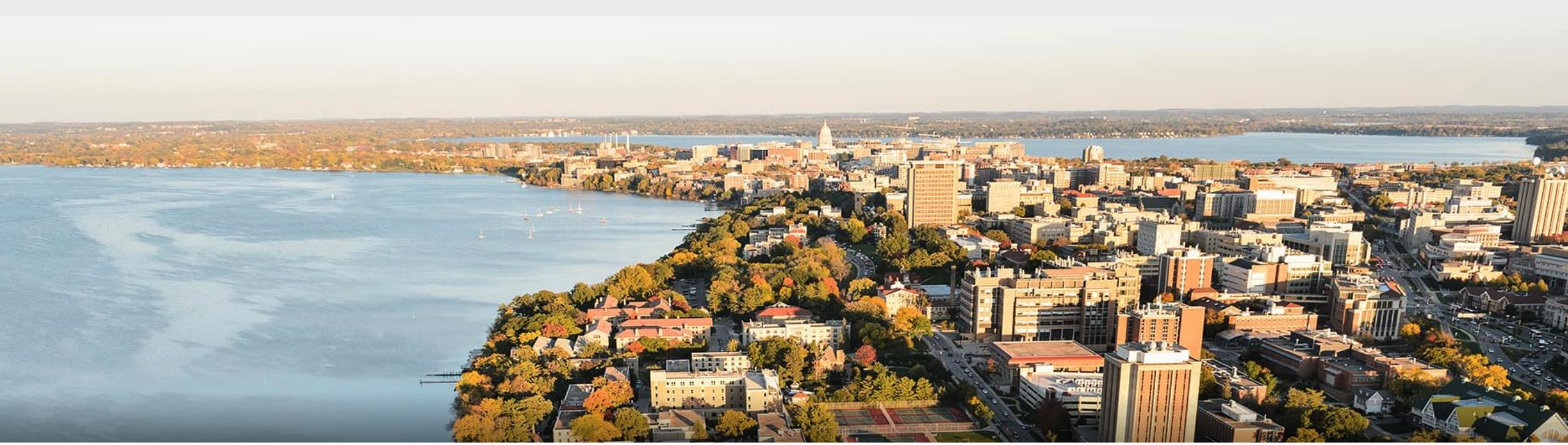
AI Alignment: A Comprehensive Survey

Jiaming Ji[*,1] Tianyi Qiu[*,1] Boyuan Chen[*,1] Borong Zhang[*,1] Hantao Lou[1] Kaile Wang[1] Yawen Duan[2] Zhonghao He[2] Jiayi Zhou[1] Zhaowei Zhang[1] Fanzhi Zeng[1] Juntao Dai[1] Xuehai Pan[1] Kwan Yee Ng Aidan O’Gara[5] Hua Xu[1] Brian Tse Jie Fu[4] Stephen McAleer[3] Yaodong Yang[1,✉] Yizhou Wang[1] Song-Chun Zhu[1] Yike Guo[4] Wen Gao[1]

[1]Peking University [2]University of Cambridge [3]Carnegie Mellon University
[4]Hong Kong University of Science and Technology [5]University of Southern California

Abstract

AI alignment aims to make AI systems behave in line with human intentions and values. As AI systems grow more capable, so do risks from misalignment. To provide a comprehensive and up-to-date overview of the alignment field, in this survey, we delve into the core concepts, methodology, and practice of alignment. First, we identify four principles as the key objectives of AI alignment: Robustness, Interpretability, Controllability, and Ethicality (**RICE**). Guided by these four principles, we outline the landscape of current alignment research and decompose them into two key components: **forward alignment** and **backward alignment**. The former aims to make AI systems aligned via alignment training, while the latter aims to gain evidence about the systems’ alignment and govern them appropriately to avoid exacerbating misalignment risks. On forward alignment, we discuss techniques for learning from feedback and learning under distribution shift. Specifically, we survey traditional preference modeling methods and reinforcement learning from human feedback, and further discuss potential frameworks to reach scalable oversight for tasks where effective human oversight is hard to obtain. Within learning under distribution shift, we also cover data distribution interventions such as adversarial training that help expand the distribution of



Break & Questions

Outline

- **Alignment: Review and Why Does It Work?**
  - Alignment Review, Failures of supervised learning, knowledge-seeking interactions, abstains
- **Variations + Open Questions**
  - Direct Preference Optimization (DPO), RLAIF, other techniques
- **Efficient Training**
  - Scale, memory optimization (FlashAttention), parallelism, heterogenous training

Training Foundation Models: **Scale**

Llama family of models,

- *"we estimate that we used 2048 A100-80GB for a period of approximately 5 months to develop our models"*

OPT (Open Pre-trained Transformers),

- *"training OPT-175B on 992 80GB A100 GPUs"*

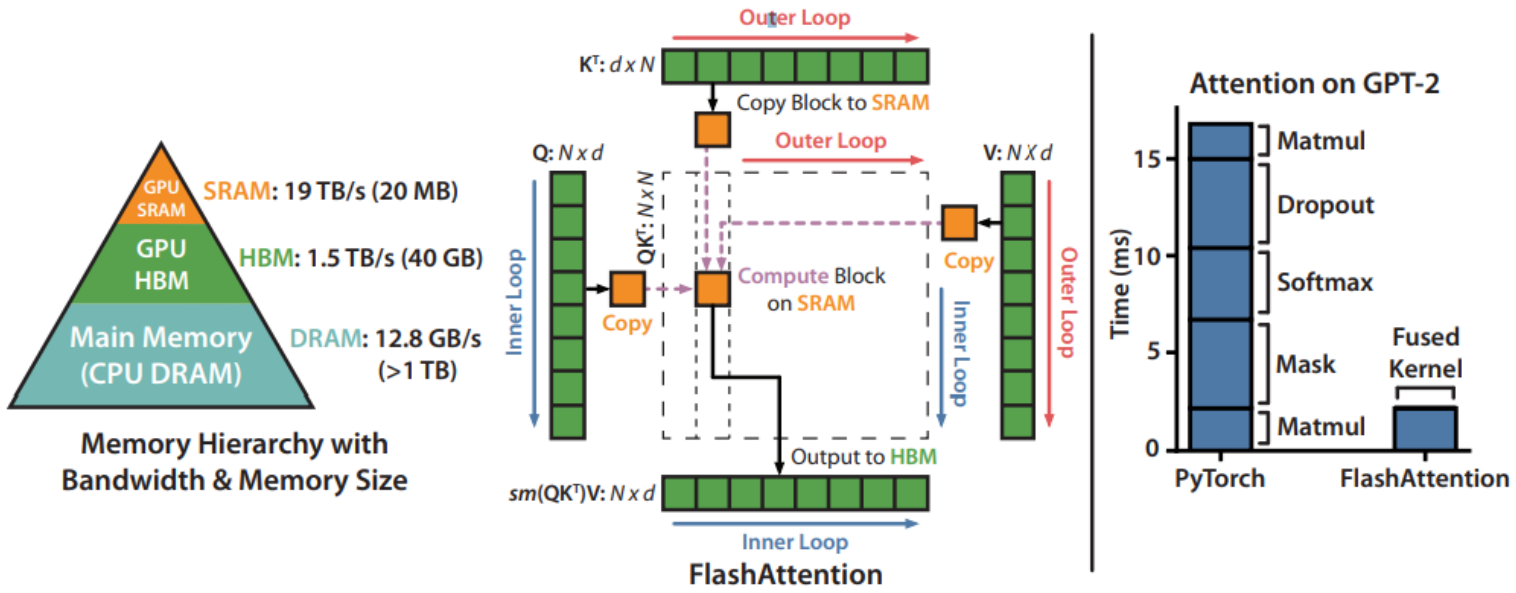
	GPU Type	GPU Power consumption	GPU-hours	Total power consumption	Carbon emitted (tCO_2eq)
OPT-175B	A100-80GB	400W	809,472	356 MWh	137
BLOOM-175B	A100-80GB	400W	1,082,880	475 MWh	183
LLaMA-7B	A100-80GB	400W	82,432	36 MWh	14
LLaMA-13B	A100-80GB	400W	135,168	59 MWh	23
LLaMA-33B	A100-80GB	400W	530,432	233 MWh	90
LLaMA-65B	A100-80GB	400W	1,022,362	449 MWh	173

Touvron et al, 23

Training Foundation Models: **GPU Usage**

Even for each GPU, there's additional considerations

- A little bit of fast memory, lots of slower memory
- Avoid using slow memory when possible
  - FlashAttention: Tiling + computing tricks

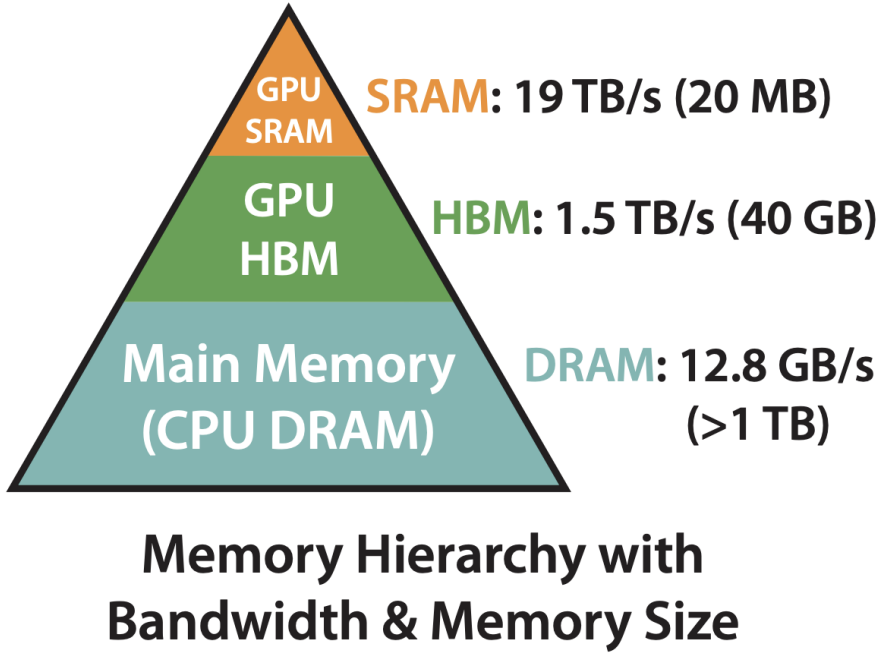


Dao et al '22

Flash Attention

Idea for FlashAttention

- Different kinds of GPU memory



Memory Hierarchy with Bandwidth & Memory Size

  - Fast: on-chip SRAM
    - But very little of this: 192KB for each of ~100 processors for an A100 (20MB)
  - Slow(er): HBM
    - But lots: 40-80GB for an A100

- **Goal**: use fast as much as possible, avoid moving to HBM

Flash Attention: **Basic Idea**

Will use two tricks for higher efficiency

- Tiling and re-computing.

First, recall standard attention

- Will use HBM memory repeatedly
  - Lots of reads and writes:

Algorithm 0 Standard Attention Implementation

Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ in HBM.

1: Load $\mathbf{Q}, \mathbf{K}$ by blocks from HBM, compute $\mathbf{S} = \mathbf{Q}\mathbf{K}^\top$, write $\mathbf{S}$ to HBM.
2: Read $\mathbf{S}$ from HBM, compute $\mathbf{P} = \mathrm{softmax}(\mathbf{S})$, write $\mathbf{P}$ to HBM.
3: Load $\mathbf{P}$ and $\mathbf{V}$ by blocks from HBM, compute $\mathbf{O} = \mathbf{P}\mathbf{V}$, write $\mathbf{O}$ to HBM.
4: Return $\mathbf{O}$.

Flash Attention: **Tiling**

Will use two tricks for higher efficiency

- Tiling and re-computing.

How do we avoid writing and reading from HBM?

- A: don't load the whole thing, use custom **tiling** and save the pieces (small). Standard version

$$m(x) := \max_i \; x_i, \quad f(x) := \begin{bmatrix} e^{x_1 - m(x)} & \dots & e^{x_B - m(x)} \end{bmatrix}, \quad \ell(x) := \sum_i f(x)_i, \quad \text{softmax}(x) := \frac{f(x)}{\ell(x)}.$$

- Tiling version: two components (can extend)

$$m(x) = m(\begin{bmatrix} x^{(1)} & x^{(2)} \end{bmatrix}) = \max(m(x^{(1)}), m(x^{(2)})), \quad f(x) = \begin{bmatrix} e^{m(x^{(1)}) - m(x)} f(x^{(1)}) & e^{m(x^{(2)}) - m(x)} f(x^{(2)}) \end{bmatrix},$$

$$\ell(x) = \ell(\begin{bmatrix} x^{(1)} & x^{(2)} \end{bmatrix}) = e^{m(x^{(1)}) - m(x)} \ell(x^{(1)}) + e^{m(x^{(2)}) - m(x)} \ell(x^{(2)}), \quad \text{softmax}(x) = \frac{f(x)}{\ell(x)}.$$

Flash Attention: **Recomputing**

Will use two tricks for higher efficiency

- Tiling and re-computing.

How do we avoid writing and reading from HBM?

- A: don't load the whole thing, use custom **tiling** and save the pieces

"Tiling enables us to implement our algorithm in one CUDA kernel, loading input from HBM, performing all the computation steps (matrix multiply, softmax, optionally masking and dropout, matrix multiply), then write the result back to HBM (masking and dropout in Appendix B). This avoids repeatedly reading and writing of inputs and outputs from and to HBM."

Don't we need to store full S, P for backwards pass, anyway?

- A: **No!** Can recompute on the fly S, P on the fly

Flash Attention: **Tradeoffs?**

Will use two tricks for higher efficiency

- Tiling and re-computing.

What's the tradeoff?

- Using tiling and computing/re-computing things normally trades off **memory consumption** for **speed**

- **But…** by reducing memory consumption, we can stick to fast memory only
  - And this makes us **much faster**
  - So **no tradeoff** at all (except for needing custom CUDA kernels ☺)

Flash Attention: **Tradeoffs?**

Will use two tricks for higher efficiency

- Tiling and re-computing.

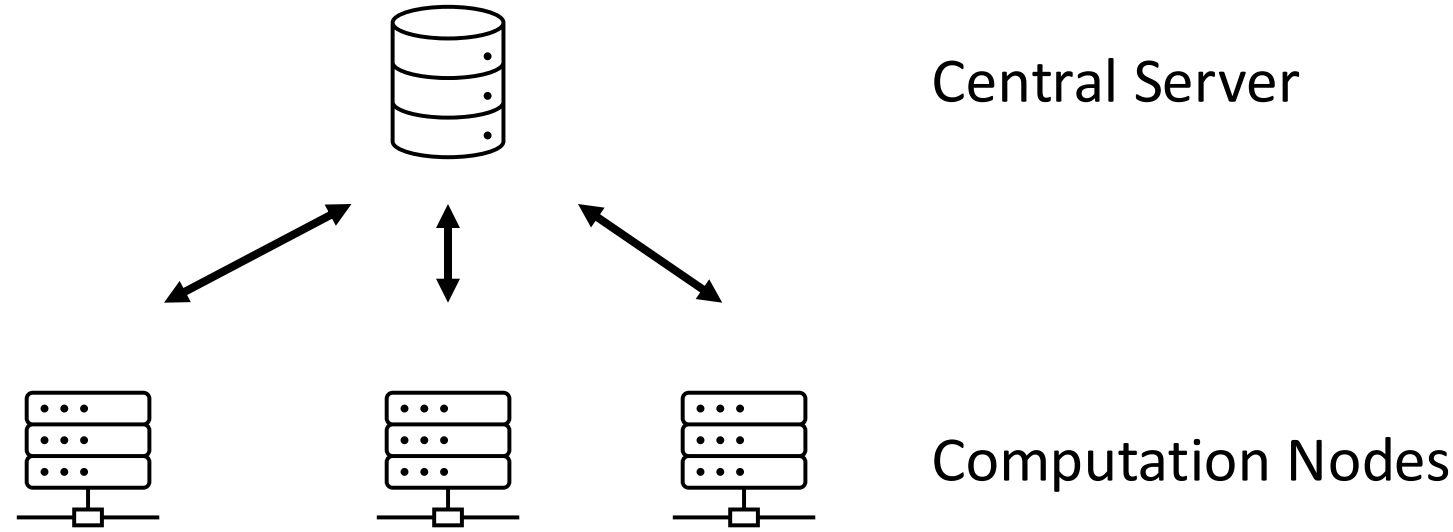
Results:

Model implementations	OpenWebText (ppl)	Training time (speedup)
GPT-2 small - Huggingface [87]	18.2	9.5 days (1.0×)
GPT-2 small - Megatron-LM [77]	18.2	4.7 days (2.0×)
GPT-2 small - FlashAttention	18.2	**2.7 days (3.5×)**
GPT-2 medium - Huggingface [87]	14.2	21.0 days (1.0×)
GPT-2 medium - Megatron-LM [77]	14.3	11.5 days (1.8×)
GPT-2 medium - FlashAttention	14.3	**6.9 days (3.0×)**

Training Foundation Models: **Parallelization**

Traditional approach is to **distribute** training loads

- Classic centralized distributed training
  - Synchronize each local gradient update
  - Send synchronized vector back to each node (lots of communication!)



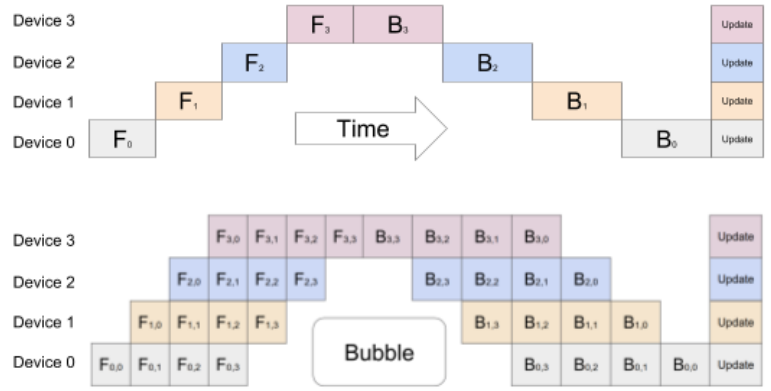
Training Foundation Models: **Parallelization**

Traditional approach is to **distribute** training loads

- This is by itself impossible (each node *can't* handle full model for large models)
- Need further parallelism:
  - **Data**: each node sees a different slice of data
  - **Weights/tensors**: chunks so no GPU sees whole model
  - **Pipeline**: only a few layers per GPU

- Great resource:

https://huggingface.co/blog/bloom-megatron-deepspeed



Top: The naive model parallelism strategy leads to severe underutilization due to the sequential nature of the network. Only one accelerator is active at a time. Bottom: GPipe divides the input mini-batch into smaller micro-batches, enabling different accelerators to work on separate micro-batches at the same time.

Bibliography

- Chip Huyen: https://huyenchip.com/2023/05/02/rlhf.html
- Nathan Lambert et al: https://huggingface.co/blog/rlhf
- Ouyang et al '22: 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, "Training language models to follow instructions with human feedback" (https://arxiv.org/abs/2203.02155)
- Lambert et al '24: Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, Hannaneh Hajishirzi , "RewardBench: Evaluating Reward Models for Language Modeling" (https://arxiv.org/abs/2403.13787)
- Schulman et al: John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov, "Proximal Policy Optimization Algorithms" (https://arxiv.org/abs/1707.06347)
- Yoav Golderbg: https://gist.github.com/yoavg/6bff0fecd65950898eba1bb321cfbd81
- Casper et al: Stephen Casper, Xander Davies, and many others, "Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback" (https://arxiv.org/abs/2307.15217)
- Rafailov et al '23: Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, Chelsea Finn, "Direct Preference Optimization: Your Language Model is Secretly a Reward Model" (https://arxiv.org/abs/2305.18290)
- Ji et al '24: Jiaming Ji, Tianyi Qiu, Boyuan Chen, Borong Zhang, Hantao Lou, Kaile Wang, Yawen Duan, Zhonghao He, Jiayi Zhou, Zhaowei Zhang, Fanzhi Zeng, Kwan Yee Ng, Juntao Dai, Xuehai Pan, Aidan O'Gara, Yingshan Lei, Hua Xu, Brian Tse, Jie Fu, Stephen McAleer, Yaodong Yang, Yizhou Wang, Song-Chun Zhu, Yike Guo, Wen Gao, "AI Alignment: A Comprehensive Survey" (https://arxiv.org/abs/2310.19852)
- Touvron et al '23: Hugo Touvron and many others, "LLaMA: Open and Efficient Foundation Language Models" (https://arxiv.org/abs/2302.13971)
- Stas Bekman: https://huggingface.co/blog/bloom-megatron-deepspeed
- Dao et al '22: Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, Christopher Ré, "FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness" (https://arxiv.org/abs/2205.14135)

Thank You!